Concurrent Kleene algebra, a done deal

Pomset languages and concurrent Kleene algebras

Séminaire Automates – November 24, 2017

Paul Brunet[1], Damien Pous[2], Georg Struth[3], Tobias Kappé[1], Alexandra Silva[1], Bas Luttik[4], and Fabio Zanasi[1]

University College London[1], ENS de Lyon – CNRS[2], University of Sheffield[3], Eindhoven University of Technology[4]

Kleene Algebra
Equivalence of sequential programs

$$(x := 1; y := 2); (x := y \oplus y := x) \qquad \equiv \qquad x := 1; (y := 2; x := y) \oplus (y := 2; y := x)$$

Kleene Algebra

Equivalence of sequential programs

$$(x := 1; y := 2); (x := y \oplus y := x) \quad \equiv \quad x := 1; (y := 2; x := y) \oplus (y := 2; y := x)$$

$$(x1 \cdot y2) \cdot (xy + yx)$$

Kleene Algebra

Equivalence of sequential programs

$$(x := 1; y := 2); (x := y \oplus y := x) \quad \equiv \quad x := 1; (y := 2; x := y) \oplus (y := 2; y := x)$$

$$(x1 \cdot y2) \cdot (xy + yx) = x1 \cdot (y2 \cdot (xy + yx)) \qquad \text{(associativity of } \cdot\text{)}$$

Kleene Algebra

Equivalence of sequential programs

$$(x := 1; y := 2); (x := y \oplus y := x) \qquad \equiv \qquad x := 1; (y := 2; x := y) \oplus (y := 2; y := x)$$

$$\begin{aligned}(x1 \cdot y2) \cdot (xy + yx) &= x1 \cdot (y2 \cdot (xy + yx)) && \text{(associativity of } \cdot\text{)}\\ &= x1 \cdot ((y2 \cdot xy) + (y2 \cdot yx)) && \text{(distributivity)}\end{aligned}$$

Kleene Algebra

Equivalence of sequential programs

A Kleene algebra is structure $\langle K, 0, 1, +, \cdot, \star \rangle$ such that:

1. $\langle K, 0, 1, +, \cdot \rangle$ is an idempotent semiring;
2. $\forall x \in K, 1 + x \cdot x^\star = x^\star$;
3. $\forall x, y, z \in K, x + y \cdot z \leq z \Rightarrow y^\star \cdot x \leq z$.

Theorem

$$\mathrm{KA} \vdash e = f \Leftrightarrow \mathcal{L}(e) = \mathcal{L}(f).$$

Kleene Algebra

Equivalence of sequential programs

Kleene Algebra

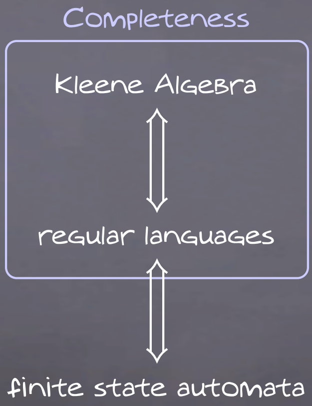
$\Updownarrow$

regular languages

$\Updownarrow$

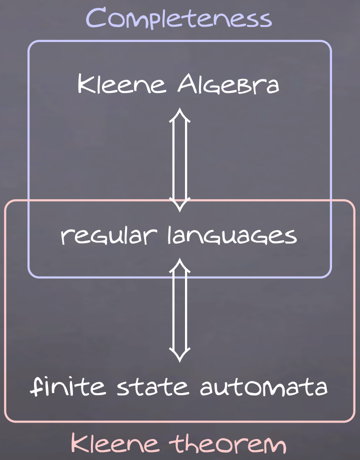
finite state automata

Kleene Algebra
Equivalence of sequential programs



Kleene Algebra

Equivalence of sequential programs



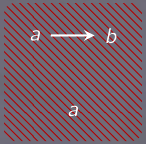
Concurrent Kleene Algebras

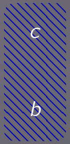
Equivalence of parallel programs

$$e, f \in \mathbb{E} ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^\star \mid e \parallel f$$

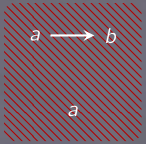
Bi-Kleene Algebra	Concurrent Kleene Algebra
$\Updownarrow$	$\Updownarrow$
series-rational pomset languages	down-closed series-rational languages
$\Updownarrow$	$\Updownarrow$
automata ?	automata ?

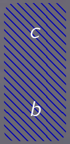
Pomsets

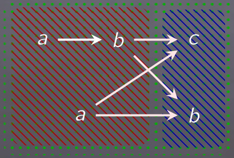
$P_1 =$ 

$P_2 =$ 

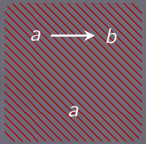
Pomsets

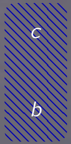
$P_1 =$ 

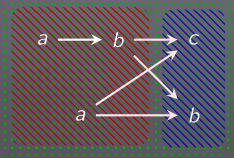
$P_2 =$ 

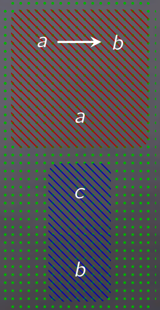
$P_1 \cdot P_2 =$ 

Pomsets

$P_1 =$ 

$P_2 =$ 

$P_1 \cdot P_2 =$ 

$P_1 \parallel P_2 =$ 

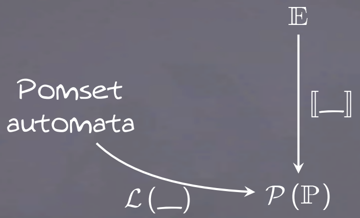
Outline

$\mathcal{P}(\mathbb{P})$

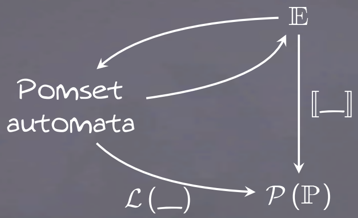
Outline

$$\mathbb{E} \xrightarrow{\llbracket - \rrbracket} \mathcal{P}(\mathbb{P})$$

Outline

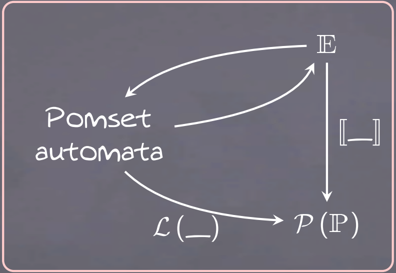


Outline



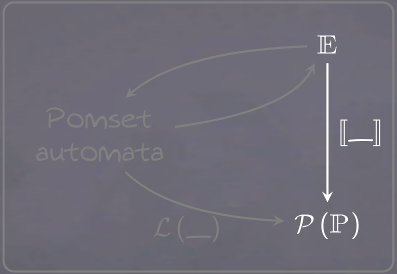
Outline

1. Brzozowski derivatives

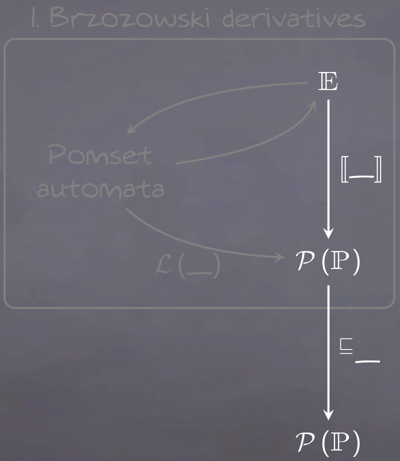


Outline

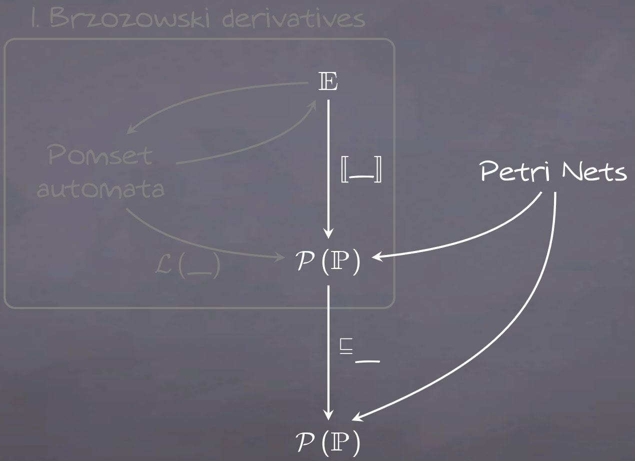
1. Brzozowski derivatives



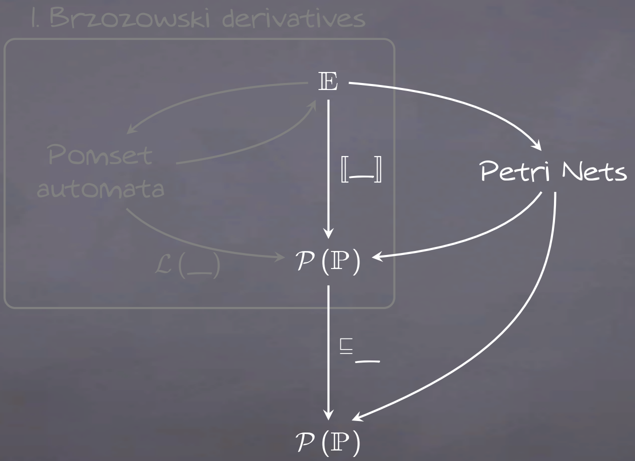
Outline


1. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
$[\![\_]\!]$
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\sqsubseteq\_$
$\mathcal{P}(\mathbb{P})$

Outline

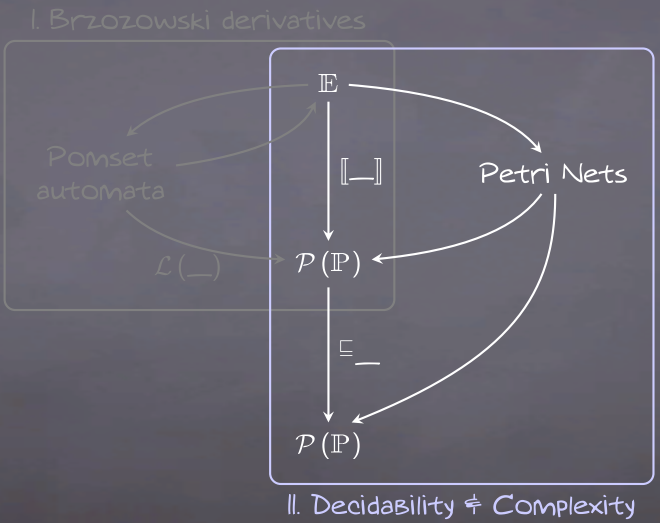

1. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
$[\![\_]\!]$
Petri Nets
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\sqsubseteq\_$
$\mathcal{P}(\mathbb{P})$

Outline



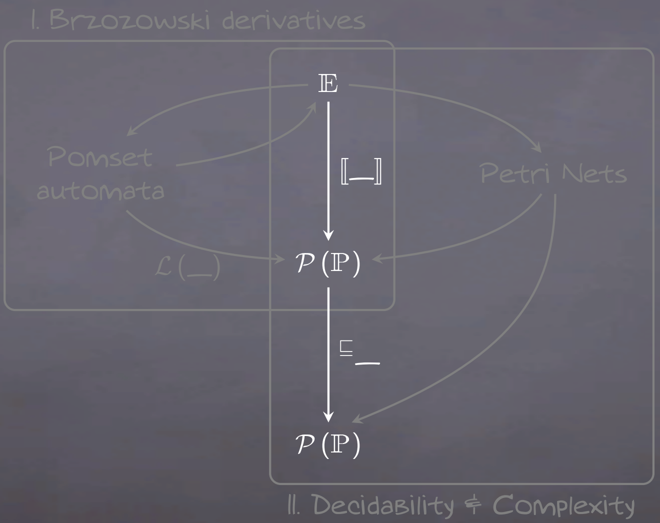
1. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
$[\![\_]\!]$
Petri Nets
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\sqsubseteq\_$
$\mathcal{P}(\mathbb{P})$

Outline



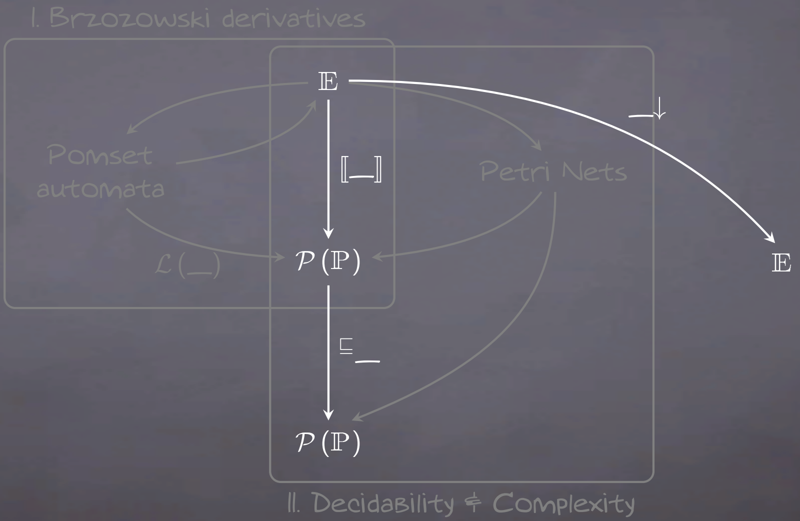
I. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
$[\![\_]\!]$
Petri Nets
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\sqsubseteq\_$
$\mathcal{P}(\mathbb{P})$
II. Decidability & Complexity

Outline



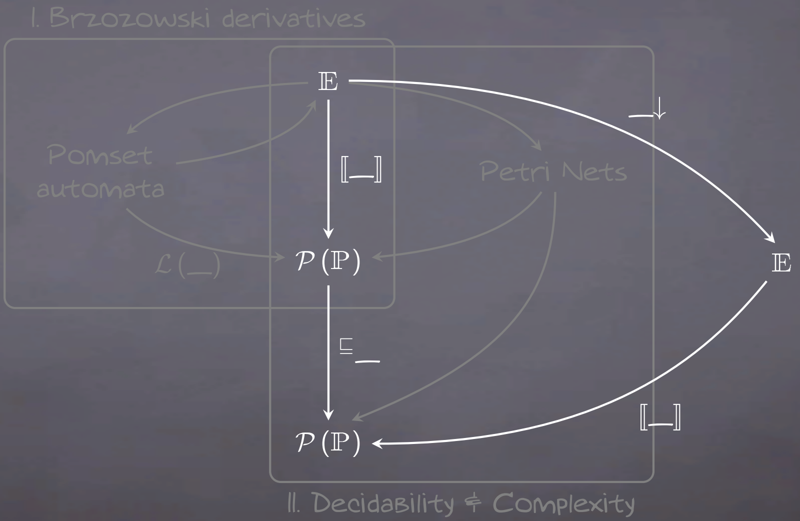
I. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
Petri Nets
$[\![\_]\!]$
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\sqsubseteq_{\_}$
$\mathcal{P}(\mathbb{P})$
II. Decidability & Complexity

Outline



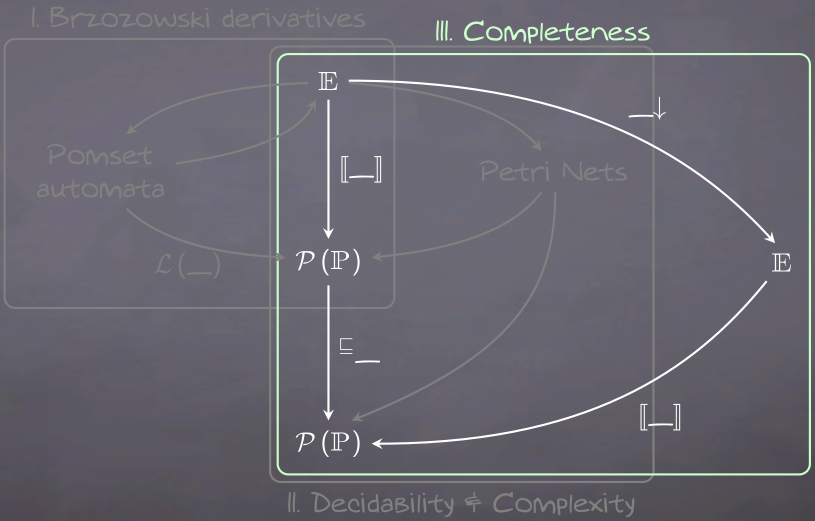
I. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
$[\![\_]\!]$
Petri Nets
$\_\downarrow$
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\mathbb{E}$
$\sqsubseteq_{\_}$
$\mathcal{P}(\mathbb{P})$
II. Decidability & Complexity

Outline



I. Brzozowski derivatives
$\mathbb{E}$
Pomset automata
Petri Nets
$[\![\_]\!]$
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\_\downarrow$
$\mathbb{E}$
$\sqsubseteq\_$
$[\![\_]\!]$
$\mathcal{P}(\mathbb{P})$
II. Decidability & Complexity

Outline

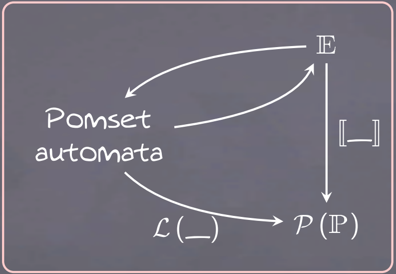


I. Brzozowski derivatives
III. Completeness
Pomset automata
Petri Nets
$\mathbb{E}$
$[\![\_]\!]$
$\_\downarrow$
$\mathcal{P}(\mathbb{P})$
$\mathcal{L}(\_)$
$\mathbb{E}$
$\sqsubseteq\_$
$[\![\_]\!]$
$\mathcal{P}(\mathbb{P})$
II. Decidability & Complexity

Outline

Kappé, Brunet, Luttik, Silva & Zanasi, "Brzozowski Goes Concurrent", 2017

1. Brzozowski derivatives



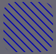
Rational pomset languages

$$e, f \in \mathbb{E} ::= a \mid 0 \mid 1 \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star.$$

Rational pomset languages

$$e, f \in \mathbb{E} ::= a \mid 0 \mid 1 \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star.$$

$[\![a]\!] := \Big\{$  $\Big\}$

$[\![1]\!] := \Big\{$  $\Big\}$

$$[\![0]\!] := \emptyset$$

$$[\![e + f]\!] := [\![e]\!] \cup [\![f]\!]$$

$$[\![e \cdot f]\!] := [\![e]\!] \cdot [\![f]\!]$$

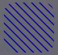
$$[\![e \parallel f]\!] := [\![e]\!] \parallel [\![f]\!]$$

$$[\![e^\star]\!] := \bigcup_{n \in \mathbb{N}} [\![e]\!]^n$$

Rational pomset languages

$$e, f \in \mathbb{E} ::= a \mid 0 \mid 1 \mid e \cdot f \mid e \parallel f \mid e + f \mid e^{\star}.$$

$[\![a]\!] := \{$  $\}$

$[\![1]\!] := \{$  $\}$

$$[\![0]\!] := \emptyset$$

$$[\![e + f]\!] := [\![e]\!] \cup [\![f]\!]$$

$$[\![e \cdot f]\!] := [\![e]\!] \cdot [\![f]\!]$$

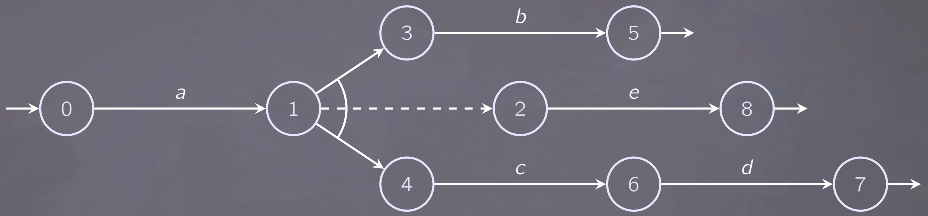
$$[\![e \parallel f]\!] := [\![e]\!] \parallel [\![f]\!]$$

$$[\![e^{\star}]\!] := \bigcup_{n \in \mathbb{N}} [\![e]\!]^{n}$$

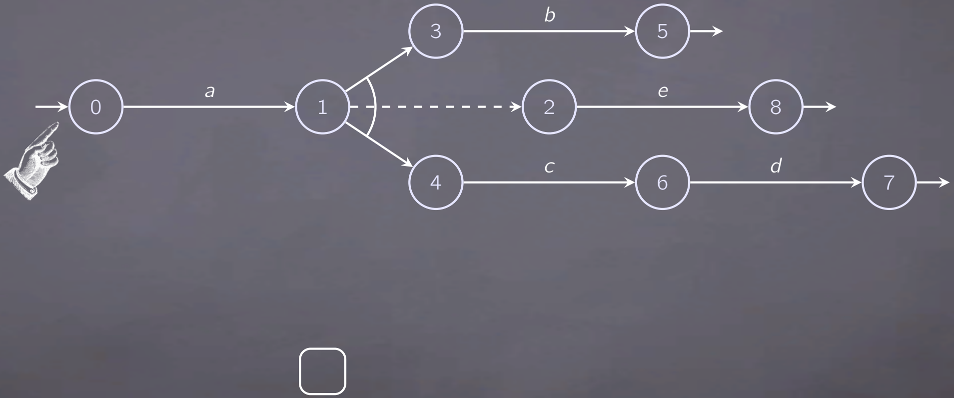
Definition

A set of pomsets S is called a rational pomset language if there is an expression $e \in \mathbb{E}$ such that $S = [\![e]\!]$.

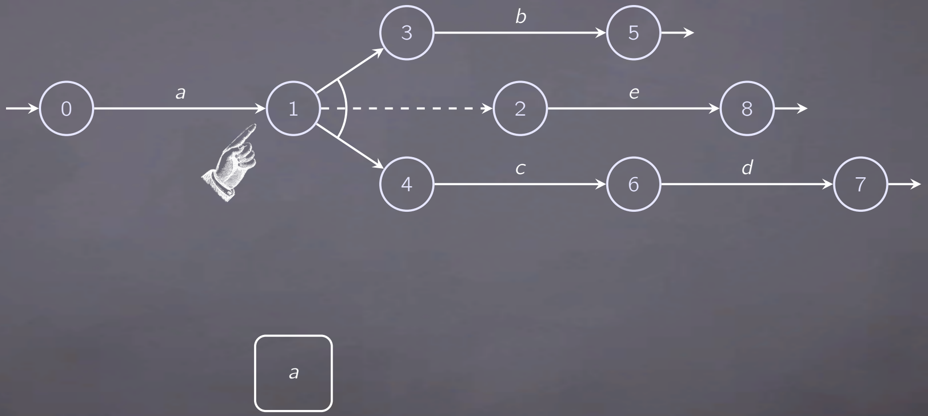
Pomset automata



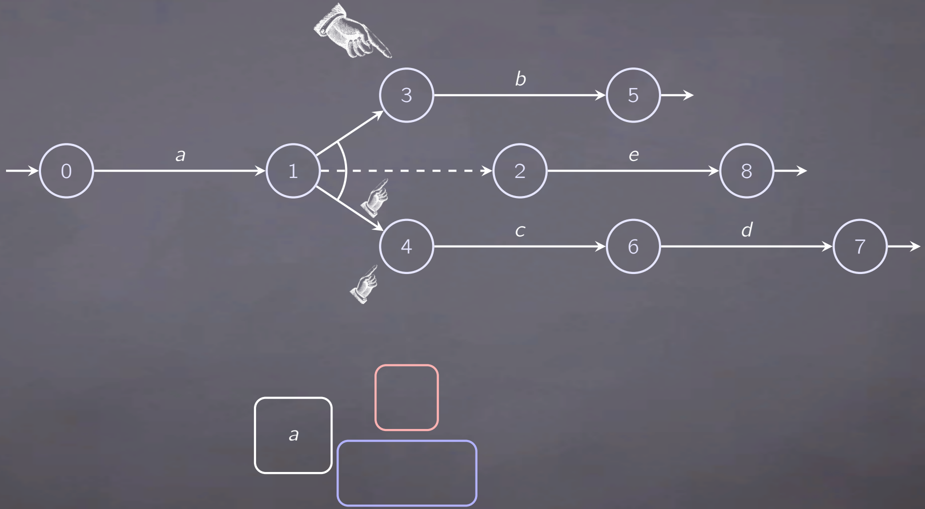
Pomset automata



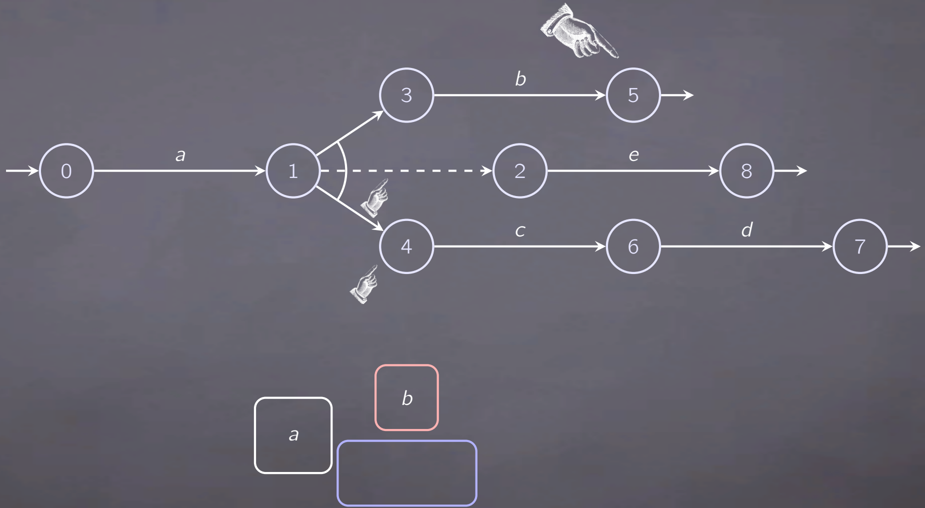
Pomset automata



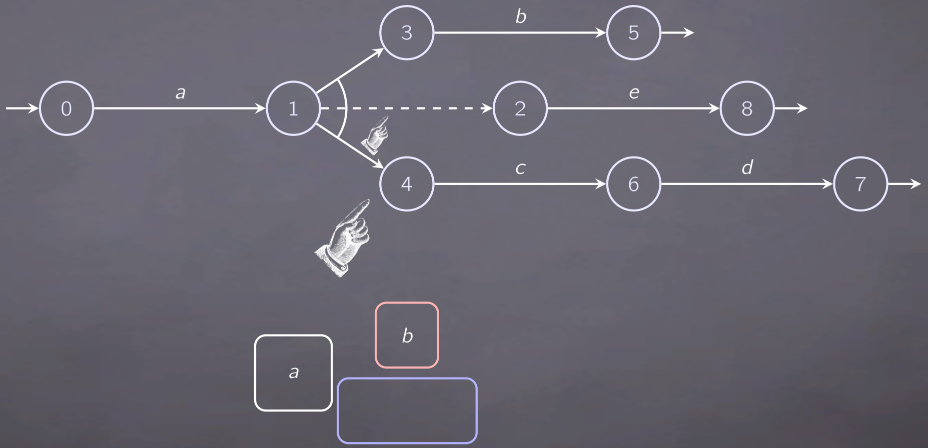
Pomset automata



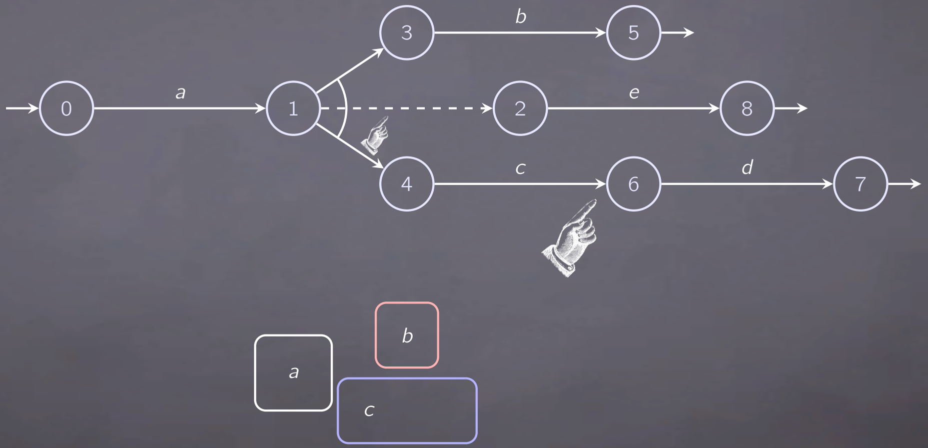
Pomset automata



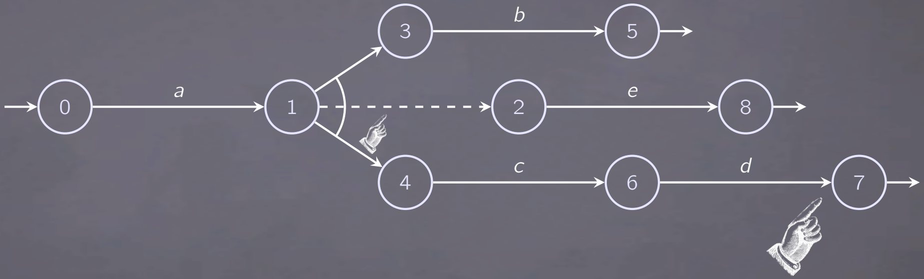
Pomset automata

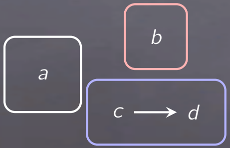


Pomset automata

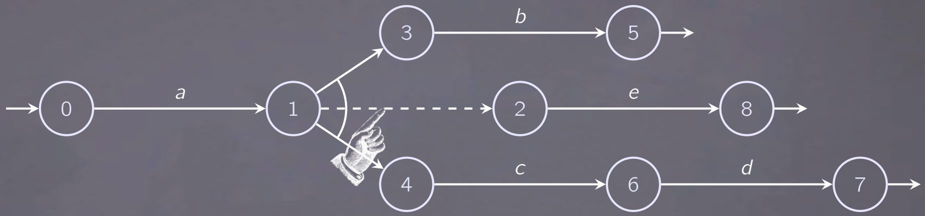


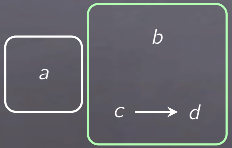
Pomset automata



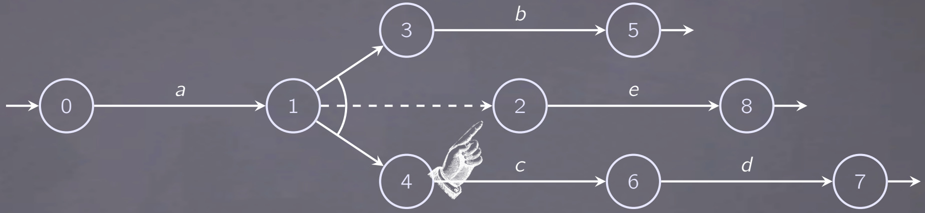


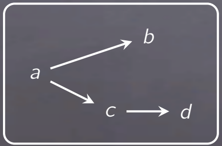
Pomset automata



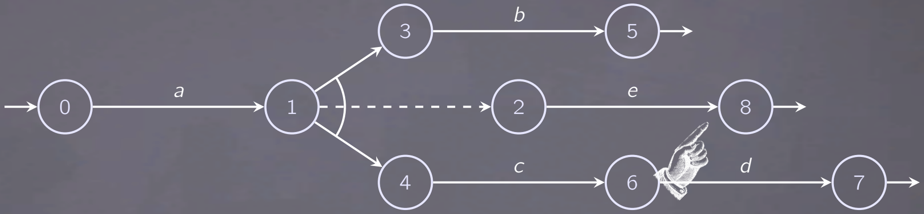


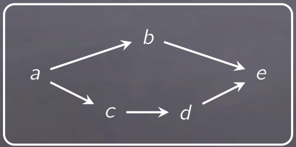
Pomset automata



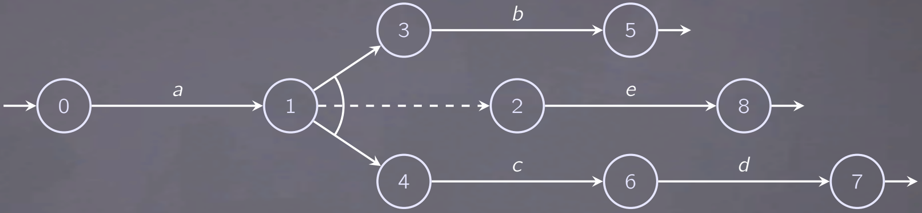


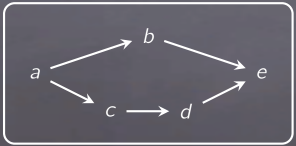
Pomset automata



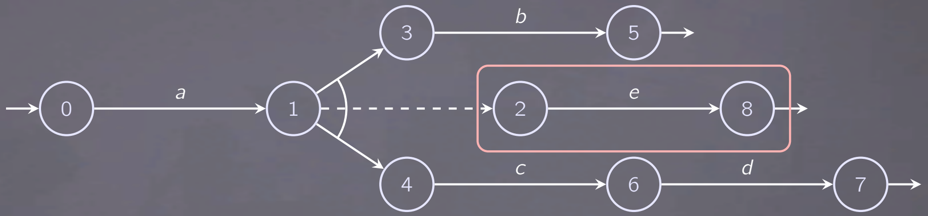


Pomset automata





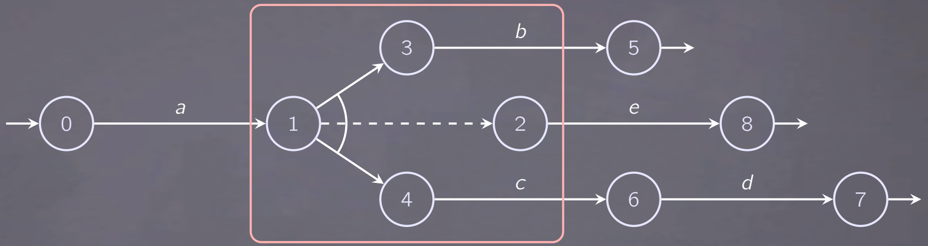
Pomset automata



Sequential transition function

$$\delta(2, e) = 8$$

Pomset automata



Sequential transition function

$$\delta(2, e) = 8$$

Parallel transition function

$$\gamma(1, \{\!|3, 4|\!\}) = 2$$

From expressions to automata

Brzozowski goes concurrent ▸ Definitions

1. Build an infinite pomset automaton:

From expressions to automata

Brzozowski goes concurrent (▸ Definitions)

1. Build an infinite pomset automaton:
   - states are expressions;

From expressions to automata

Brzozowski goes concurrent ▸ Definitions

1. Build an infinite pomset automaton:
   - states are expressions;
   - the initial state is the expression we want to translate;

From expressions to automata
Brzozowski goes concurrent ▸ Definitions

1. Build an infinite pomset automaton:
   - states are expressions;
   - the initial state is the expression we want to translate;
   - an expression is accepting if it contains 1;

From expressions to automata

Brzozowski goes concurrent ▸ Definitions

1. Build an infinite pomset automaton:
   - states are expressions;
   - the initial state is the expression we want to translate;
   - an expression is accepting if it contains 1;
   - the sequential transitions are defined by sequential derivatives;

From expressions to automata

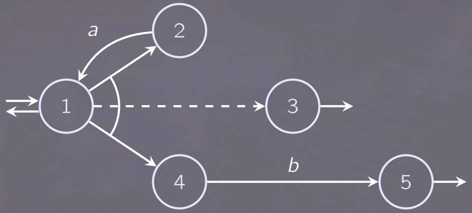
Brzozowski goes concurrent ▸ Definitions

1. Build an infinite pomset automaton:
   - states are expressions;
   - the initial state is the expression we want to translate;
   - an expression is accepting if it contains 1;
   - the sequential transitions are defined by sequential derivatives;
   - the parallel transitions are defined by parallel derivatives.

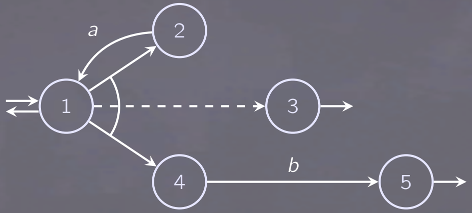
From expressions to automata
Brzozowski goes concurrent ▸ Definitions

1. Build an infinite pomset automaton:
   - states are expressions;
   - the initial state is the expression we want to translate;
   - an expression is accepting if it contains 1;
   - the sequential transitions are defined by sequential derivatives;
   - the parallel transitions are defined by parallel derivatives.
2. Quotient the automaton into a finite one, using the properties of $\langle 0, + \rangle$.

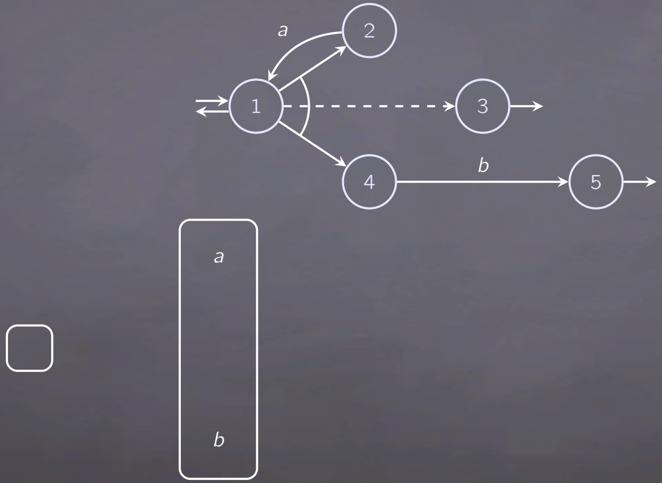
A very naughty automaton



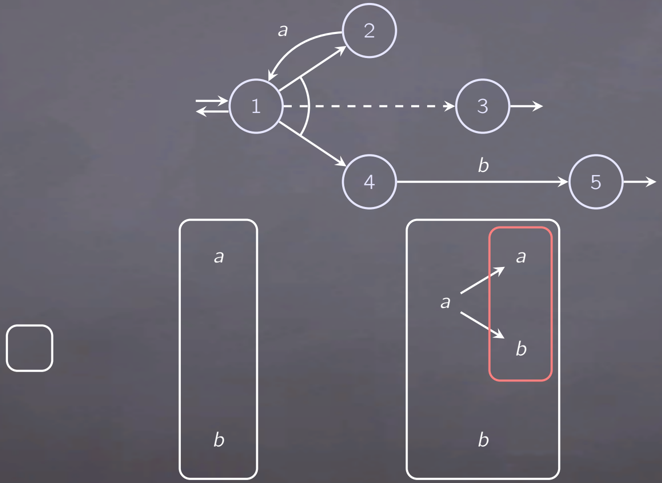
A very naughty automaton



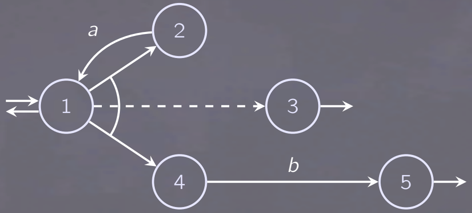
A very naughty automaton

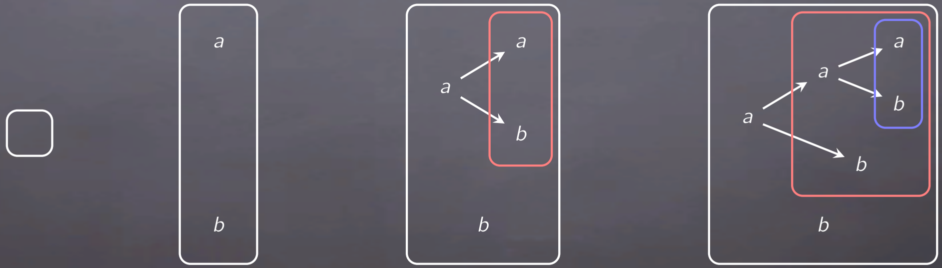


A very naughty automaton

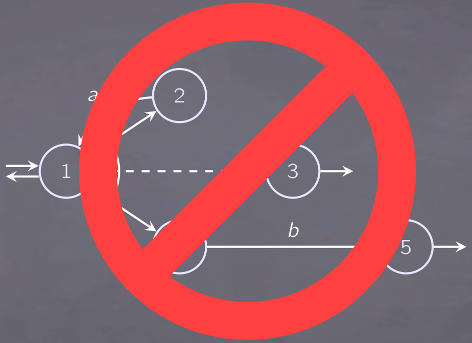


A very naughty automaton

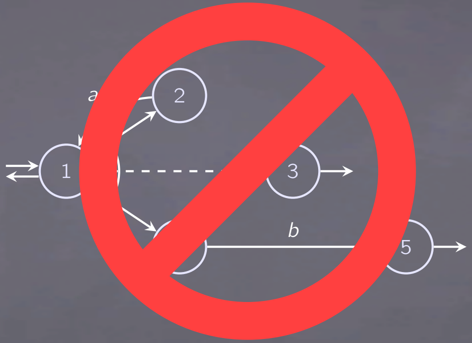




A very naughty automaton



A very naughty automaton



Fork-Acyclicity: there is a partial order over states $\preceq$ such that:

$$\frac{q \preceq r}{\gamma(q, \{\!|r, s|\!\}) = \bot} \qquad \frac{}{\delta(q, a) \preceq q} \qquad \frac{}{\gamma(q, \varphi) \preceq q.}$$

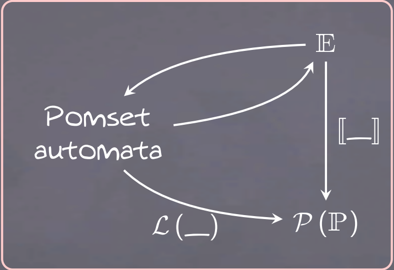
Kleene Theorem

Theorem

A pomset language is series-rational if and only if it is recognisable with fork-acyclic pomset automata.

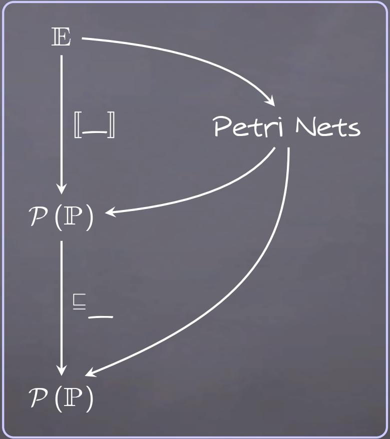
Outline

1. Brzozowski derivatives



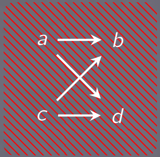
Outline

Brunet, Pous & Struth, "On decidability of concurrent Kleene algebra", 2017

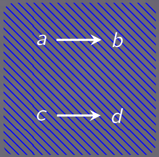


II. Decidability & Complexity

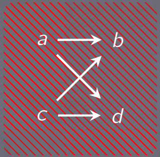
Pomset order



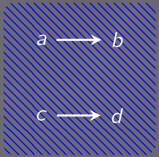
$\sqsubseteq$



Pomset order



$\sqsubseteq$



Definition

$P_1 \sqsubseteq P_2$ if there is a function $\varphi : P_2 \to P_1$ such that:

1. φ is a bijection
2. φ preserves labels
3. φ preserves ordered pairs

Gischer, "The equational theory of pomsets", 1988

Grabowski, "On partial languages", 1981

Two decision problems

Notation

${}^{\sqsubseteq}S := \{P \mid \exists P' \in S : P \sqsubseteq P'\}.$

Two decision problems

Notation

${}^{\sqsubseteq}S := \{P \mid \exists P' \in S : P \sqsubseteq P'\}.$

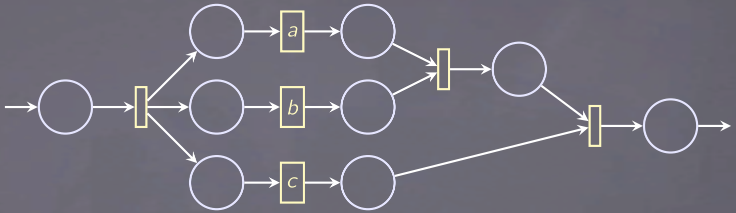
biKA

Given two expressions e, f, are $[\![e]\!]$ and $[\![f]\!]$ equal?

CKA

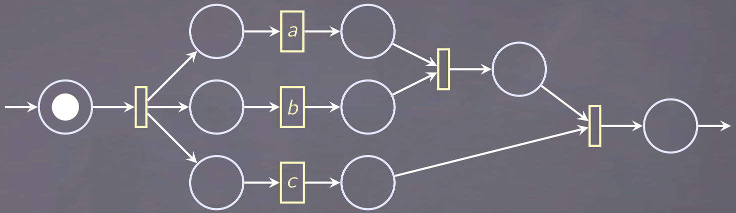
Given two expressions e, f, are ${}^{\sqsubseteq}[\![e]\!]$ and ${}^{\sqsubseteq}[\![f]\!]$ equal?

Labelled Petri nets



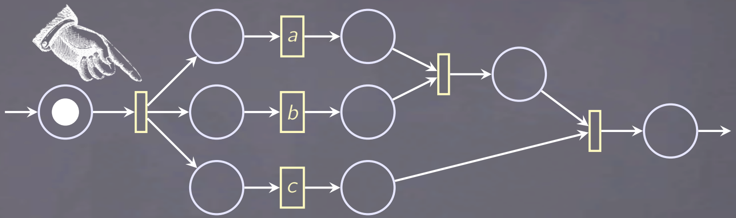
⏩ skip

Labelled Petri nets



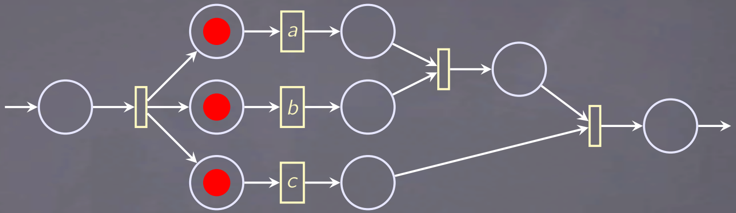
⏩ skip

Labelled Petri nets



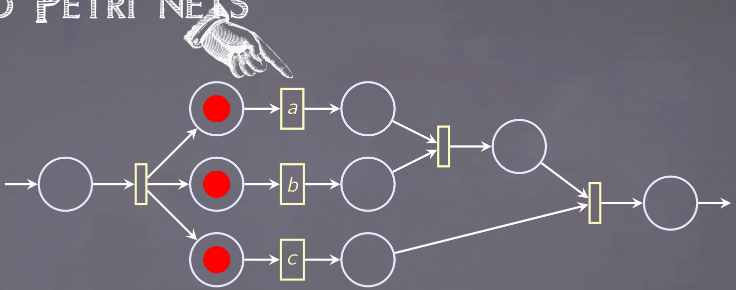
⏩ skip

Labelled Petri nets



⏩ skip

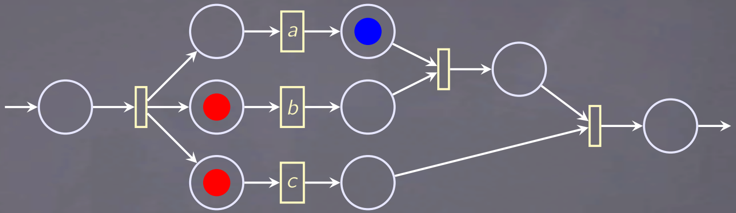
Labelled Petri nets



▸▸ skip

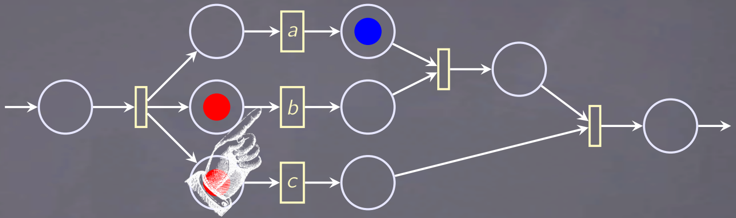
τ

Labelled Petri nets



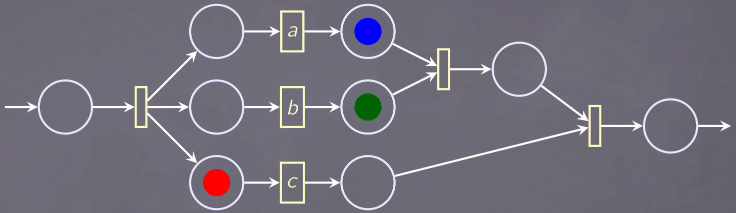
⏩ skip

Labelled Petri nets



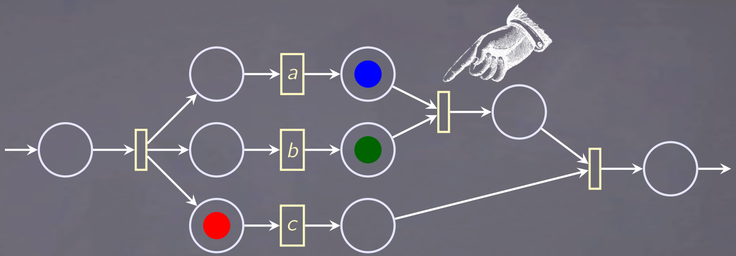
⏩ skip

Labelled Petri nets



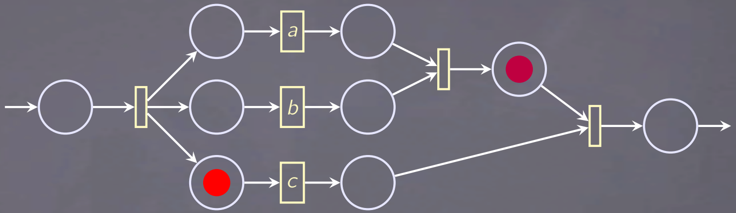
⏩ skip

Labelled Petri nets

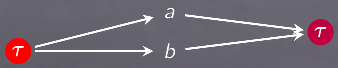


⏩ skip

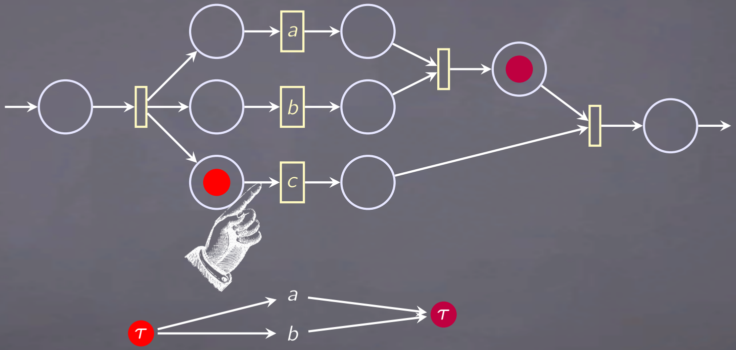
Labelled Petri nets



skip

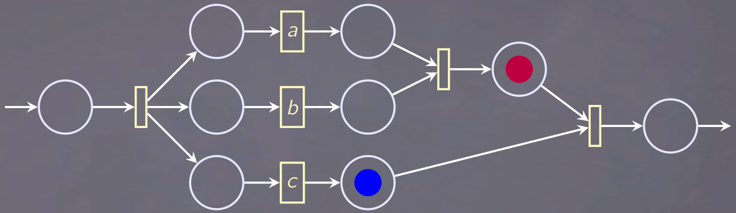


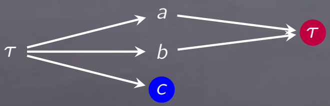
Labelled Petri nets



⏩ skip

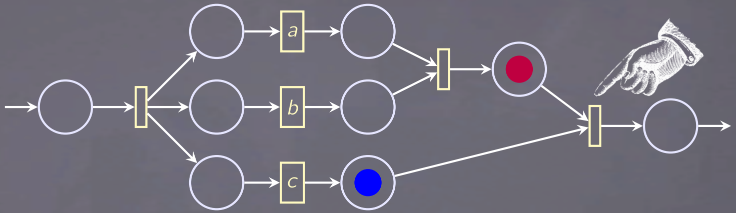
Labelled Petri nets

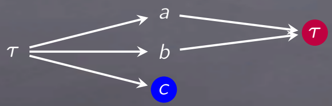




⏩ skip

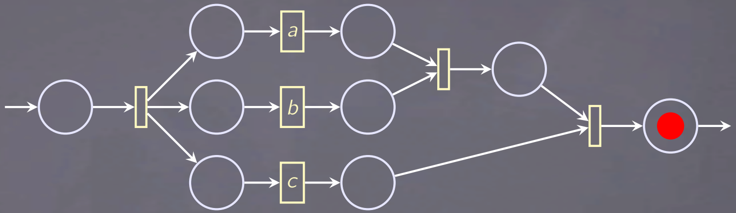
Labelled Petri nets





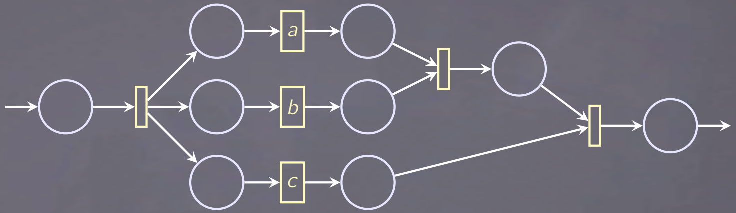
⏩ skip

Labelled Petri nets



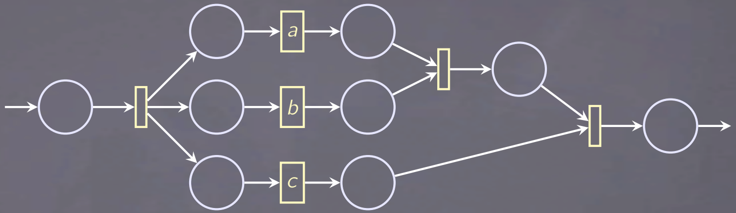
skip

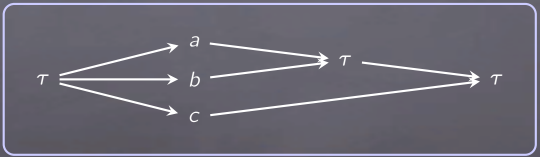
Labelled Petri nets



▸▸ skip

Labelled Petri nets

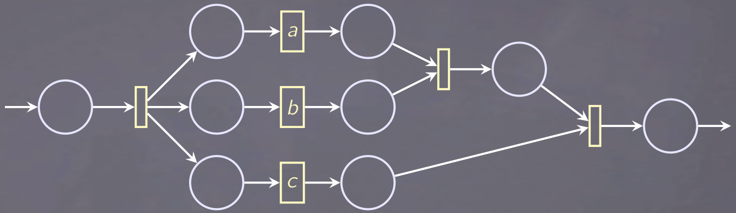


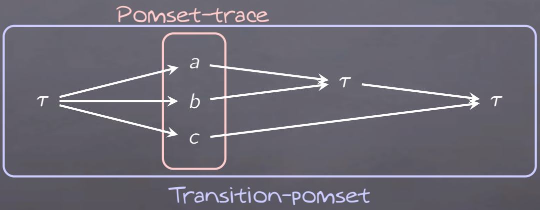


Transition-pomset

⏩ skip

Labelled Petri nets





▸▸ skip

Recognisable pomset languages

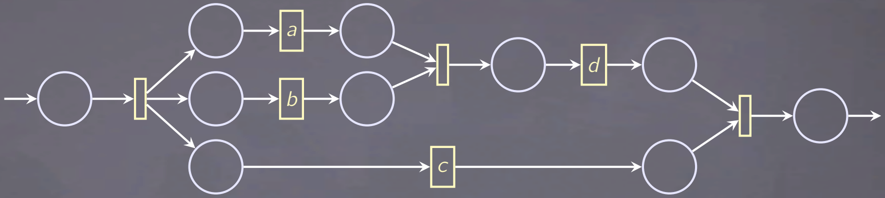
Language generated by a net

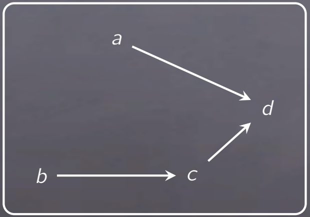
$[\![\mathcal{N}]\!]$ is the set of pomset-traces of accepting runs of $\mathcal{N}$.

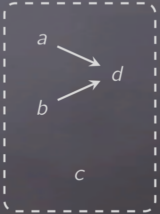
Definition

A set of pomsets S is a recognisable pomset language if there is a net $\mathcal{N}$ such that $S = [\![\mathcal{N}]\!]$.

Reading a pomset in a net

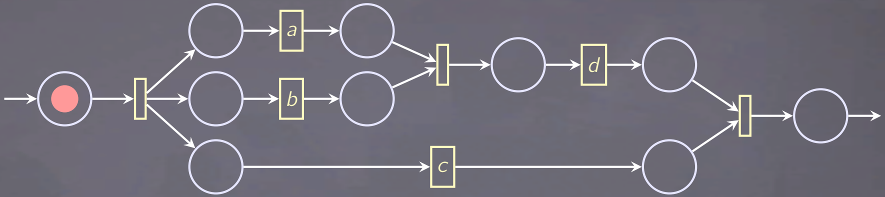


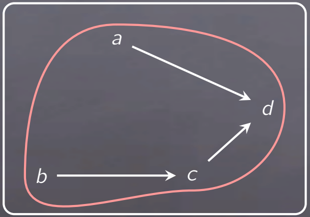


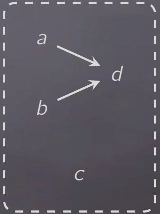


skip

Reading a pomset in a net

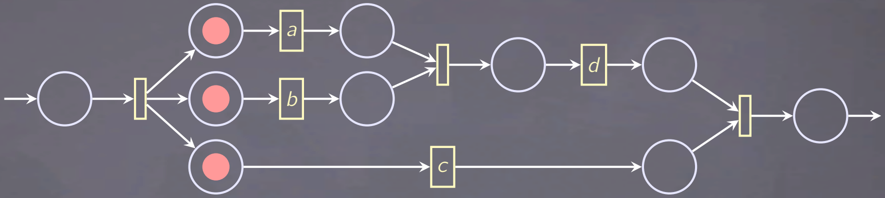


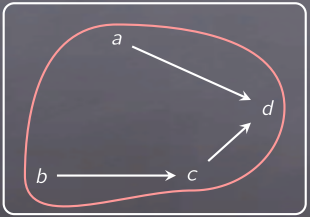


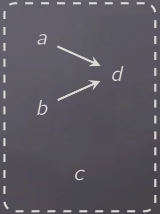


skip

Reading a pomset in a net

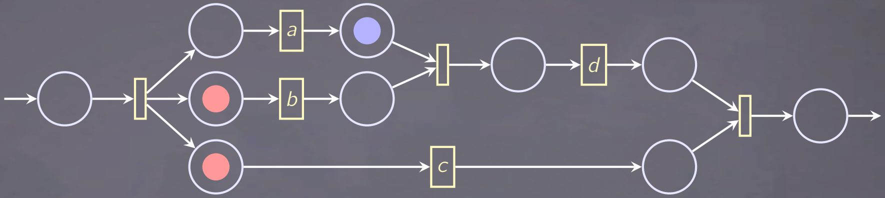


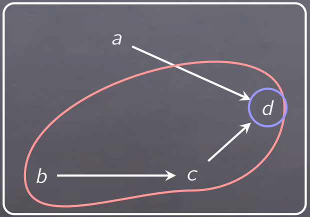


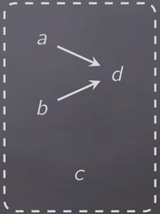


skip

Reading a pomset in a net

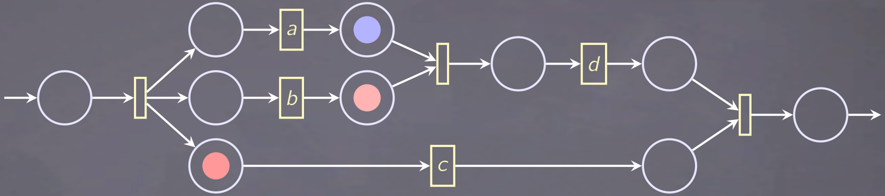


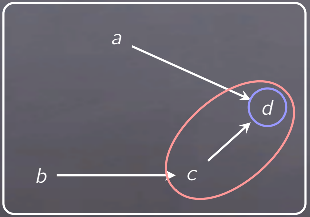


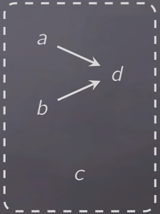


⏩ skip

Reading a pomset in a net

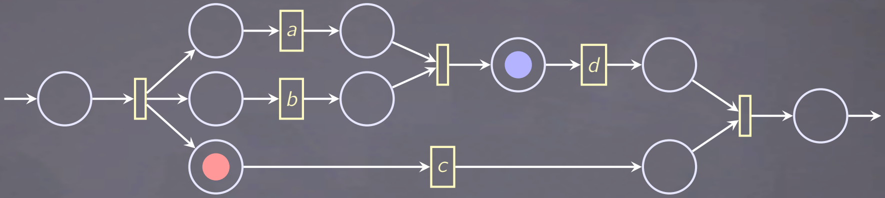


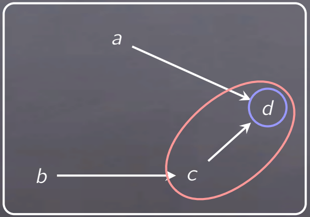


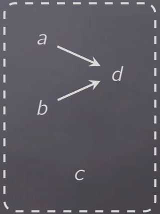


▸▸ skip

Reading a pomset in a net

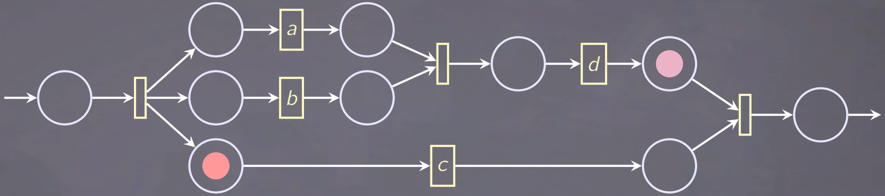


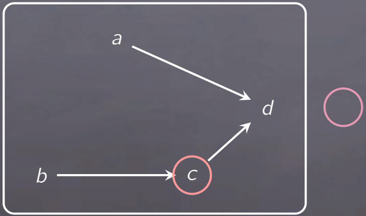


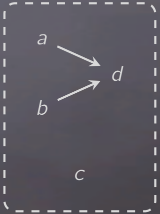


⏩ skip

Reading a pomset in a net

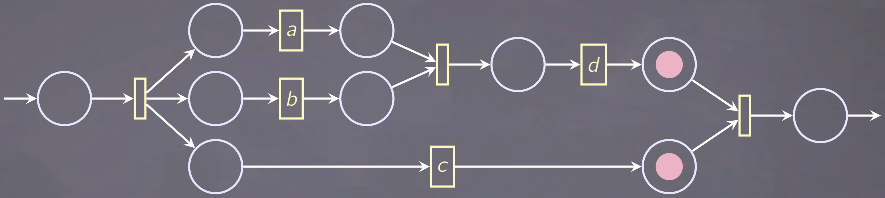


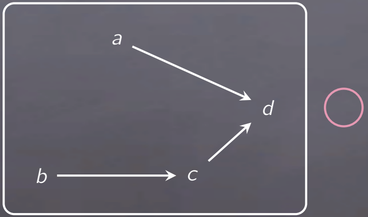


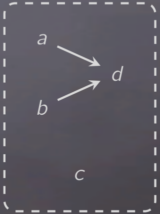


skip

Reading a pomset in a net

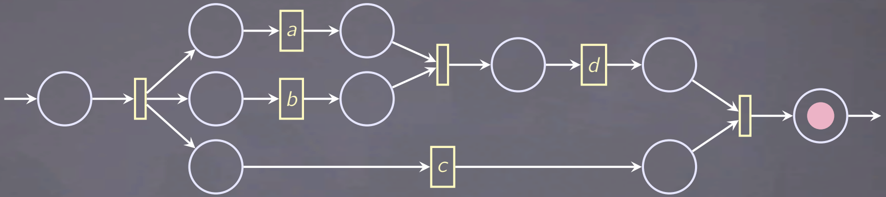


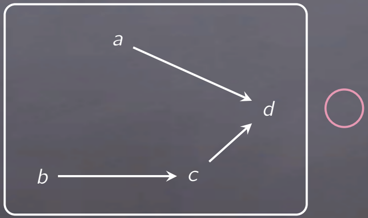


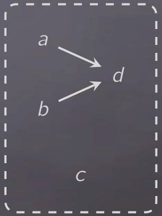


skip

Reading a pomset in a net

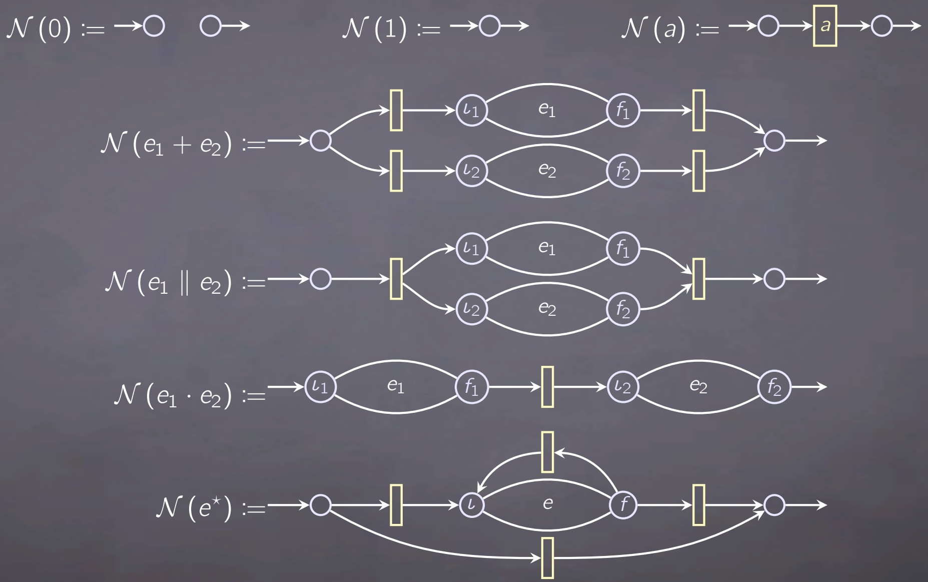






skip

From expressions to automata



Solving biKA

Lemma

$$[\![e]\!] = [\![\mathcal{N}(e)]\!].$$

Corollary

Rational pomset languages are recognisable.

Solving biKA

Lemma

$$[\![e]\!] = [\![\mathcal{N}(e)]\!].$$

Corollary

Rational pomset languages are recognisable.

Theorem

Testing containment of pomset-trace languages of two Petri nets is an ExpSpace-complete problem.

Jategaonkar & Meyer, "Deciding true concurrency equivalences on safe, finite nets", 1996

Corollary

The problem biKA lies in the class ExpSpace.

What about CKA?

$$^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!]$$

What about CKA?

$$ {}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!] \Leftrightarrow \quad {}^{\sqsubseteq}[\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \quad \wedge \quad {}^{\sqsubseteq}[\![e]\!] \supseteq {}^{\sqsubseteq}[\![f]\!] $$

What about CKA?

$$\begin{aligned} {}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!] &\Leftrightarrow \quad {}^{\sqsubseteq}[\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \quad \wedge \quad {}^{\sqsubseteq}[\![e]\!] \supseteq {}^{\sqsubseteq}[\![f]\!] \\ &\Leftrightarrow \quad [\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \quad \wedge \quad {}^{\sqsubseteq}[\![e]\!] \supseteq [\![f]\!] \end{aligned}$$

What about CKA?

$$
\begin{aligned}
{}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!] &\Leftrightarrow \quad {}^{\sqsubseteq}[\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \quad \wedge \quad {}^{\sqsubseteq}[\![e]\!] \supseteq {}^{\sqsubseteq}[\![f]\!] \\
&\Leftrightarrow \quad [\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \quad \wedge \quad {}^{\sqsubseteq}[\![e]\!] \supseteq [\![f]\!] \\
&\Leftrightarrow [\![\mathcal{N}(e)]\!] \subseteq {}^{\sqsubseteq}[\![\mathcal{N}(f)]\!] \wedge {}^{\sqsubseteq}[\![\mathcal{N}(e)]\!] \supseteq [\![\mathcal{N}(f)]\!]
\end{aligned}
$$

What about CKA?

$$
\begin{aligned}
{}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!] &\Leftrightarrow {}^{\sqsubseteq}[\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \wedge {}^{\sqsubseteq}[\![e]\!] \supseteq {}^{\sqsubseteq}[\![f]\!] \\
&\Leftrightarrow [\![e]\!] \subseteq {}^{\sqsubseteq}[\![f]\!] \wedge {}^{\sqsubseteq}[\![e]\!] \supseteq [\![f]\!] \\
&\Leftrightarrow [\![\mathcal{N}(e)]\!] \subseteq {}^{\sqsubseteq}[\![\mathcal{N}(f)]\!] \wedge {}^{\sqsubseteq}[\![\mathcal{N}(e)]\!] \supseteq [\![\mathcal{N}(f)]\!]
\end{aligned}
$$

Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run R_1 of $\mathcal{N}_1$ there is a run R_2 in $\mathcal{N}_2$ such that

$$\mathcal{P}om(R_1) \sqsubseteq \mathcal{P}om(R_2)?$$

Idea of the algorithm

Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run R_1 of $\mathcal{N}_1$ there is a run R_2 in $\mathcal{N}_2$ such that

$$\mathcal{P}om(R_1) \sqsubseteq \mathcal{P}om(R_2)?$$

Idea of the algorithm

Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run R_1 of $\mathcal{N}_1$ there is a run R_2 in $\mathcal{N}_2$ such that

$$\mathcal{P}om(R_1) \sqsubseteq \mathcal{P}om(R_2)?$$

- Build an automaton $\mathscr{A}_1$ for $[\![\mathcal{N}_1]\!]$

Idea of the algorithm

Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run R_1 of $\mathcal{N}_1$ there is a run R_2 in $\mathcal{N}_2$ such that

$$\mathcal{P}om(R_1) \sqsubseteq \mathcal{P}om(R_2)?$$

- Build an automaton $\mathscr{A}_1$ for $[\![\mathcal{N}_1]\!]$
- Build an automaton $\mathscr{A}_2$ for $[\![\mathcal{N}_1]\!] \cap^{\sqsubseteq} [\![\mathcal{N}_2]\!]$

Idea of the algorithm

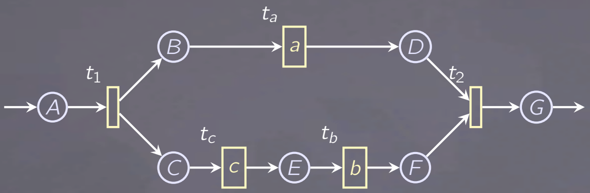
Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run R_1 of $\mathcal{N}_1$ there is a run R_2 in $\mathcal{N}_2$ such that

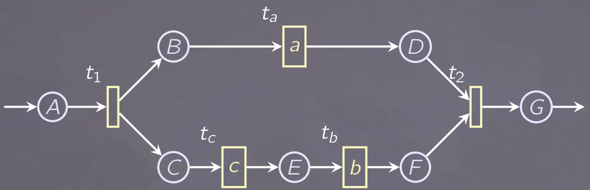
$$\mathcal{P}om(R_1) \sqsubseteq \mathcal{P}om(R_2)?$$

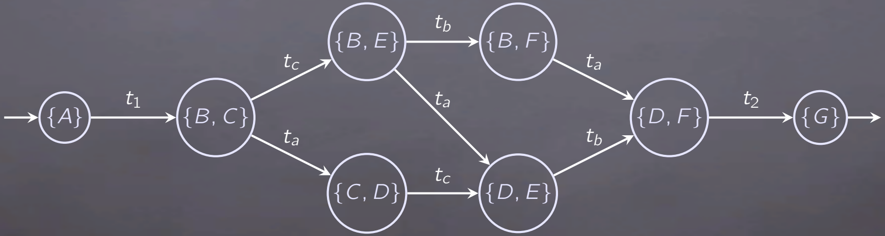
- Build an automaton $\mathscr{A}_1$ for $[\![\mathcal{N}_1]\!]$
- Build an automaton $\mathscr{A}_2$ for $[\![\mathcal{N}_1]\!] \cap^{\sqsubseteq} [\![\mathcal{N}_2]\!]$
- $[\![\mathcal{N}_1]\!] \subseteq^{\sqsubseteq} [\![\mathcal{N}_2]\!]$ if and only if $\mathcal{L}(\mathscr{A}_1) = \mathcal{L}(\mathscr{A}_2)$.

Transition automaton

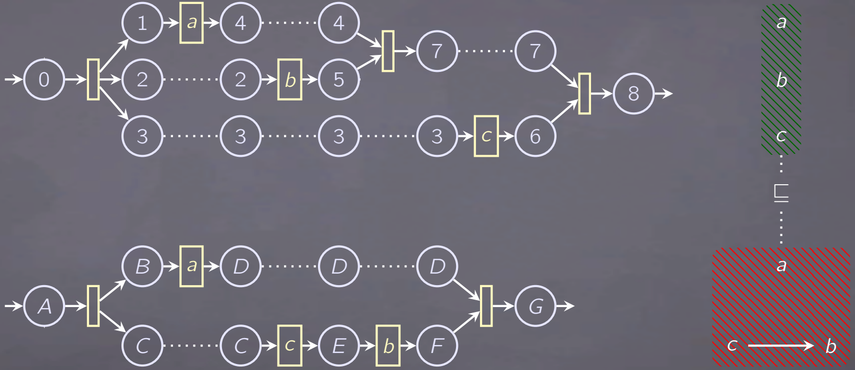


Transition automaton



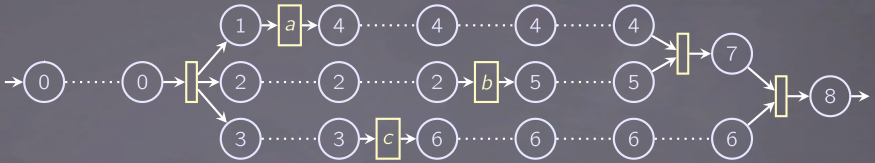


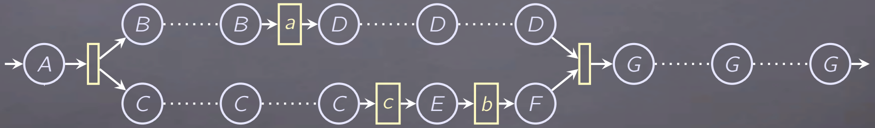
Massaging runs



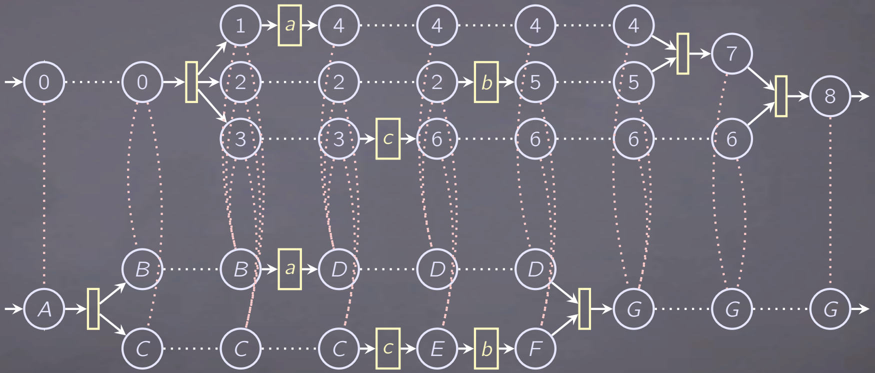
Massaging runs

Massaging runs

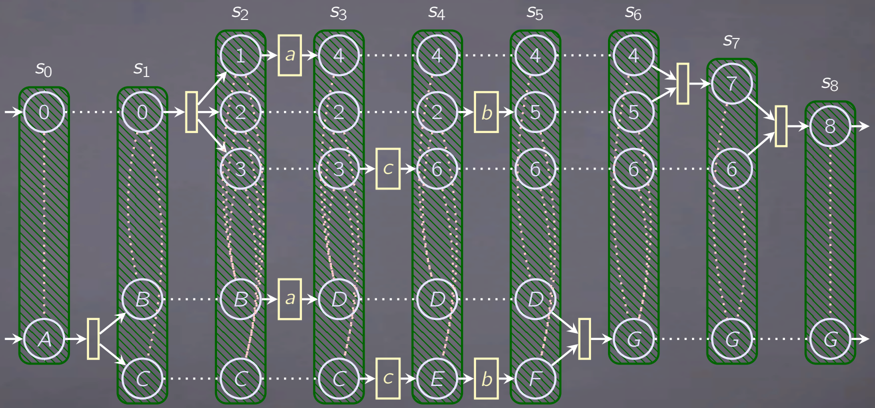




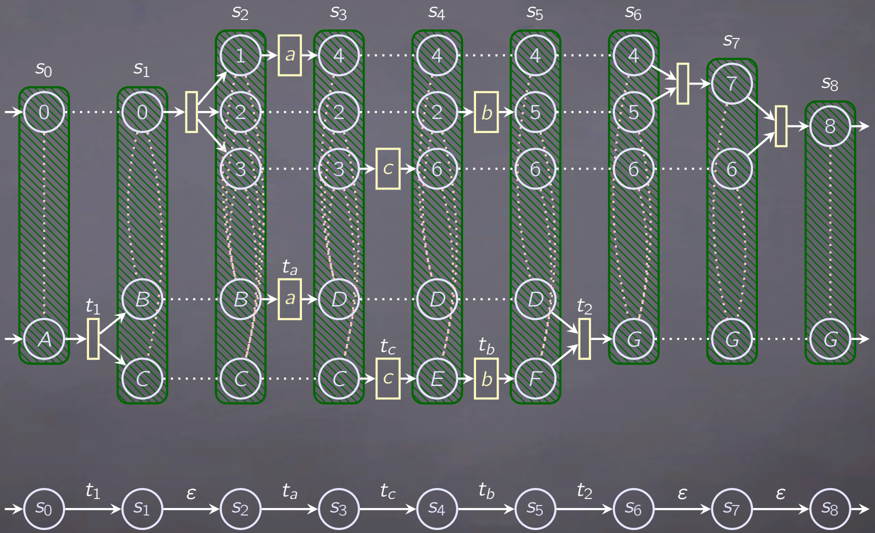
Massaging runs



Massaging runs



Massaging runs



Reduction to automata

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be some polite nets, of size n, m.

Lemma

There is an automaton $\mathscr{A}(\mathcal{N}_1)$ with $\mathcal{O}(2^n)$ states that recognises the set of accepting runs in $\mathcal{N}_1$.

Reduction to automata

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be some polite nets, of size n, m.

Lemma

There is an automaton $\mathscr{A}(\mathcal{N}_1)$ with $\mathcal{O}(2^n)$ states that recognises the set of accepting runs in $\mathcal{N}_1$.

Lemma

There is an automaton $\mathcal{N}_1 \prec \mathcal{N}_2$ with $\mathcal{O}(2^{n+m+nm})$ states that recognises the set of accepting runs in $\mathcal{N}_1$ whose pomset belongs to ${}^{\sqsubseteq}[\![\mathcal{N}_2]\!]$.

Decidability + Complexity

Theorem

Given two expressions $e, f \in \mathbb{E}$, we can test if $[\![e]\!] \subseteq^{\sqsubseteq} [\![f]\!]$ in ExpSpace.

Proof.

1. Build $\mathcal{N}(e)$ and $\mathcal{N}(f)$;
2. Build $\mathscr{A}(\mathcal{N}(e))$ and $\mathcal{N}(e) \prec \mathcal{N}(f)$;
3. compare them.

$\square$

Decidability + Complexity

Theorem

Given two expressions $e, f \in \mathbb{E}$, we can test if $[\![e]\!] \subseteq^{\sqsubseteq} [\![f]\!]$ in ExpSpace.

Proof.

1. Build $\mathcal{N}(e)$ and $\mathcal{N}(f)$;
2. Build $\mathscr{A}(\mathcal{N}(e))$ and $\mathcal{N}(e) \prec \mathcal{N}(f)$;
3. compare them.

□

Theorem

The problem CKA is ExpSpace-complete.

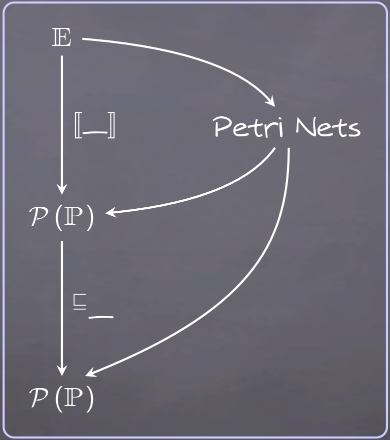
Proof.

1. In the class ExpSpace: see above.
2. ExpSpace-hard: Reduction from the universality problem for regular expressions with interleaving.

Mayer & Stockmeyer, "The complexity of word problems – this time with interleaving", 1994

□

Outline

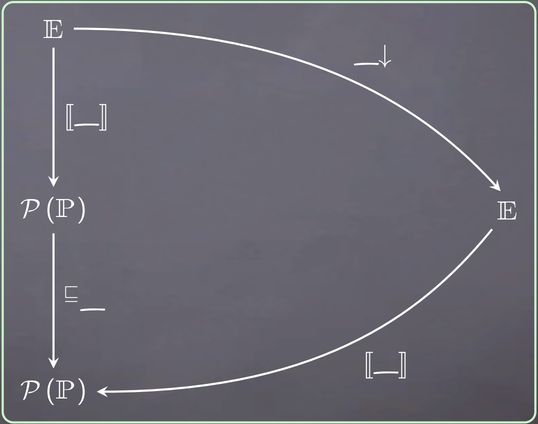


II. Decidability & Complexity

Outline

Kappé, Brunet, Silva & Zanasi, "Concurrent Kleene Algebra: Free Model and Completeness", 2017 (submitted)

III. Completeness



Kleene Algebra

Equivalence of sequential programs

A Kleene algebra is structure $\langle K, 0, 1, +, \cdot, \star\rangle$ such that:

1. $\langle K, 0, 1, +, \cdot\rangle$ is an idempotent semiring;
2. $\forall x \in K, 1 + x \cdot x^\star = x^\star$;
3. $\forall x, y, z \in K, x + y \cdot z \leq z \Rightarrow y^\star \cdot x \leq z$.

Theorem

$$\mathrm{KA} \vdash e = f \Leftrightarrow \mathcal{L}(e) = \mathcal{L}(f)\,.$$

Bi-Kleene algebra

A bi-Kleene algebra is structure $\langle K, 0, 1, +, \cdot, \star, \parallel \rangle$ such that:

1. $\langle K, 0, 1, +, \cdot \rangle$ is an idempotent semiring;
2. $\langle K, 0, 1, +, \parallel \rangle$ is a commutative idempotent semiring;
3. $\forall x \in K, 1 + x \cdot x^\star = x^\star$;
4. $\forall x, y, z \in K, x + y \cdot z \leq z \Rightarrow y^\star \cdot x \leq z$.

Theorem

$$\text{biKA} \vdash e = f \Leftrightarrow [\![e]\!] = [\![f]\!].$$

Laurence & Struth, "Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages", 2014

Concurrent Kleene algebra

A concurrent Kleene algebra is bi-Kleene algebra $\langle K, 0, 1, +, \cdot, \star, \parallel\rangle$ such that:

$$(a \parallel b) \cdot (c \parallel d) \leq (a \cdot c) \parallel (b \cdot d)$$

Concurrent Kleene algebra

A concurrent Kleene algebra is bi-Kleene algebra $\langle K, 0, 1, +, \cdot, \star, \parallel \rangle$ such that:

$$(a \parallel b) \cdot (c \parallel d) \leq (a \cdot c) \parallel (b \cdot d)$$

Concurrent Kleene algebra

A concurrent Kleene algebra is bi-Kleene algebra $\langle K, 0, 1, +, \cdot, \star, \parallel\rangle$ such that:

$$(a \parallel b) \cdot (c \parallel d) \leq (a \cdot c) \parallel (b \cdot d)$$

Theorem

$$\mathrm{CKA} \vdash e = f \Rightarrow {}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!].$$

Hoare, Möller, Struth & Wehrman, "Concurrent Kleene Algebra", 2009

Syntactic closures are nice...

Definition

An expression $e{\downarrow}$ is a closure of e if $\mathrm{CKA} \vdash e{\downarrow} = e$ and $[\![e{\downarrow}]\!] = {}^{\sqsubseteq}[\![e]\!]$.

Syntactic closures are nice...

Definition

An expression $e{\downarrow}$ is a <u>closure</u> of e if $\overbrace{\mathrm{CKA} \vdash e{\downarrow} = e}^{(1)}$ and $\overbrace{[\![e{\downarrow}]\!] = {}^{\sqsubseteq}[\![e]\!]}^{(2)}$.

Lemma

If every series-rational expression admits a closure, the axioms of CKA are complete with respect to down-closed pomset languages.

Laurence & Struth, "Completeness theorems for pomset languages and concurrent Kleene Algebras", (draft)

Syntactic closures are nice...

Definition

An expression $e{\downarrow}$ is a closure of e if $\overbrace{\mathrm{CKA} \vdash e{\downarrow} = e}^{(1)}$ and $\overbrace{[\![e{\downarrow}]\!] = {}^{\sqsubseteq}[\![e]\!]}^{(2)}$.

Lemma

If every series-rational expression admits a closure, the axioms of CKA are complete with respect to down-closed pomset languages.

Laurence & Struth, "Completeness theorems for pomset languages and concurrent Kleene Algebras", (draft)

Proof. Assume ${}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!]$.

Syntactic closures are nice...

Definition

An expression $e{\downarrow}$ is a closure of e if $\overbrace{\mathrm{CKA} \vdash e{\downarrow} = e}^{(1)}$ and $\overbrace{[\![e{\downarrow}]\!] = {}^{\sqsubseteq}[\![e]\!]}^{(2)}$.

Lemma

If every series-rational expression admits a closure, the axioms of CKA are complete with respect to down-closed pomset languages.

Laurence & Struth, "Completeness theorems for pomset languages and concurrent Kleene Algebras", (draft)

Proof. Assume ${}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!]$.

By (2), it means that $[\![e{\downarrow}]\!] = [\![f{\downarrow}]\!]$.

Syntactic closures are nice...

Definition

An expression $e{\downarrow}$ is a closure of e if $\overbrace{\mathrm{CKA} \vdash e{\downarrow} = e}^{(1)}$ and $\overbrace{[\![e{\downarrow}]\!] = {}^{\sqsubseteq}[\![e]\!]}^{(2)}$.

Lemma

If every series-rational expression admits a closure, the axioms of CKA are complete with respect to down-closed pomset languages.

Laurence & Struth, "Completeness theorems for pomset languages and concurrent Kleene Algebras", (draft)

Proof. Assume ${}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!]$.
By (2), it means that $[\![e{\downarrow}]\!] = [\![f{\downarrow}]\!]$.
By compleness of biKA, it follows that $\mathrm{biKA} \vdash e{\downarrow} = f{\downarrow}$, thus $\mathrm{CKA} \vdash e{\downarrow} = f{\downarrow}$.

Syntactic closures are nice...

Definition

An expression $e{\downarrow}$ is a <u>closure</u> of e if $\overbrace{\mathrm{CKA} \vdash e{\downarrow} = e}^{(1)}$ and $\overbrace{[\![e{\downarrow}]\!] = {}^{\sqsubseteq}[\![e]\!]}^{(2)}$.

Lemma

If every series-rational expression admits a closure, the axioms of CKA are complete with respect to down-closed pomset languages.

Laurence & Struth, "Completeness theorems for pomset languages and concurrent Kleene Algebras", (draft)

<u>Proof.</u> Assume ${}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!]$.
By (2), it means that $[\![e{\downarrow}]\!] = [\![f{\downarrow}]\!]$.
By compleness of biKA, it follows that $\mathrm{biKA} \vdash e{\downarrow} = f{\downarrow}$, thus $\mathrm{CKA} \vdash e{\downarrow} = f{\downarrow}$.
By (1) we get that $\mathrm{CKA} \vdash e = e{\downarrow} = f{\downarrow} = f$.

Syntactic closures are nice...

Definition

An expression $e\!\downarrow$ is a closure of e if $\overbrace{\mathrm{CKA} \vdash e\!\downarrow = e}^{(1)}$ and $\overbrace{[\![e\!\downarrow]\!] = {}^{\sqsubseteq}[\![e]\!]}^{(2)}$.

Lemma

If every series-rational expression admits a closure, the axioms of CKA are complete with respect to down-closed pomset languages.

Laurence & Struth, "Completeness theorems for pomset languages and concurrent Kleene Algebras", (draft)

Proof. Assume ${}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!]$.
By (2), it means that $[\![e\!\downarrow]\!] = [\![f\!\downarrow]\!]$.
By compleness of biKA, it follows that $\mathrm{biKA} \vdash e\!\downarrow = f\!\downarrow$, thus $\mathrm{CKA} \vdash e\!\downarrow = f\!\downarrow$.
By (1) we get that $\mathrm{CKA} \vdash e = e\!\downarrow = f\!\downarrow = f$. $\square$

… but do they exist?

Let's try and compute the closure by induction:

… but do they exist?

Let's try and compute the closure by induction:

- $0{\downarrow} = 0$
- $1{\downarrow} = 1$
- $a{\downarrow} = a$

… but do they exist?

Let's try and compute the closure by induction:

- $0{\downarrow} = 0$
- $1{\downarrow} = 1$
- $a{\downarrow} = a$
- $(e + f){\downarrow} = e{\downarrow} + f{\downarrow}$

… BUT DO THEY EXIST?

Let's try and compute the closure by induction:

- $0{\downarrow} = 0$
- $1{\downarrow} = 1$
- $a{\downarrow} = a$
- $(e + f){\downarrow} = e{\downarrow} + f{\downarrow}$
- $(e \cdot f){\downarrow} = e{\downarrow} \cdot f{\downarrow}$

... but do they exist?

Let's try and compute the closure by induction:

- $0\downarrow = 0$
- $1\downarrow = 1$
- $a\downarrow = a$
- $(e+f)\downarrow = e\downarrow + f\downarrow$
- $(e \cdot f)\downarrow = e\downarrow \cdot f\downarrow$
- $(e^{\star})\downarrow = e\downarrow^{\star}$

… but do they exist?

Let's try and compute the closure by induction:

- $0\downarrow = 0$
- $1\downarrow = 1$
- $a\downarrow = a$
- $(e + f)\downarrow = e\downarrow + f\downarrow$
- $(e \cdot f)\downarrow = e\downarrow \cdot f\downarrow$
- $(e^\star)\downarrow = e\downarrow^\star$
- $(e \parallel f)\downarrow = ???$

… but do they exist?

Let's try and compute the closure by induction:

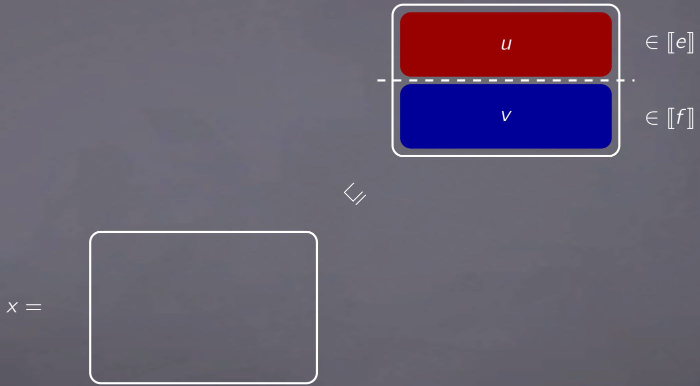
- $0{\downarrow} = 0$
- $1{\downarrow} = 1$
- $a{\downarrow} = a$
- $(e + f){\downarrow} = e{\downarrow} + f{\downarrow}$
- $(e \cdot f){\downarrow} = e{\downarrow} \cdot f{\downarrow}$
- $(e^\star){\downarrow} = e{\downarrow}^\star$
- $(e \parallel f){\downarrow} = ???$

We strengthen our induction, by assuming that we have closures for

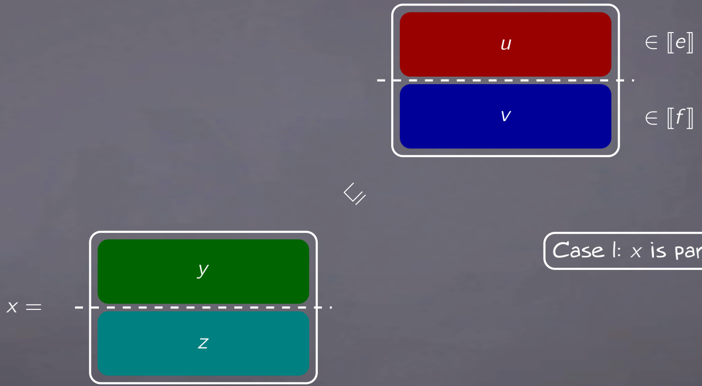
1. every strict subterm of $e \parallel f$,
2. every term with smaller width than $e \parallel f$.

We write the corresponding strict ordering $\prec$.

Who's smaller than a parallel product?

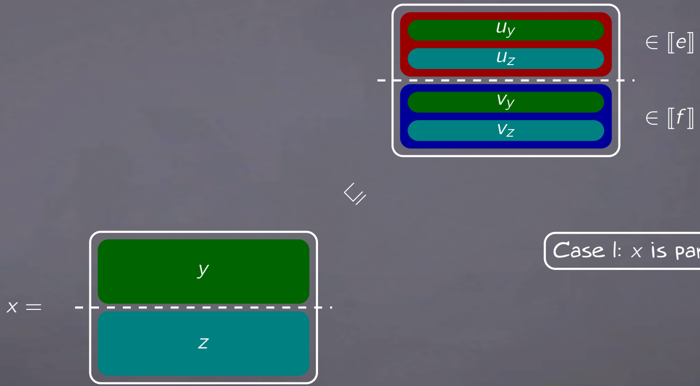


Who's smaller than a parallel product?



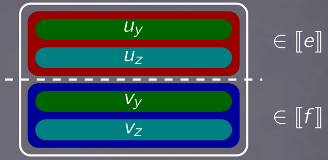
Case 1: x is parallel.

Who's smaller than a parallel product?

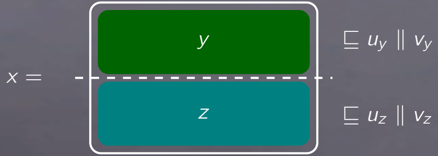


Case 1: x is parallel.

Who's smaller than a parallel product?



$\sqsubseteq_{/\!/}$



Case 1: x is parallel.

Parallel splicing and preclosure

Parallel splicing

Δ_e is a finite relation over $\mathbb{E}$ such that:

$$u \parallel v \in [\![e]\!] \Leftrightarrow \exists l \; \Delta_e \; r : u \in [\![l]\!] \wedge v \in [\![r]\!].$$

Parallel splicing and preclosure

Parallel splicing

Δ_e is a finite relation over $\mathbb{E}$ such that:

$$u \parallel v \in [\![e]\!] \Leftrightarrow \exists l \; \Delta_e \; r : u \in [\![l]\!] \wedge v \in [\![r]\!].$$

$$e \odot f = e \parallel f + \sum_{l \Delta_{e \parallel f} r} (l{\downarrow}) \parallel (r{\downarrow}).$$

Parallel splicing and preclosure

Parallel splicing

Δ_e is a finite relation over $\mathbb{E}$ such that:

$$u \parallel v \in [\![e]\!] \Leftrightarrow \exists l \; \Delta_e \; r : u \in [\![l]\!] \wedge v \in [\![r]\!].$$

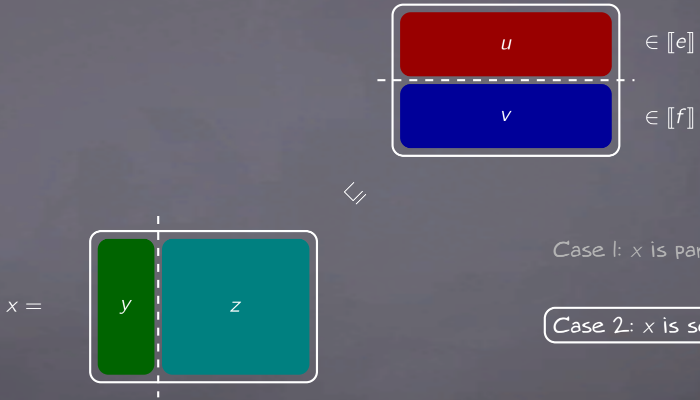
$$e \odot f = e \parallel f + \sum_{l \Delta_{e\parallel f} r} (l{\downarrow}) \parallel (r{\downarrow}) \, .$$

Lemma

$$u \parallel v \in {}^{\sqsubseteq}[\![e \parallel f]\!] \Leftrightarrow u \parallel v \in [\![e \odot f]\!].$$

$$\mathrm{CKA} \vdash e \odot f = e \parallel f.$$

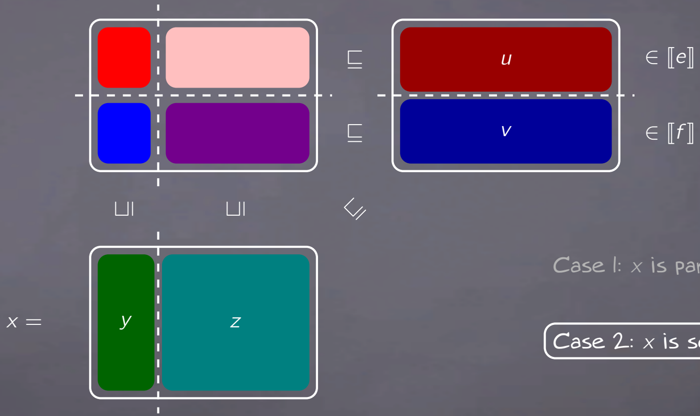
Who's smaller than a parallel product?



Case 1: x is parallel.

Case 2: x is sequential.

Who's smaller than a parallel product?



Case 1: x is parallel.

Case 2: x is sequential.

Sequential splicing

Sequential splicing

∇_e is a finite relation over $\mathbb{E}$ such that:

$$u \cdot v \in [\![e]\!] \Leftrightarrow \exists l \nabla_e r : u \in [\![l]\!] \wedge v \in [\![r]\!].$$

Sequential splicing

Sequential splicing

∇_e is a finite relation over $\mathbb{E}$ such that:

$$u \cdot v \in [\![e]\!] \Leftrightarrow \exists l \; \nabla_e \; r : u \in [\![l]\!] \wedge v \in [\![r]\!].$$

$$u \cdot v \in {}^{\sqsubseteq}[\![e \parallel f]\!] \Leftrightarrow u \cdot v \in [\![e \parallel f + \sum_{\substack{l_e \nabla_e r_e \\ l_f \nabla_f r_f}} (l_e \odot l_f) \cdot (r_e \parallel r_f) {\downarrow}]\!]$$

Sequential splicing

Sequential splicing

∇_e is a finite relation over $\mathbb{E}$ such that:

$$u \cdot v \in [\![e]\!] \Leftrightarrow \exists l \nabla_e r : u \in [\![l]\!] \wedge v \in [\![r]\!].$$

$$u \cdot v \in {}^{\sqsubseteq}[\![e \parallel f]\!] \Leftrightarrow u \cdot v \in [\![e \parallel f + \sum_{\substack{l_e \nabla_e r_e \\ l_f \nabla_f r_f}} (l_e \odot l_f) \cdot (r_e \parallel r_f)\!\downarrow]\!]$$

Problem: $r_e \parallel r_f$ is not always smaller than $e \parallel f$...

And then, some magic happens...

And then, some magic happens...

- We repeat the construction to get successive equations, involving closures.

And then, some magic happens…

- We repeat the construction to get successive equations, involving closures.

- Only a finite number of unknown closures appear.

And then, some magic happens...

- We repeat the construction to get successive equations, involving closures.

- Only a finite number of unknown closures appear.

- These equations can be structured as a linear system.

And then, some magic happens...

- We repeat the construction to get successive equations, involving closures.
- Only a finite number of unknown closures appear.
- These equations can be structured as a linear system.
- With a fancy fixpoint theorem, we compute the least solution of the system.

And then, some magic happens...

- We repeat the construction to get successive equations, involving closures.
- Only a finite number of unknown closures appear.
- These equations can be structured as a linear system.
- With a fancy fixpoint theorem, we compute the least solution of the system.
- This solution is a closure.

Completeness of CKA

Lemma

Every series-rational expression admits a closure.

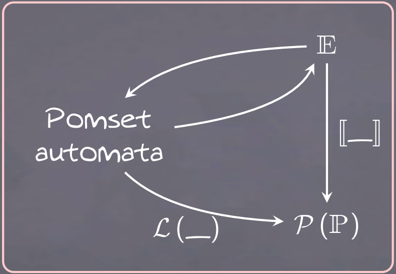
Theorem

$$\mathrm{CKA} \vdash e = f \Leftrightarrow {}^{\sqsubseteq}[\![e]\!] = {}^{\sqsubseteq}[\![f]\!].$$

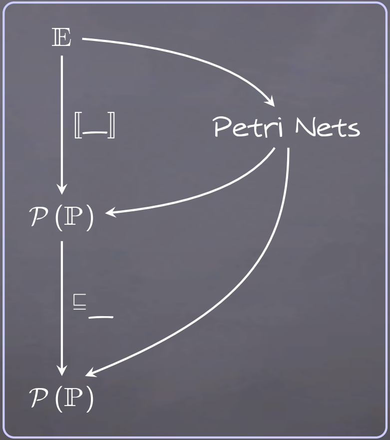
Implementation: `https://doi.org/10.5281/zenodo.926651`.

Outline

1. Brzozowski derivatives



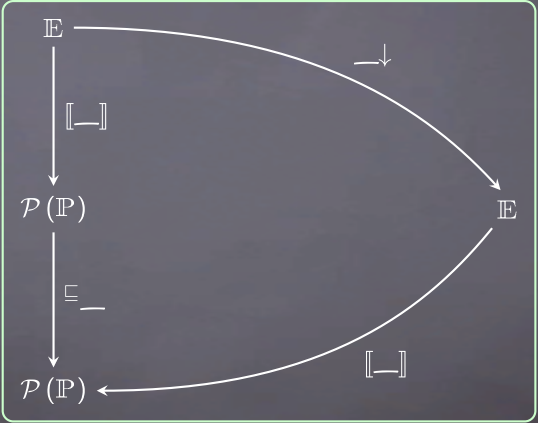
Outline



II. Decidability & Complexity

Outline

III. Completeness



Further questions

Further questions

- Can we extend the algorithm to a larger class of Petri nets?

Further questions

- Can we extend the algorithm to a larger class of Petri nets?
- What about the parallel star?

Further questions

- Can we extend the algorithm to a larger class of Petri nets?
- What about the parallel star?
- Can I have tests?

Further questions

- Can we extend the algorithm to a larger class of Petri nets?
- What about the parallel star?
- Can I have tests?
- Might I dream of adding names?

Further questions

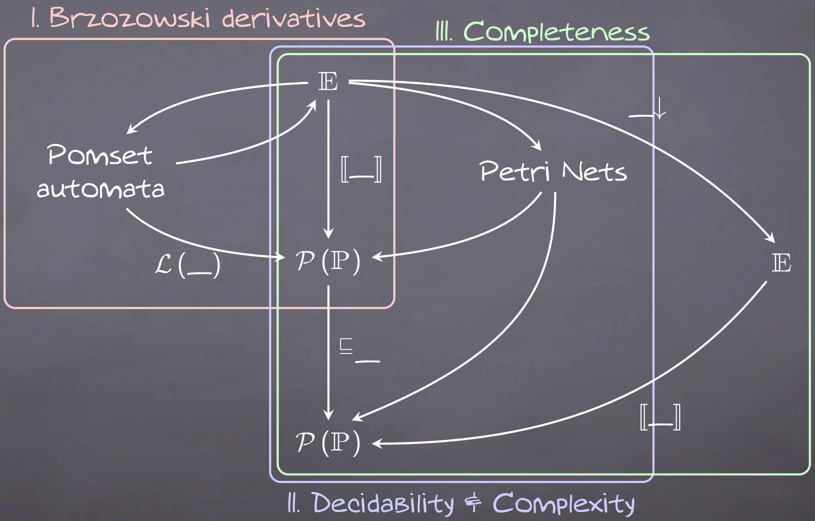
- Can we extend the algorithm to a larger class of Petri nets?
- What about the parallel star?
- Can I have tests?
- Might I dream of adding names?
- Insert you favourite operator here...

That's all folks!

Thank you!

See more at:
http://paul.brunet-zamansky.fr

Outline


I. Brzozowski derivatives
III. Completeness
$\mathbb{E}$
Pomset automata
$[\![\_]\!]$
Petri Nets
$\_\downarrow$
$\mathcal{L}(\_)$
$\mathcal{P}(\mathbb{P})$
$\mathbb{E}$
$\sqsubseteq\_$
$[\![\_]\!]$
$\mathcal{P}(\mathbb{P})$
II. Decidability & Complexity

Outline

I. Introduction

II. Pomset languages and Brzozowski derivatives

III. Decidability & Complexity

IV. Completeness

V. Summary and Outlook

Brzozowski goes concurrent ◂ Back

$$e \star f := \begin{cases} f & \text{if } 1 \in [\![e]\!] \\ 0 & \text{otherwise.} \end{cases}$$

$$[x = y] := \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases}$$

Brzozowski goes concurrent ◂ Back

$$e \star f := \begin{cases} f & \text{if } 1 \in [\![e]\!] \\ 0 & \text{otherwise.} \end{cases} \qquad [x = y] := \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases}$$

- Sequential derivatives: $\delta : \mathbb{E} \times \Sigma \to \mathbb{E}$

$$\delta(0, a) = 0 \qquad \delta(1, a) = 0 \qquad \delta(b, a) = [a = b] \qquad \delta(e^\star, a) = \delta(e, a) \cdot e^\star$$

$$\delta(e \cdot f, a) = \delta(e, a) \cdot f + e \star \delta(f, a) \qquad \delta(e + f, a) = \delta(e, a) + \delta(f, a)$$

$$\delta(e \parallel f, a) = e \star \delta(f, a) + f \star \delta(e, a).$$

Brzozowski goes concurrent (Back)

$$e \star f := \begin{cases} f & \text{if } 1 \in [\![e]\!] \\ 0 & \text{otherwise.} \end{cases} \qquad [x = y] := \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases}$$

- Sequential derivatives: $\delta : \mathbb{E} \times \Sigma \to \mathbb{E}$

$$\delta(0, a) = 0 \qquad \delta(1, a) = 0 \qquad \delta(b, a) = [a = b] \qquad \delta(e^\star, a) = \delta(e, a) \cdot e^\star$$

$$\delta(e \cdot f, a) = \delta(e, a) \cdot f + e \star \delta(f, a) \qquad \delta(e + f, a) = \delta(e, a) + \delta(f, a)$$

$$\delta(e \parallel f, a) = e \star \delta(f, a) + f \star \delta(e, a).$$

- Parallel derivatives: $\gamma : \mathbb{E} \times \binom{\mathbb{E}}{2} \to \mathbb{E}$

$$\gamma(0, \varphi) = 0 \qquad \gamma(b, \varphi) = 0 \qquad \gamma(1, \varphi) = 0 \qquad \gamma(e^\star, \varphi) = \delta(e, \varphi) \cdot e^\star$$

$$\gamma(e \cdot f, \varphi) = \gamma(e, \varphi) \cdot f + e \star \gamma(f, \varphi) \qquad \gamma(e + f, \varphi) = \gamma(e, \varphi) + \gamma(f, \varphi)$$

$$\gamma(e \parallel f, \varphi) = [\varphi = \{\!|e, f|\!\}] + e \star \gamma(f, \varphi) + f \star \gamma(e, \varphi)$$