Regular Separability of WSTS

Roland Meyer

joint work with Wojciech Czerwiński, Sławomir Lasota, Sebastian Muskalla, K Narayan Kumar, and Prakash Saivasan

IFIP WG 2.2, September 2018, Brno

Separability

Separability

Given $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from class $\mathcal{F}$.

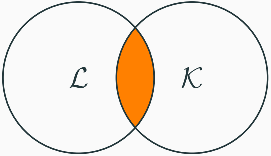
What is their relationship?

Separability

Given $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from class $\mathcal{F}$.

What is their relationship?

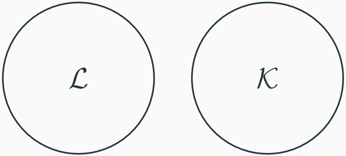
Case 1: $\mathcal{L} \cap \mathcal{K} \neq \varnothing$



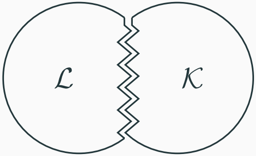
↳ Study $\mathcal{L} \cap \mathcal{K}$.

Separability

Case 2: $\mathcal{L} \cap \mathcal{K} = \varnothing$



vs.



Separability

Consider separability.

Separability of $\mathcal{F}$ by $\mathcal{S}$

Given: Languages $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from $\mathcal{F}$

Decide: Is there $\mathcal{R} \subseteq \Sigma^*$ from $\mathcal{S}$ such that

$$\mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing?$$

Separability

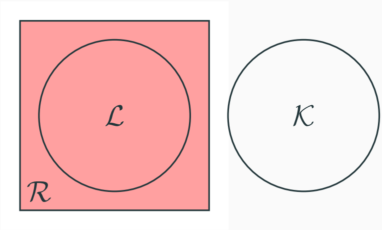
Consider separability.

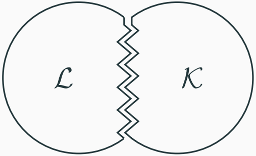
Separability of $\mathcal{F}$ by $\mathcal{S}$

Given: Languages $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from $\mathcal{F}$

Decide: Is there $\mathcal{R} \subseteq \Sigma^*$ from $\mathcal{S}$ such that

$$\mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing?$$





Separability

Consider separability.

Separability of $\mathcal{F}$ by $\mathcal{S}$	
Given:	Languages $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from $\mathcal{F}$
Decide:	Is there $\mathcal{R} \subseteq \Sigma^*$ from $\mathcal{S}$ such that
	$\mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing$?

Commonly studied:

- $\mathcal{S} \subsetneq \mathcal{F} = REG$

  e.g. $\mathcal{S}$ = Star-free languages

  ↳ Separability is decidable [Place, Zeitoun 2016].

Separability

Consider separability.

Separability of $\mathcal{F}$ by $\mathcal{S}$

Given: Languages $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from $\mathcal{F}$

Decide: Is there $\mathcal{R} \subseteq \Sigma^*$ from $\mathcal{S}$ such that

$$\mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing?$$

Commonly studied:

- $\mathcal{S} \subsetneq \mathcal{F} = \mathit{REG}$

  e.g. $\mathcal{S} =$ Star-free languages

  ↳ Separability is decidable [Place, Zeitoun 2016].

- $\mathcal{S} = \mathit{REG} \subsetneq \mathcal{F}$

  Regular separability.

Regular separability

Regular separability of $\mathcal{F}$

Given: Languages $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from $\mathcal{F}$

Decide: Is there $\mathcal{R} \subseteq \Sigma^*$ regular such that

$$\mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing?$$

Observation:

Problem is symmetric in the input:

If $\quad \mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing$

then $\quad \mathcal{K} \subseteq \overline{\mathcal{R}}, \quad \mathcal{L} \cap \overline{\mathcal{R}} = \varnothing$.

↳ Call $\mathcal{L}, \mathcal{K}$ regularly separable if separator $\mathcal{R}$ exists.

Regular separability

Regular separability of $\mathcal{F}$

Given: Languages $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$ from $\mathcal{F}$

Decide: Is there $\mathcal{R} \subseteq \Sigma^*$ regular such that

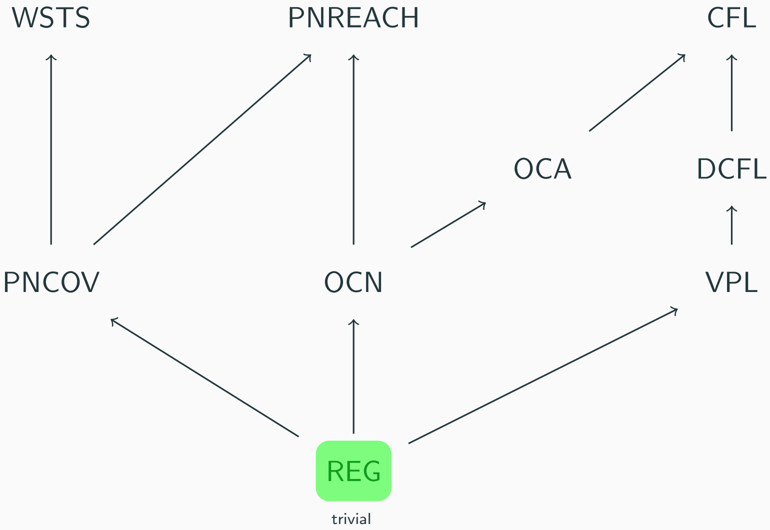
$$\mathcal{L} \subseteq \mathcal{R}, \quad \mathcal{K} \cap \mathcal{R} = \varnothing?$$

Disjointness is always necessary for (any kind of) separability.

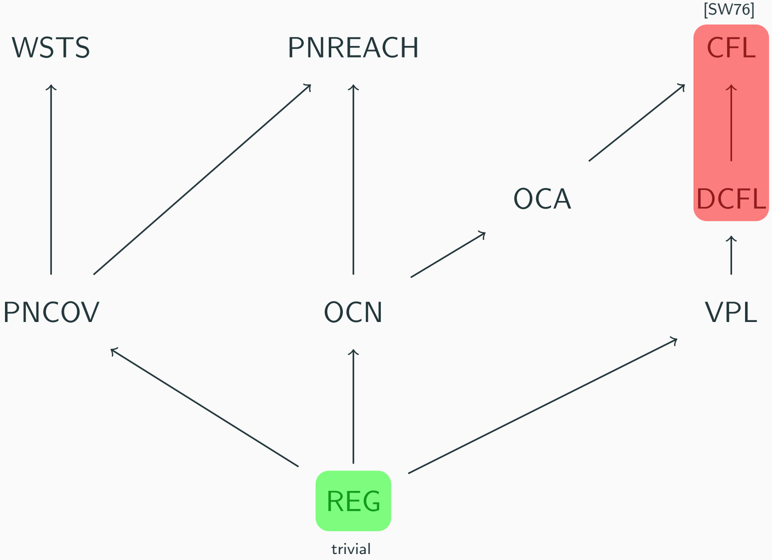
It is not always sufficient:

$$\mathcal{L} = a^n b^n, \quad \mathcal{K} = \overline{\mathcal{L}} \ .$$

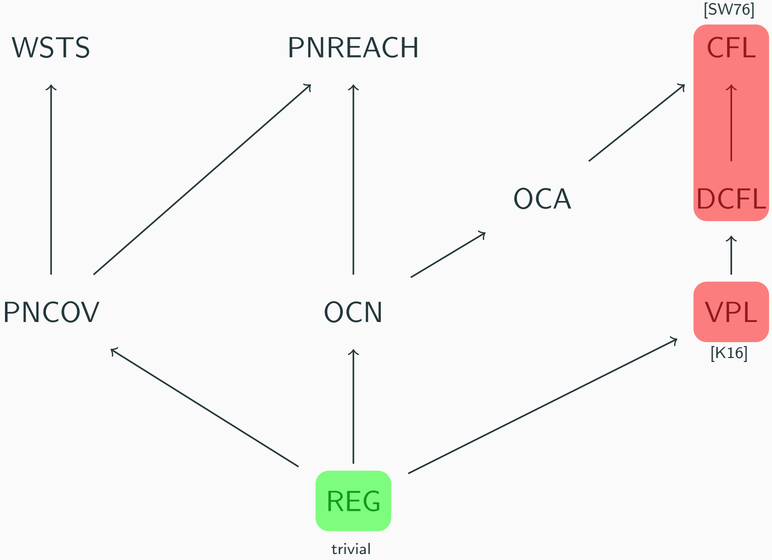
Regular separability — related work



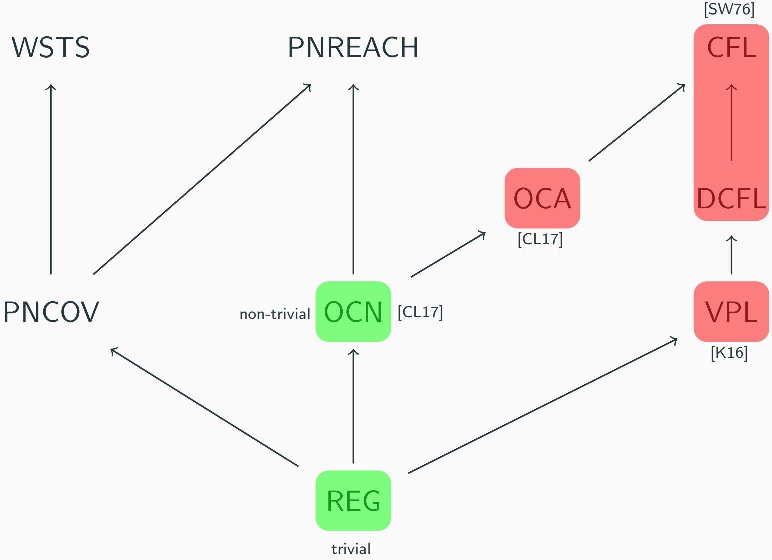
Regular separability — related work



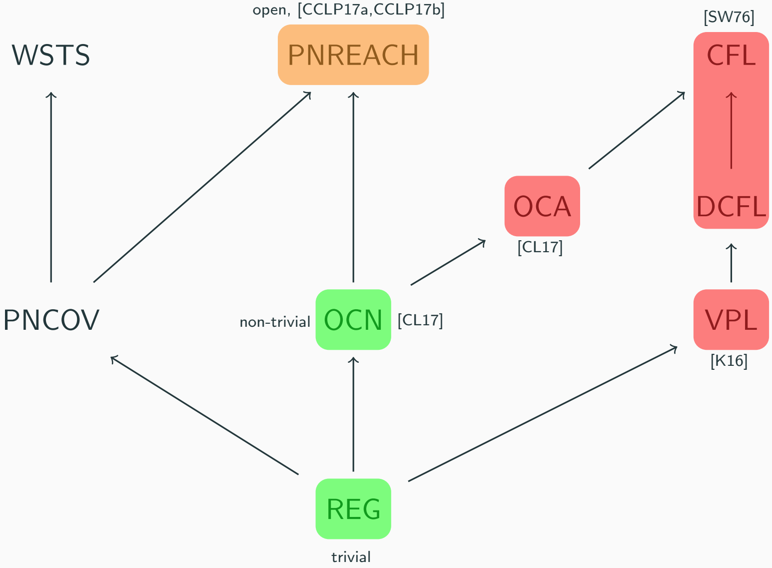
Regular separability — related work



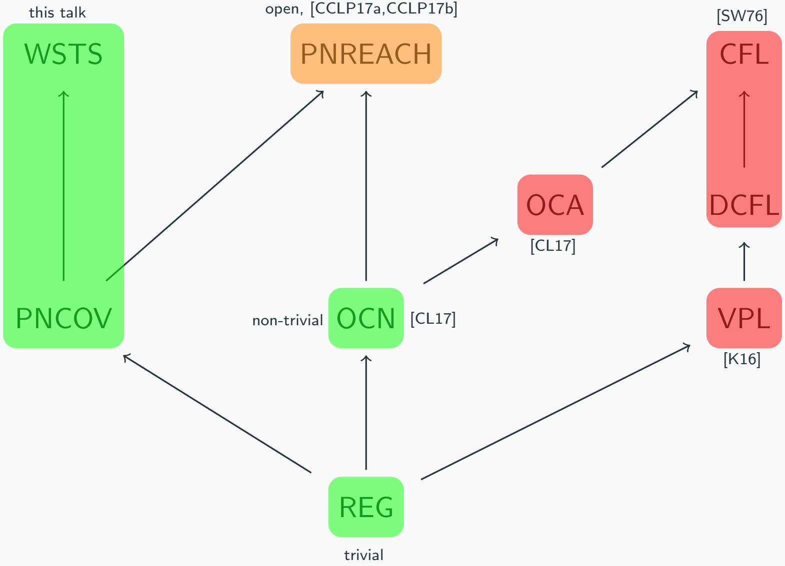
Regular separability — related work



Regular separability — related work



Regular separability — related work



The result

Well-structured transiton systems [F87,AJ93,ACJT96,FS01]

Consider labeled version of WSTS:

Well-structured transiton systems [F87,AJ93,ACJT96,FS01]

Consider labeled version of WSTS:

$$\mathcal{W} = (S, \leqslant, T, I, F).$$

- $(S, \leqslant)$ states well quasi ordering
- $T \subseteq S \times \Sigma \times S$ labeled transitions
- $I \subseteq S$ initial states
- $F \subseteq S$ final states, upward-closed

Well-structured transiton systems [F87,AJ93,ACJT96,FS01]

Consider labeled version of WSTS:

$$\mathcal{W} = (S, \leqslant, T, I, F).$$

- $(S, \leqslant)$ states well quasi ordering
- $T \subseteq S \times \Sigma \times S$ labeled transitions
- $I \subseteq S$ initial states
- $F \subseteq S$ final states, upward-closed
- Monotonicity / Simulation property:

$$\begin{array}{ccc} s' & \overset{a}{\dashrightarrow} & r' \ (\exists) \\ \curlyvee\!| & & \curlyvee\!| \\ s & \overset{a}{\longrightarrow} & r \end{array}$$

Well-structured transiton systems [F87,AJ93,ACJT96,FS01]

Consider labeled version of WSTS:

$$\mathcal{W} = (S, \leqslant, T, I, F).$$

$(S, \leqslant)$ states well quasi ordering

$T \subseteq S \times \Sigma \times S$ labeled transitions

$I \subseteq S$ initial states

$F \subseteq S$ final states, upward-closed

Coverability language

$$\mathcal{L}(\mathcal{W}) = \left\{ w \in \Sigma^* \;\middle|\; c_i \xrightarrow{w} c_f \text{ for some } c_i \in I, c_f \in F \right\}.$$

Well-structured transiton systems [F87,AJ93,ACJT96,FS01]

Consider labeled version of WSTS:

$$\mathcal{W} = (S, \leqslant, T, I, F).$$

Example 1:

Labeled Petri nets with covering acceptance condition yield WSTS

$$(\mathbb{N}^P, \leqslant^P, T, M_0, M_f\!\uparrow)\ .$$

Well-structured transiton systems [F87,AJ93,ACJT96,FS01]

Consider labeled version of WSTS:

$$\mathcal{W} = (S, \leqslant, T, I, F).$$

Example 1:

Labeled Petri nets with covering acceptance condition yield WSTS

$$(\mathbb{N}^P, \leqslant^P, T, M_0, M_f\!\uparrow) \ .$$

Example 2:

Labeled lossy channel systems (LCS) [AJ93] yield WSTS.

The result

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Applications and speculation

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Parallel program $P \parallel Q$ safe

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

$$\begin{array}{ll} & \text{Parallel program } P \parallel Q \text{ safe} \\ \text{iff} & \text{Language } \mathcal{L}(P \times Q) = \varnothing \end{array}$$

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

$$
\begin{aligned}
& \text{Parallel program } P \parallel Q \text{ safe} \\
\text{iff} \quad & \text{Language } \mathcal{L}(P \times Q) = \varnothing \\
\text{iff} \quad & \text{Language } \mathcal{L}(P) \cap \mathcal{L}(Q) = \varnothing
\end{aligned}
$$

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

		Parallel program $P \parallel Q$ safe
	iff	Language $\mathcal{L}(P \times Q) = \varnothing$
	iff	Language $\mathcal{L}(P) \cap \mathcal{L}(Q) = \varnothing$
(Theorem)	iff	$\exists$ regular separator of $\mathcal{L}(P)$ and $\mathcal{L}(Q)$

Compositional Safety Verification

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

		Parallel program $P \parallel Q$ safe
	iff	Language $\mathcal{L}(P \times Q) = \varnothing$
	iff	Language $\mathcal{L}(P) \cap \mathcal{L}(Q) = \varnothing$
(Theorem)	iff	$\exists$ regular separator of $\mathcal{L}(P)$ and $\mathcal{L}(Q)$
	iff	$\exists\ \mathcal{L}_1, \mathcal{L}_2$ regular with $\mathcal{L}(P) \subseteq \mathcal{L}_1$, $\mathcal{L}(Q) \subseteq \mathcal{L}_2$, and $\mathcal{L}_1 \cap \mathcal{L}_2 = \varnothing$.

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

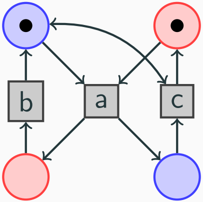
Applies to Petri net coverability, split set of places arbitrarily:

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:



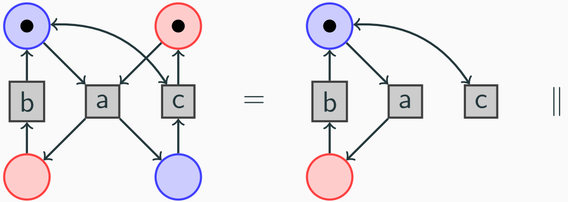
$=$

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:

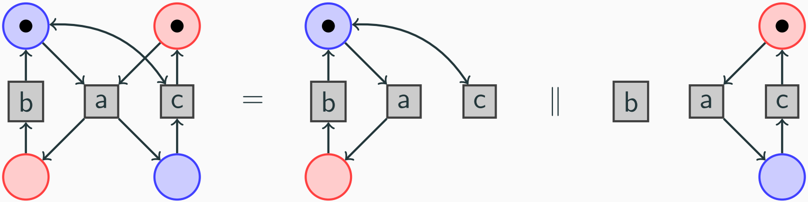


Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:

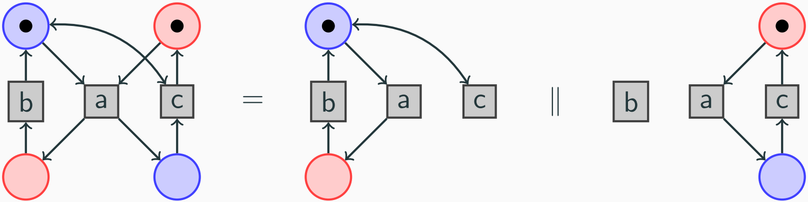


Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:



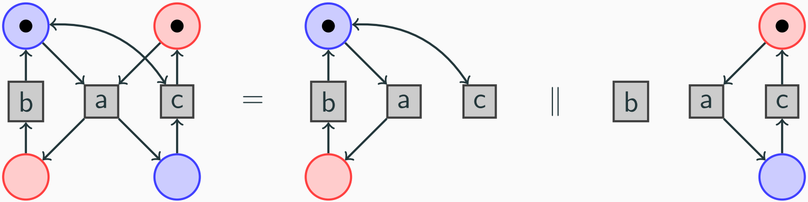
$$(ab + c)^*.a$$

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:



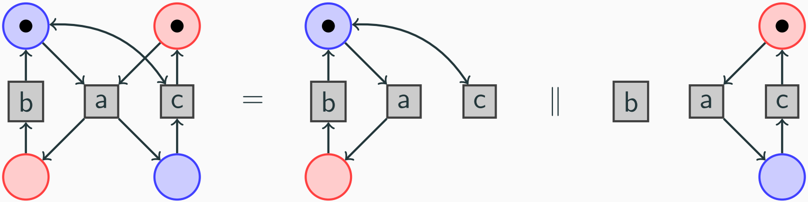
$(ab + c)^*.a \qquad \cap \qquad (ac)^* \sqcup\!\sqcup\, b^*$

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:



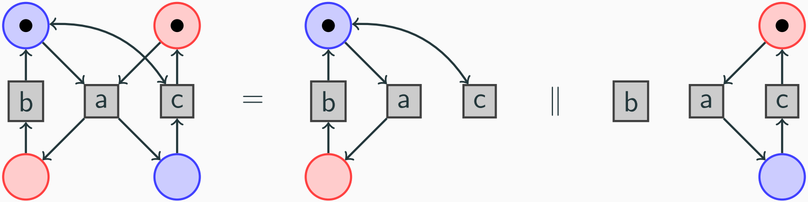
$$(ab+c)^*.a \quad \cap \quad (ac)^* \sqcup b^* = \varnothing$$

Compositional Safety Verification

Corollary

Regular approximations are complete for compositional verification of safety properties for parallel (well-structured) programs.

Applies to Petri net coverability, split set of places arbitrarily:



$$(ab+c)^*.a \quad \cap \quad (ac)^* \sqcup\!\sqcup\, b^* \; = \varnothing$$

Petri nets seem to have a regular type.

Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

Learn: Separator S of G and B.

Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

Learn: Separator S of G and B. $\Rightarrow$ Candidate for an invariant!

Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

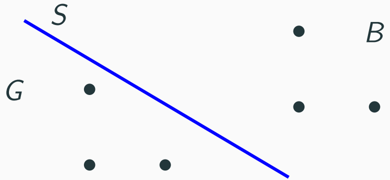
Learn: Separator S of G and B. $\Rightarrow$ Candidate for an invariant!

Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

Learn: Separator S of G and B. $\Rightarrow$ Candidate for an invariant!

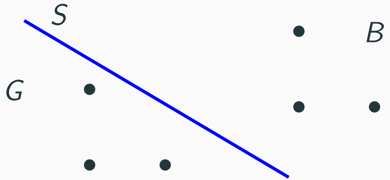


Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

Learn: Separator S of G and B. $\Rightarrow$ Candidate for an invariant!



Inductiveness problem: What if $x \in S$ but $y = post(x) \notin S$?

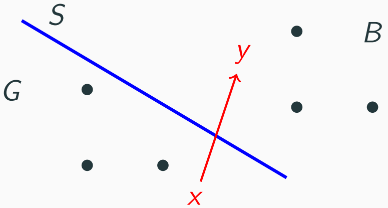
Should x be outside S or y be in S?

Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

Learn: Separator S of G and B. $\Rightarrow$ Candidate for an invariant!



Inductiveness problem: What if $x \in S$ but $y = post(x) \notin S$?

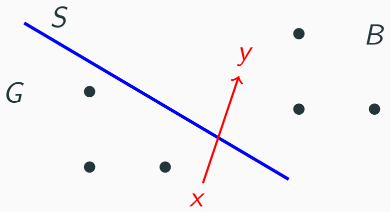
Should x be outside S or y be in S?

Learning-based verification without ICE

Learning invariants [Madhusudan, Neider et al. since 2014]

Given: Configurations G reachable from init, B leading to bad.

Learn: Separator S of G and B. $\Rightarrow$ Candidate for an invariant!



Inductiveness problem: What if $x \in S$ but $y = post(x) \notin S$?
Should x be outside S or y be in S?

Solution [Madhusudan, Neider et al.]:
Generalize learning algorithms to take into account pairs (x, y).

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Idea: Replace configurations by computations.

Learn a regular separator rather than an invariant.

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Idea: Replace configurations by computations.

Learn a regular separator rather than an invariant.

Learning-based verification with separators

Given: Computations G feasible in P, B feasible in Q.

Learn: Separator $\mathcal{R}$ of G and B.

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Idea: Replace configurations by computations.

Learn a regular separator rather than an invariant.

Learning-based verification with separators

Given: Computations G feasible in P, B feasible in Q.

Learn: Separator $\mathcal{R}$ of G and B. $\Rightarrow$ Candidate for $\mathcal{L}(P), \mathcal{L}(Q)$!

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Idea: Replace configurations by computations.

Learn a regular separator rather than an invariant.

Learning-based verification with separators

Given: Computations G feasible in P, B feasible in Q.

Learn: Separator $\mathcal{R}$ of G and B. $\Rightarrow$ Candidate for $\mathcal{L}(P), \mathcal{L}(Q)$!

Inductiveness problem:

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Idea: Replace configurations by computations.

Learn a regular separator rather than an invariant.

Learning-based verification with separators

Given: Computations G feasible in P, B feasible in Q.

Learn: Separator $\mathcal{R}$ of G and B. $\Rightarrow$ Candidate for $\mathcal{L}(P), \mathcal{L}(Q)$!

Inductiveness problem:

Inclusion of $\mathcal{L}(P)$ and disjointness from $\mathcal{L}(Q)$ have to be checked.

Learning-based verification without ICE

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Idea: Replace configurations by computations.

Learn a regular separator rather than an invariant.

Learning-based verification with separators

Given: Computations G feasible in P, B feasible in Q.

Learn: Separator $\mathcal{R}$ of G and B. $\Rightarrow$ Candidate for $\mathcal{L}(P), \mathcal{L}(Q)$!

Inductiveness problem:

Inclusion of $\mathcal{L}(P)$ and disjointness from $\mathcal{L}(Q)$ have to be checked.

But: No new framework needed!

Learning-based verification without ICE

$$G := \varnothing =: B$$

Learning-based verification without ICE

$$G := \varnothing =: B$$

$$\downarrow$$

Learn $\mathcal{R}$
separating G from B

Learning-based verification without ICE

$$G := \varnothing =: B$$

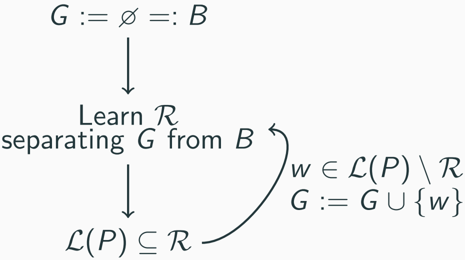
$\downarrow$

Learn $\mathcal{R}$
separating G from B

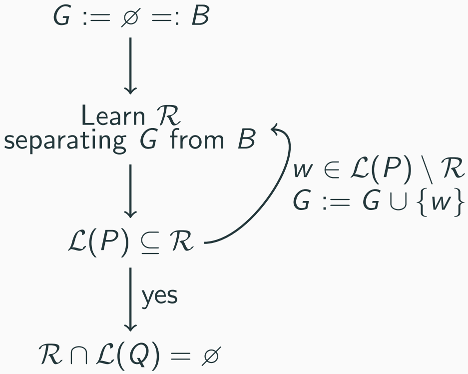
$\downarrow$

$$\mathcal{L}(P) \subseteq \mathcal{R}$$

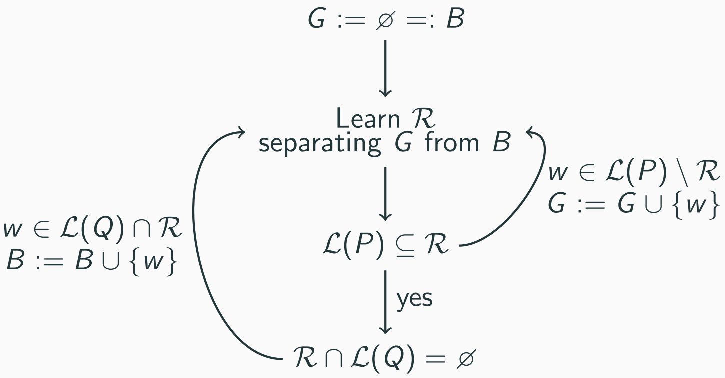
Learning-based verification without ICE



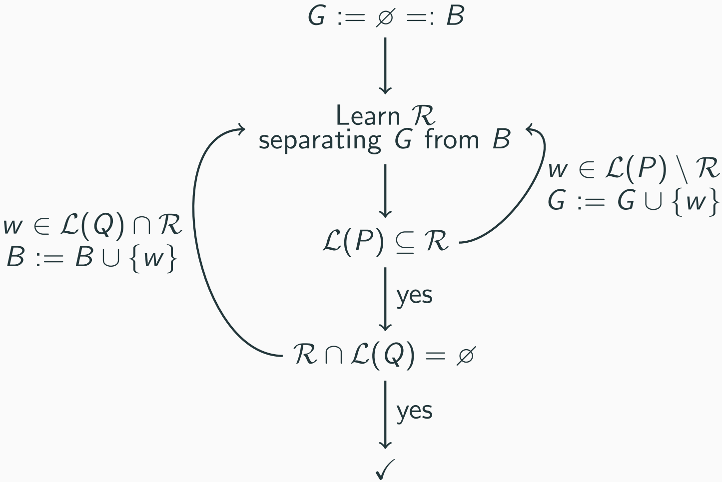
Learning-based verification without ICE



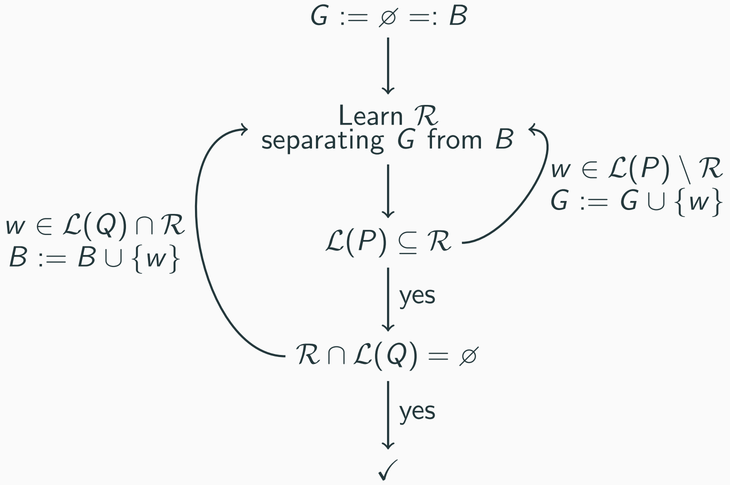
Learning-based verification without ICE



Learning-based verification without ICE



Learning-based verification without ICE



There is a dual algorithm learning $\mathcal{L}_1$ and $\mathcal{L}_2$ from above.

Interpolation-based regular model checking

Interpolation-based model checking [McMillan since 2003]

Given: Formulas $F = init \vee post(init)$, $G = pre^{\leqslant k}(bad)$.

Compute: Interpolant of F and G.

Interpolation-based regular model checking

Interpolation-based model checking [McMillan since 2003]

Given: Formulas $F = \mathit{init} \lor \mathit{post}(\mathit{init})$, $G = \mathit{pre}^{\leqslant k}(\mathit{bad})$.

Compute: Interpolant of F and G. $\Rightarrow$ Candidate for an invariant!

Interpolation-based regular model checking

Interpolation-based model checking [McMillan since 2003]

Given: Formulas $F = \mathit{init} \vee \mathit{post}(\mathit{init})$, $G = \mathit{pre}^{\leqslant k}(\mathit{bad})$.

Compute: Interpolant of F and G. $\Rightarrow$ Candidate for an invariant!

Needs representation for which interpolants can be computed.

Interpolation-based regular model checking

Interpolation-based model checking [McMillan since 2003]

Given: Formulas $F = \mathit{init} \vee \mathit{post}(\mathit{init})$, $G = \mathit{pre}^{\leqslant k}(\mathit{bad})$.

Compute: Interpolant of F and G. $\Rightarrow$ Candidate for an invariant!

Needs representation for which interpolants can be computed.

Craig's theorem 1957: First-order logic has interpolants.

Interpolation-based regular model checking

Separators are interpolants!

.

Interpolation-based regular model checking

Separators are interpolants!

Regular model checking [Abdulla et al. since 1997]

Analyze programs where configurations are words:

Interpolation-based regular model checking

Separators are interpolants!

Regular model checking [Abdulla et al. since 1997]

Analyze programs where configurations are words:

$$init, bad = \text{regular languages}$$

$$transitions = \text{regular transductions.}$$

Interpolation-based regular model checking

Separators are interpolants!

Regular model checking [Abdulla et al. since 1997]

Analyze programs where configurations are words:

$$\mathit{init}, \mathit{bad} = \text{regular languages}$$
$$\mathit{transitions} = \text{regular transductions.}$$

Since $\mathit{post}(\mathit{reg})$ regular, languages in McMillan's approach regular.

Interpolation-based regular model checking

Separators are interpolants!

Regular model checking [Abdulla et al. since 1997]

Analyze programs where configurations are words:

$$\mathit{init}, \mathit{bad} = \text{regular languages}$$

$$\mathit{transitions} = \text{regular transductions.}$$

Since $\mathit{post}(\mathit{reg})$ regular, languages in McMillan's approach regular.

Separators trivially exist!

Interpolation-based regular model checking

Separators are interpolants!

Regular model checking [Abdulla et al. since 1997]

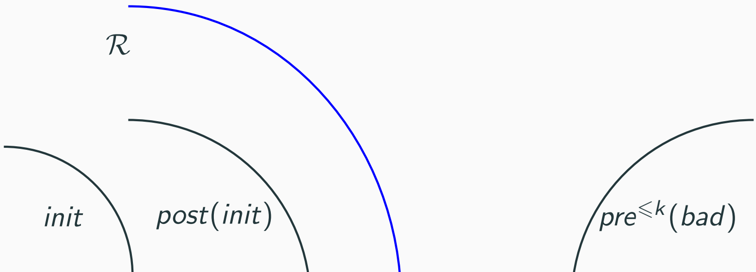
Analyze programs where configurations are words:

$$init, bad = \text{regular languages}$$

$$transitions = \text{regular transductions.}$$

Since $post(reg)$ regular, languages in McMillan's approach regular.

Separators trivially exist!



Interpolation of string-manipulating programs

Again: Separators may be the right thing!

Language-theoretic consequences

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Language-theoretic consequences

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

If a language and its complement are finitely branching WSTS languages, they are necessarily regular.

Language-theoretic consequences

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

If a language and its complement are finitely branching WSTS languages, they are necessarily regular.

Generalizes results for Petri nets [Kumar et al. 1998].

Language-theoretic consequences

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

Corollary

If a language and its complement are finitely branching WSTS languages, they are necessarily regular.

Generalizes results for Petri nets [Kumar et al. 1998].

Corollary

No subclass of finitely branching WSTS beyond REG is closed under complement.

Expressiveness results:
Languages of finitely branching WSTS

Our result - Recall

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

$\mathcal{W}$ finitely branching: I finite, $\mathrm{Post}_\Sigma(c)$ finite for all c.

Our result - Recall

Theorem

If two WSTS languages, one of them finitely branching, are disjoint, then they are regularly separable.

$\mathcal{W}$ finitely branching: I finite, $\mathrm{Post}_\Sigma(c)$ finite for all c.

How much of a restriction is it to assume finite branching?

What do we gain by assuming finite branching?

Expressibility I

Proposition

Languages of ω^2-WSTS
$\subseteq$ Languages of finitely branching WSTS.

$$\begin{aligned} & (S, \leqslant)\ \omega^2\text{-wqo} \\ \text{iff} \quad & \left(\mathcal{P}^{\downarrow}(S), \subseteq\right) \text{ wqo} \\ \text{iff} \quad & (S, \leqslant) \text{ does not embed the Rado order.} \end{aligned}$$

Our result applies to all WSTS of practical interest!

Expressibility II

Proposition

Languages of finitely branching WSTS
= Languages of deterministic WSTS.

Sufficient to show:

Theorem

If two WSTS languages, one of them deterministic, are disjoint, then they are regularly separable.

Proof sketch

Proof approach

Theorem

If two WSTS languages, one of them deterministic, are disjoint, then they are regularly separable.

Proof approach:

Relate separability to the existence of certain invariants.

Separability talks about the languages,

invariants talk about the state space!

Inductive invariant [Manna, Pnueli 1995]

Inductive invariant X

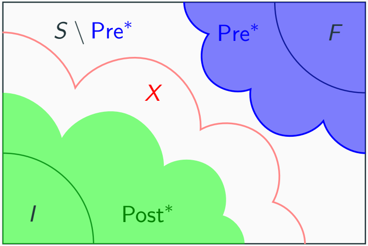
for WSTS $\mathcal{W}$:

(1) $X \subseteq S$ downward-closed

(2) $I \subseteq X$

(3) $F \cap X = \varnothing$

(4) $\mathrm{Post}_\Sigma(X) \subseteq X$



Inductive invariant [Manna, Pnueli 1995]

Inductive invariant X

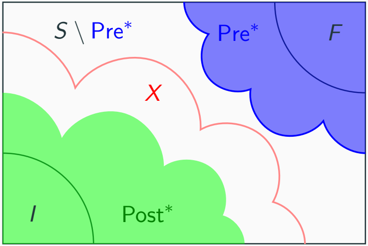
for WSTS $\mathcal{W}$:

(1) $X \subseteq S$ downward-closed

(2) $I \subseteq X$

(3) $F \cap X = \varnothing$

(4) $\mathrm{Post}_\Sigma(X) \subseteq X$



Lemma

$\mathcal{L}(\mathcal{W}) = \varnothing$ iff inductive invariant for $\mathcal{W}$ exists.

Proof approach

$$\mathcal{L}(\mathcal{W}_1), \mathcal{L}(\mathcal{W}_2) \text{ reg. sep} \overset{!}{\iff} \mathcal{L}(\mathcal{W}_1) \cap \mathcal{L}(\mathcal{W}_2) = \mathcal{L}(\mathcal{W}_1 \times \mathcal{W}_2) = \varnothing$$

Proof approach

$$\mathcal{L}(\mathcal{W}_1), \mathcal{L}(\mathcal{W}_2) \text{ reg. sep} \overset{!}{\iff} \mathcal{L}(\mathcal{W}_1) \cap \mathcal{L}(\mathcal{W}_2) = \mathcal{L}(\mathcal{W}_1 \times \mathcal{W}_2) = \varnothing$$

$$\Updownarrow$$

$\mathcal{W}_1 \times \mathcal{W}_2$ has inductive invariant

Proof approach

$$\mathcal{L}(\mathcal{W}_1), \mathcal{L}(\mathcal{W}_2) \text{ reg. sep} \overset{!}{\iff} \mathcal{L}(\mathcal{W}_1) \cap \mathcal{L}(\mathcal{W}_2) = \mathcal{L}(\mathcal{W}_1 \times \mathcal{W}_2) = \varnothing$$

?

$\mathcal{W}_1 \times \mathcal{W}_2$ has inductive invariant

Finitely represented invariants

The desired implication does not hold.

Call an invariant X finitely represented if $X = Q\downarrow$ for Q finite.

Finitely represented invariants

The desired implication does not hold.

Call an invariant X finitely represented if $X = Q\downarrow$ for Q finite.

Recall:

$(S, \leqslant)$ well quasi order (wqo)

iff upward-closed sets have finitely many minimal elements.

No such statement for downward-closed sets and maximal elements!

Finitely represented invariants

The desired implication does not hold.

Call an invariant X finitely represented if $X = Q\downarrow$ for Q finite.

We can show:

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

Proof approach II

$$\mathcal{L}(\mathcal{W}_1), \mathcal{L}(\mathcal{W}_2) \text{ reg. sep} \overset{!}{\iff} \mathcal{L}(\mathcal{W}_1) \cap \mathcal{L}(\mathcal{W}_2) = \mathcal{L}(\mathcal{W}_1 \times \mathcal{W}_2) = \varnothing$$

✓

$\mathcal{W}_1 \times \mathcal{W}_2$ has fin. rep. invariant

Proof approach II

$$\mathcal{L}(\mathcal{W}_1), \mathcal{L}(\mathcal{W}_2) \text{ reg. sep} \overset{!}{\iff} \mathcal{L}(\mathcal{W}_1) \cap \mathcal{L}(\mathcal{W}_2) = \mathcal{L}(\mathcal{W}_1 \times \mathcal{W}_2) = \varnothing$$

✓ ✗

$\mathcal{W}_1 \times \mathcal{W}_2$ has fin. rep. invariant

Ideals

Finitely represented invariants do not necessarily exist.

Solution: Ideals

Definition

For WSTS $\mathcal{W}$, let $\widehat{\mathcal{W}}$ be its ideal completion [KP92,BFM14,FG12].

Lemma

$\mathcal{L}(\mathcal{W}) = \mathcal{L}(\widehat{\mathcal{W}})$.
$\widehat{\mathcal{W}}$ is deterministic if so is $\mathcal{W}$.

Ideals

Finitely represented invariants do not necessarily exist.

Solution: Ideals

Definition

For WSTS $\mathcal{W}$, let $\widehat{\mathcal{W}}$ be its ideal completion [KP92,BFM14,FG12].

Lemma

$\mathcal{L}(\mathcal{W}) = \mathcal{L}(\widehat{\mathcal{W}})$.
$\widehat{\mathcal{W}}$ *is deterministic if so is* $\mathcal{W}$.

Proposition

If X *is an inductive invariant for* $\mathcal{W}$,
then its ideal decomposition $\textsc{Idec}(X)\!\downarrow$
is a finitely represented inductive invariant for $\widehat{\mathcal{W}}$.

Proof

Putting everything together:

If $\mathcal{W}_1, \mathcal{W}_2$ are disjoint, $\mathcal{W}_1 \times \mathcal{W}_2$ admits an invariant X.

Then $\textsc{Idec}(X)\downarrow$ is a finitely represented invariant for $\widehat{\mathcal{W}_1 \times \mathcal{W}_2} \cong \widehat{\mathcal{W}_1} \times \widehat{\mathcal{W}_2}$.

This finitely represented invariant gives rise to a regular separator.

Proof

Putting everything together:

If $\mathcal{W}_1, \mathcal{W}_2$ are disjoint, $\mathcal{W}_1 \times \mathcal{W}_2$ admits an invariant X.

Then $\textsc{Idec}(X)\downarrow$ is a finitely represented invariant for $\widehat{\mathcal{W}_1 \times \mathcal{W}_2} \cong \widehat{\mathcal{W}_1} \times \widehat{\mathcal{W}_2}$.

This finitely represented invariant gives rise to a regular separator.

We have shown:

Theorem

If two WSTS languages are disjoint,
one of them finitely branching or deterministic or ω^2,
then they are regularly separable.

Proof details:
From fin. rep. invariants to regular separators

From invariants to separability

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

From invariants to separability

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

Assume $Q\downarrow$ is an invariant.

Idea: Construct separating NFA with Q as states.

From invariants to separability

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

Definition

$\mathcal{A} = (Q, \to, Q_I, Q_F)$ where

From invariants to separability

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

Definition

$\mathcal{A} = (Q, \rightarrow, Q_I, Q_F)$ where

$$Q_I = \{(s, s') \in Q \mid (c, c') \leqslant (s, s') \text{ for some } (c, c') \text{ initial}\}$$

From invariants to separability

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

Definition

$\mathcal{A} = (Q, \rightarrow, Q_I, Q_F)$ where

$$Q_I = \{(s, s') \in Q \mid (c, c') \leqslant (s, s') \text{ for some } (c, c') \text{ initial}\}$$

$$Q_F = \{(s, s') \in Q \mid s \in F_1\}$$

From invariants to separability

Theorem

Let $\mathcal{W}_1, \mathcal{W}_2$ WSTS, $\mathcal{W}_2$ deterministic.

If $\mathcal{W}_1 \times \mathcal{W}_2$ admits a finitely represented inductive invariant, then $\mathcal{L}(\mathcal{W}_1)$ and $\mathcal{L}(\mathcal{W}_2)$ are regularly separable.

Definition

$\mathcal{A} = (Q, \to, Q_I, Q_F)$ where

$$Q_I = \{(s, s') \in Q \mid (c, c') \leqslant (s, s') \text{ for some } (c, c') \text{ initial}\}$$

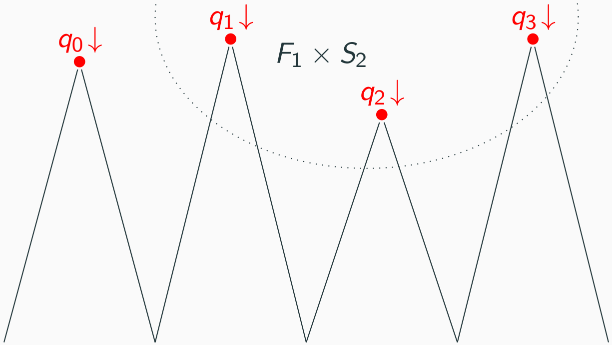
$$Q_F = \{(s, s') \in Q \mid s \in F_1\}$$

$$Q \ni (s, s') \xrightarrow[\text{in } \mathcal{A}]{a} (r, r') \in Q$$

$$(r, r') \geqslant (t, t')$$

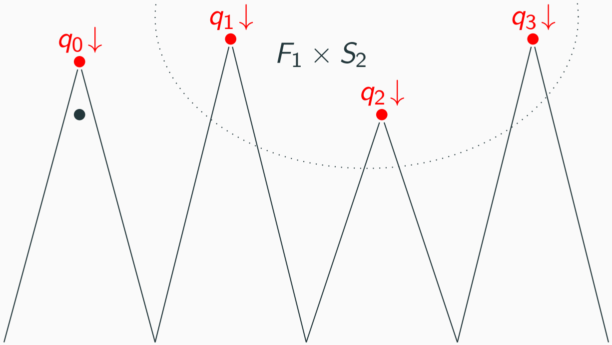
$$Q \ni (s, s') \xrightarrow[\text{in } \mathcal{W}_1 \times \mathcal{W}_2]{a} (t, t') \in S_1 \times S_2$$

Behavior of $\mathcal{A}$



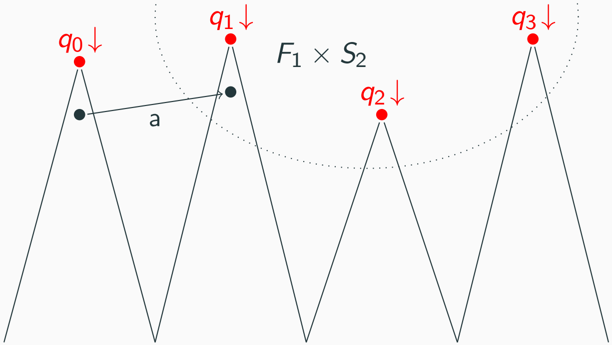
$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



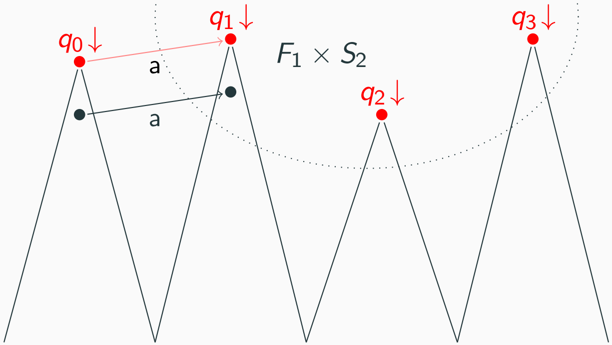
$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



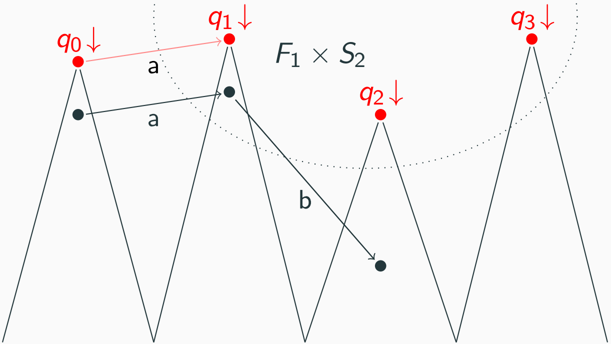
$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



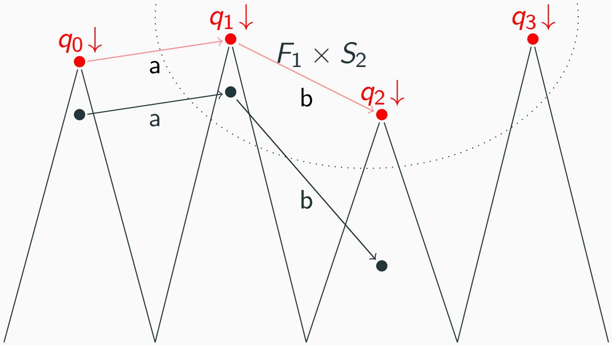
$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



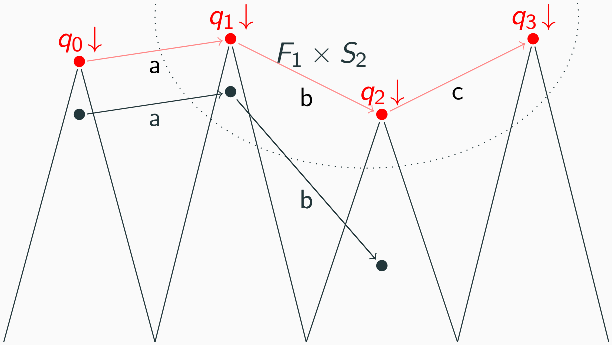
$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



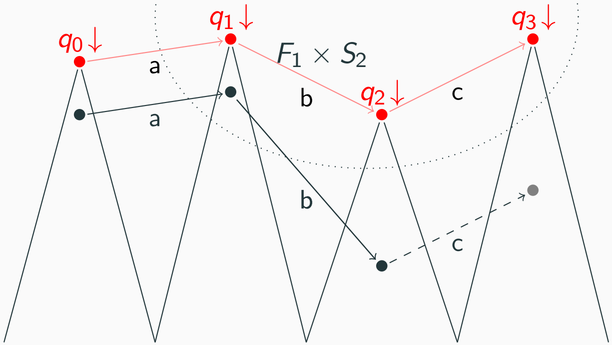
$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Behavior of $\mathcal{A}$



$\mathcal{A}$ over-approximates the behavior of the product system using the configurations from Q.

Proving separability: Inclusion

Lemma

$\mathcal{L}(\mathcal{W}_1) \subseteq \mathcal{L}(\mathcal{A})$.

Proving separability: Inclusion

Lemma

$\mathcal{L}(\mathcal{W}_1) \subseteq \mathcal{L}(\mathcal{A})$.

Proof.

Any run $c \xrightarrow{w} d$ of $\mathcal{W}_1$

synchronizes with *the* run of $\mathcal{W}_2$ for w

in the run $(c, c') \xrightarrow{w} (d, d')$ of $\mathcal{W}_1 \times \mathcal{W}_2$.

□

Proving separability: Inclusion

Lemma

$\mathcal{L}(\mathcal{W}_1) \subseteq \mathcal{L}(\mathcal{A})$.

Proof.

Any run $c \xrightarrow{w} d$ of $\mathcal{W}_1$

synchronizes with *the* run of $\mathcal{W}_2$ for w

in the run $(c, c') \xrightarrow{w} (d, d')$ of $\mathcal{W}_1 \times \mathcal{W}_2$.

This run can be over-approximated in $\mathcal{A}$.

$\square$

Proving separability: Inclusion

Lemma

$\mathcal{L}(\mathcal{W}_1) \subseteq \mathcal{L}(\mathcal{A})$.

Proof.

Any run $c \xrightarrow{w} d$ of $\mathcal{W}_1$

synchronizes with *the* run of $\mathcal{W}_2$ for w

in the run $(c, c') \xrightarrow{w} (d, d')$ of $\mathcal{W}_1 \times \mathcal{W}_2$.

This run can be over-approximated in $\mathcal{A}$.

If d is final in $\mathcal{W}_1$,

the over-approximation of (d, d') is final in $\mathcal{A}$. □

Proving separability: Disjointness

Lemma

$\mathcal{L}(\mathcal{W}_2) \cap \mathcal{L}(\mathcal{A}) = \varnothing$.

Proving separability: Disjointness

Lemma

$\mathcal{L}(\mathcal{W}_2) \cap \mathcal{L}(\mathcal{A}) = \varnothing$.

Proof.

Any run of $\mathcal{A}$ for w over-approximates

in the second component the unique run of $\mathcal{W}_2$ for w.

□

Proving separability: Disjointness

Lemma

$\mathcal{L}(\mathcal{W}_2) \cap \mathcal{L}(\mathcal{A}) = \varnothing$.

Proof.

Any run of $\mathcal{A}$ for w over-approximates

in the second component the unique run of $\mathcal{W}_2$ for w.

If $w \in \mathcal{L}(\mathcal{W}_2) \cap \mathcal{L}(\mathcal{A})$

then some run of $\mathcal{A}$ reaches a state (q, q') with

- q final in $\mathcal{W}_1$ (def. of Q_F)
- q' final in $\mathcal{W}_2$ ($w \in \mathcal{L}(\mathcal{W}_2)$ + argument above).

□

Proving separability: Disjointness

Lemma

$\mathcal{L}(\mathcal{W}_2) \cap \mathcal{L}(\mathcal{A}) = \varnothing$.

Proof.

Any run of $\mathcal{A}$ for w over-approximates

in the second component the unique run of $\mathcal{W}_2$ for w.

If $w \in \mathcal{L}(\mathcal{W}_2) \cap \mathcal{L}(\mathcal{A})$

then some run of $\mathcal{A}$ reaches a state (q, q') with

- q final in $\mathcal{W}_1$ (def. of Q_F)

- q' final in $\mathcal{W}_2$ ($w \in \mathcal{L}(\mathcal{W}_2)$ + argument above).

Contradiction to $(F_1 \times F_2) \cap Q{\downarrow} = \varnothing$! □

Proof details:
The ideal completion and fin. rep. invariants

Finitely represented invariants

Lemma

Let $U \subseteq S$ be an upward-closed set in a wqo.

There is a finite set U_{min} such that $U = U_{min}\uparrow$.

A similar result for downward-closed subsets and maximal elements does not hold.

Finitely represented invariants

Lemma

Let $U \subseteq S$ be an upward-closed set in a wqo.

There is a finite set U_{min} such that $U = U_{min}\uparrow$.

A similar result for downward-closed subsets and maximal elements does not hold.

Example:

Consider $\mathbb{N}$ in $(\mathbb{N}, \leqslant)$

Intuitively, $\mathbb{N} = \omega\downarrow$.

Finitely represented invariants

Lemma

Let $U \subseteq S$ be an upward-closed set in a wqo.

There is a finite set U_{min} such that $U = U_{min}\uparrow$.

A similar result for downward-closed subsets and maximal elements does not hold.

Consequence:

Finitely represented invariants may not exist!

Solution:

Move to a language-equivalent system for which they always exist.

Ideals

Let $(S, \leqslant)$ be a wqo

An ideal $\mathcal{I} \subseteq S$ is a set that is

- non-empty
- downward-closed

Ideals

Let $(S, \leqslant)$ be a wqo

An ideal $\mathcal{I} \subseteq S$ is a set that is

- non-empty
- downward-closed
- directed: $\forall x, y \in \mathcal{I}\, \exists z \in \mathcal{I} \colon x \leqslant z, y \leqslant z$.

Ideals

Let $(S, \leqslant)$ be a wqo

An ideal $\mathcal{I} \subseteq S$ is a set that is

- non-empty
- downward-closed
- directed: $\forall x, y \in \mathcal{I}\, \exists z \in \mathcal{I}\colon x \leqslant z, y \leqslant z$.

Example 1:

For each $c \in S$, $c{\downarrow}$ is an ideal.

Ideals

Let $(S, \leqslant)$ be a wqo

An ideal $\mathcal{I} \subseteq S$ is a set that is

- non-empty
- downward-closed
- directed: $\forall x, y \in \mathcal{I}\, \exists z \in \mathcal{I} \colon x \leqslant z, y \leqslant z$.

Example 2:

Consider $(\mathbb{N}^k, \leqslant)$

The ideals are the sets $u\!\downarrow$ for $u \in (\mathbb{N} \cup \{\omega\})^k$.

Ideal decomposition

Lemma ([Kabil, Pouzet 1992])

Let $(S, \leqslant)$ *be a wqo.*

For $D \subseteq S$ *downward closed, let* $\textsc{Idec}(D)$ *be the set of* *inclusion-maximal ideals in* D*.*

$\textsc{Idec}(D)$ *is unique, finite, and we have*

$$D = \bigcup \textsc{Idec}(D) \ .$$

Ideal completion

Definition ([FG12,BFM14])

Let $\mathcal{W} = (S, \leqslant, T, I, F)$ WSTS.

Its ideal completion is
$\widehat{\mathcal{W}} = (\{\mathcal{I} \subseteq S \mid \mathcal{I} \text{ ideal}\}, \subseteq, \widehat{T}, \textsc{IDec}(I\downarrow), \widehat{F})$ with

Ideal completion

Definition ([FG12,BFM14])

Let $\mathcal{W} = (S, \leqslant, T, I, F)$ WSTS.

Its ideal completion is
$\widehat{\mathcal{W}} = (\{\mathcal{I} \subseteq S \mid \mathcal{I} \text{ ideal}\}, \subseteq, \widehat{T}, \textsc{Idec}(I\downarrow), \widehat{F})$ with

$$\widehat{F} = \{\mathcal{I} \mid \mathcal{I} \cap F \neq \varnothing\}$$

Ideal completion

Definition ([FG12,BFM14])

Let $\mathcal{W} = (S, \leqslant, T, I, F)$ WSTS.

Its ideal completion is
$\widehat{\mathcal{W}} = (\{\mathcal{I} \subseteq S \mid \mathcal{I} \text{ ideal}\}, \subseteq, \widehat{T}, \textsc{Idec}(I{\downarrow}), \widehat{F})$ with

$$\widehat{F} = \{\mathcal{I} \mid \mathcal{I} \cap F \neq \varnothing\}$$

$\widehat{T}$ defined by $\mathrm{Post}_a^{\widehat{\mathcal{W}}}(\mathcal{I}) = \textsc{Idec}\big(\mathrm{Post}_a^{\mathcal{W}}(\mathcal{I}){\downarrow}\big)$.

Ideal completion

Definition ([FG12,BFM14])

Let $\mathcal{W} = (S, \leqslant, T, I, F)$ WSTS.

Its ideal completion is
$\widehat{\mathcal{W}} = (\{\mathcal{I} \subseteq S \mid \mathcal{I} \text{ ideal}\}, \subseteq, \widehat{T}, \textsc{Idec}(I\downarrow), \widehat{F})$ with

$$\widehat{F} = \{\mathcal{I} \mid \mathcal{I} \cap F \neq \varnothing\}$$

$\widehat{T}$ defined by $\mathrm{Post}_a^{\widehat{\mathcal{W}}}(\mathcal{I}) = \textsc{Idec}\big(\mathrm{Post}_a^{\mathcal{W}}(\mathcal{I})\downarrow\big)$.

Lemma

- $\widehat{\mathcal{W}}$ *finitely branching.*

Ideal completion

Definition ([FG12,BFM14])

Let $\mathcal{W} = (S, \leqslant, T, I, F)$ WSTS.

Its ideal completion is
$\widehat{\mathcal{W}} = (\{\mathcal{I} \subseteq S \mid \mathcal{I} \text{ ideal}\}, \subseteq, \widehat{T}, \textsc{IDec}(I\downarrow), \widehat{F})$ with

$$\widehat{F} = \{\mathcal{I} \mid \mathcal{I} \cap F \neq \varnothing\}$$

$\widehat{T}$ defined by $\text{Post}_a^{\widehat{\mathcal{W}}}(\mathcal{I}) = \textsc{IDec}\big(\text{Post}_a^{\mathcal{W}}(\mathcal{I})\downarrow\big)$.

Lemma

- $\widehat{\mathcal{W}}$ *finitely branching.*
- $\mathcal{W}$ *deterministic* $\implies$ $\widehat{\mathcal{W}}$ *deterministic.*

Ideal completion

Definition ([FG12,BFM14])

Let $\mathcal{W} = (S, \leqslant, T, I, F)$ WSTS.

Its ideal completion is
$\widehat{\mathcal{W}} = (\{\mathcal{I} \subseteq S \mid \mathcal{I} \text{ ideal}\}, \subseteq, \widehat{T}, \textsc{IDec}(I\downarrow), \widehat{F})$ with

$$\widehat{F} = \{\mathcal{I} \mid \mathcal{I} \cap F \neq \varnothing\}$$

$\widehat{T}$ defined by $\text{Post}_a^{\widehat{\mathcal{W}}}(\mathcal{I}) = \textsc{IDec}\big(\text{Post}_a^{\mathcal{W}}(\mathcal{I})\downarrow\big)$.

Lemma

- $\widehat{\mathcal{W}}$ *finitely branching.*
- $\mathcal{W}$ *deterministic* $\implies$ $\widehat{\mathcal{W}}$ *deterministic.*
- $\mathcal{L}(\widehat{\mathcal{W}}) = \mathcal{L}(\mathcal{W})$.

Using the ideal completion

Proposition

If X *is an inductive invariant for* $\mathcal{W}$*,*
then its ideal decomposition $\textsc{Idec}(X)\downarrow$
is a finitely represented inductive invariant for $\widehat{\mathcal{W}}$*.*

Using the ideal completion

Proposition

If X is an inductive invariant for $\mathcal{W}$,
then its ideal decomposition $\textsc{Idec}(X)\downarrow$
is a finitely represented inductive invariant for $\widehat{\mathcal{W}}$.

Proof.

Property of being an inductive invariant carries over.

Any set of the shape $\textsc{Idec}(Y)\downarrow$ is finitely-represented in $\widehat{\mathcal{W}}$. $\square$

Using the ideal completion

Proposition

If X is an inductive invariant for $\mathcal{W}$,
then its ideal decomposition $\textsc{Idec}(X)\downarrow$
is a finitely represented inductive invariant for $\widehat{\mathcal{W}}$.

Proof.

Property of being an inductive invariant carries over.

Any set of the shape $\textsc{Idec}(Y)\downarrow$ is finitely-represented in $\widehat{\mathcal{W}}$. $\square$

Result in particular applies to $\mathsf{Cover} = \mathsf{Post}^*(I_1 \times I_2)\downarrow$.

Using the ideal completion

Proposition

If X is an inductive invariant for $\mathcal{W}$,
then its ideal decomposition $\textsc{Idec}(X)\downarrow$
is a finitely represented inductive invariant for $\widehat{\mathcal{W}}$.

Proof.

Property of being an inductive invariant carries over.

Any set of the shape $\textsc{Idec}(Y)\downarrow$ is finitely-represented in $\widehat{\mathcal{W}}$. □

Result in particular applies to $\mathsf{Cover} = \mathsf{Post}^*(I_1 \times I_2)\downarrow$.

Remark: $\widehat{\mathcal{W}}$ is not necessarily a WSTS.

Separator size: The case of Petri nets

Separator size

Question:

Number of states of the separating automaton?

Separator size

Question:

Number of states of the separating automaton?

Consider Petri nets.

Separator size

Question:

Number of states of the separating automaton?

Consider Petri nets.

Problems:

1. Determinism.

Separator size

Question:

Number of states of the separating automaton?

Consider Petri nets.

Problems:

1. Determinism.
2. Size estimation on the ideal decomposition of an invariant.

Enforcing determinism

Given: Labeled Petri nets over Σ

$$N_A = (P_A, T_A, \lambda_A, \mathrm{in}_A, \mathrm{out}_A, M_{0A}, M_{fA})$$

$$N_B = (P_B, T_B, \lambda, \mathrm{in}_B, \mathrm{out}_B, M_{0B}, M_{fB}) \ .$$

See board.

Enforcing determinism

Given: Labeled Petri nets over Σ

$$N_A = (P_A, T_A, \lambda_A, \mathrm{in}_A, \mathrm{out}_A, M_{0A}, M_{fA})$$

$$N_B = (P_B, T_B, \lambda, \mathrm{in}_B, \mathrm{out}_B, M_{0B}, M_{fB})\ .$$

Construct: Labeled Petri nets over T_B

$$N_A^{-\lambda} = (P_A, T_A^{-\lambda}, \ell, \mathrm{in}_A^{-\lambda}, \mathrm{out}_A^{-\lambda}, M_{0A}, M_{fA})$$

$$N_B^{det} = (P_B, T_B, \mathrm{id}, \mathrm{in}_B, \mathrm{out}_B, , M_{0B}, M_{fB})\ .$$

See board.

Enforcing determinism

Given: Labeled Petri nets over Σ

$$N_A = (P_A, T_A, \lambda_A, \mathrm{in}_A, \mathrm{out}_A, M_{0A}, M_{fA})$$

$$N_B = (P_B, T_B, \lambda, \mathrm{in}_B, \mathrm{out}_B, M_{0B}, M_{fB})\ .$$

Construct: Labeled Petri nets over T_B

$$N_A^{-\lambda} = (P_A, T_A^{-\lambda}, \ell, \mathrm{in}_A^{-\lambda}, \mathrm{out}_A^{-\lambda}, M_{0A}, M_{fA})$$

$$N_B^{det} = (P_B, T_B, \mathrm{id}, \mathrm{in}_B, \mathrm{out}_B, , M_{0B}, M_{fB})\ .$$

$$\mathcal{L}(N_A \times N_B) = \lambda\Big(\mathcal{L}\Big(N_A^{-\lambda} \times N_B^{det}\Big)\Big)$$

Enforcing determinism

Given: Labeled Petri nets over Σ

$$N_A = (P_A, T_A, \lambda_A, \mathrm{in}_A, \mathrm{out}_A, M_{0A}, M_{fA})$$

$$N_B = (P_B, T_B, \lambda, \mathrm{in}_B, \mathrm{out}_B, M_{0B}, M_{fB}) \ .$$

Construct: Labeled Petri nets over T_B

$$N_A^{-\lambda} = (P_A, T_A^{-\lambda}, \ell, \mathrm{in}_A^{-\lambda}, \mathrm{out}_A^{-\lambda}, M_{0A}, M_{fA})$$

$$N_B^{det} = (P_B, T_B, \mathrm{id}, \mathrm{in}_B, \mathrm{out}_B, , M_{0B}, M_{fB}) \ .$$

If $\mathcal{R}$ separates $\mathcal{L}\left(N_A^{-\lambda}\right)$ and $\mathcal{L}(N_B^{det})$,
then $\lambda(\overline{\mathcal{R}})$ separates $\mathcal{L}(N_A)$ and $\mathcal{L}(N_B)$.

Obtaining an ideal decomposition of an invariant

First idea:

Coverability graph provides ideal decomposition of Cover.

Obtaining an ideal decomposition of an invariant

First idea:

Coverability graph provides ideal decomposition of Cover.

Problem:

It may be Ackermann-large.

Obtaining an ideal decomposition of an invariant

First idea:

Coverability graph provides ideal decomposition of Cover.

Problem:

It may be Ackermann-large.

Better idea:

Use ideal decomposition of $\mathbb{N}^k \setminus \mathrm{Pre}^*(M_{fA}\uparrow \times M_{fB}\uparrow)$.

Obtaining an ideal decomposition of an invariant

First idea:

Coverability graph provides ideal decomposition of Cover.

Problem:

It may be Ackermann-large.

Better idea:

Use ideal decomposition of $\mathbb{N}^k \setminus \mathrm{Pre}^*(M_{fA}\uparrow \times M_{fB}\uparrow)$.

Theorem ([Bozzelli, Ganty 2011])

$Pre^*(M_f\uparrow) = \{v_1, \ldots, v_k\}$ *with* k *and* $||v_i||_\infty$ *doubly exponential.*

The upper bound

Theorem (BG11)

$Pre^*(M_f\uparrow) = \{v_1, \ldots, v_k\}$ *with* k *and* $||v_i||_\infty$ *doubly exponential.*

Theorem (Upper bound)

Given two disjoint Petri nets, we can construct an NFA separating their coverability languages of triply-exponential size.

Upper vs. lower bound

Theorem (Upper bound)

Given two disjoint Petri nets, we can construct an NFA separating their coverability languages of triply-exponential size.

Theorem (Lower bound)

The disjoint Petri net coverability languages

$$\mathcal{L}_{0@2^{2^k}} \text{ and } \mathcal{L}_{1@2^{2^k}} \text{ over } \{0, 1\}$$

cannot be separated by a DFA of less than triply-exponential size.

Conclusion

Regular separability for WSTS languages

Theorem

If two WSTS languages are disjoint,
one of them finitely branching or deterministic or ω^2,
then they are regularly separable.

Open problems: Expressiveness

Non-Determinism:

Does non-determinism add to the expressiveness of WSTS:

Open problems: Expressiveness

Non-Determinism:

Does non-determinism add to the expressiveness of WSTS:

$$\text{deterministic WSTS languages} \quad \subsetneq \quad \text{all WSTS languages} \quad ?$$

Open problems: Expressiveness

Non-Determinism:

Does non-determinism add to the expressiveness of WSTS:

deterministic WSTS languages $\subsetneq$ all WSTS languages ?

Open: Infinitely branching WSTS over Rado order.

Open problems: Expressiveness

Non-Determinism:

Does non-determinism add to the expressiveness of WSTS:

deterministic WSTS languages $\subsetneq$ all WSTS languages ?

Open: Infinitely branching WSTS over Rado order.

Related problem:

ω^2-WSTS languages $\subsetneq$ deterministic WSTS languages ?

Open problems: Expressiveness

Non-Determinism:

Does non-determinism add to the expressiveness of WSTS:

deterministic WSTS languages $\subsetneq$ all WSTS languages ?

Open: Infinitely branching WSTS over Rado order.

Related problem:

ω^2-WSTS languages $\subsetneq$ deterministic WSTS languages ?

Complexity:

Tight bound on the separator size for Petri nets.

Open problems: Expressiveness

Non-Determinism:

Does non-determinism add to the expressiveness of WSTS:

deterministic WSTS languages $\subsetneq$ all WSTS languages ?

Open: Infinitely branching WSTS over Rado order.

Related problem:

ω^2-WSTS languages $\subsetneq$ deterministic WSTS languages ?

Complexity:

Tight bound on the separator size for Petri nets.

Replace homomorphism trick or show combinatorial magic.

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Should explain existing (un)decidability results.

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Should explain existing (un)decidability results.

An equivalence will not do (not one separator).

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Should explain existing (un)decidability results.

An equivalence will not do (not one separator).

ω*-regular separability of WSTS?*

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Should explain existing (un)decidability results.

An equivalence will not do (not one separator).

ω-regular separability of WSTS?

Regular separability is for safety verification.

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Should explain existing (un)decidability results.

An equivalence will not do (not one separator).

ω-regular separability of WSTS?

Regular separability is for safety verification.

Is there an ω-regular separability result for liveness verification?

Open problems: Theory of regular separability

Regular separability result:

Are disjoint WSTS languages always regularly separable?

Solved if non-determinism does not add expressiveness.

Fails for WBTS [Finkel et al. 2017], strictly larger class.

Myhill-Nerode-like characterization of regular separability:

Should explain existing (un)decidability results.

An equivalence will not do (not one separator).

ω-regular separability of WSTS?

Regular separability is for safety verification.

Is there an ω-regular separability result for liveness verification?

A similarly general result would be surprising

given the negative results for LCS [Abdulla, Jonsson 1996].

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Computing regular separators:

Compute separators from automata or WMSO formulas.

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Computing regular separators:

Compute separators from automata or WMSO formulas.
Interpolation algorithms rely on resolution proofs.

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Computing regular separators:

Compute separators from automata or WMSO formulas.

Interpolation algorithms rely on resolution proofs.

Proof systems for WSMO under development [Vojnar et al. 2017].

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Computing regular separators:

Compute separators from automata or WMSO formulas.
Interpolation algorithms rely on resolution proofs.
Proof systems for WSMO under development [Vojnar et al. 2017].

Verification:

Try out ideas for verification algorithms.

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Computing regular separators:

Compute separators from automata or WMSO formulas.

Interpolation algorithms rely on resolution proofs.

Proof systems for WSMO under development [Vojnar et al. 2017].

Verification:

Try out ideas for verification algorithms.

Iterated decomposition in the Petri net case open.

Open problems: Algorithms

There are not yet practical algorithms
for and based on separability :)

Computing regular separators:

Compute separators from automata or WMSO formulas.
Interpolation algorithms rely on resolution proofs.
Proof systems for WSMO under development [Vojnar et al. 2017].

Verification:

Try out ideas for verification algorithms.
Iterated decomposition in the Petri net case open.
Learning would benefit from extrapolation.

Open problems

Beyond regular separability?

Open problems

Beyond regular separability?

Beyond WSTS?

Thank you!

Questions?