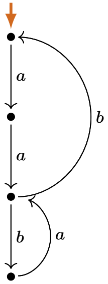
The process semantics of regular expressions

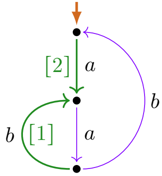
Clemens Grabmayer

Computer Science group
Gran Sasso Science Institute
L'Aquila



Postdoc Seminar

November 19, 2018



Regular Expressions *(Copi–Elgot–Wright, 1958; based on Kleene, 1951)*

Definition

The set $\mathsf{Reg}(A)$ of regular expressions over alphabet A is defined by the grammar:

$$e, f \; ::= \; 0 \mid 1 \mid a \mid (e + f) \mid (e \cdot f) \mid (e^{\star}) \qquad \text{(for } a \in A\text{)}.$$

Regular Expressions *(Copi–Elgot–Wright, 1958; based on Kleene, 1951)*

Definition

The set $\mathsf{Reg}(A)$ of regular expressions over alphabet A is defined by the grammar:

$$e, f \;::=\; 0 \mid 1 \mid a \mid (e + f) \mid (e \cdot f) \mid (e^{\star}) \qquad \text{(for } a \in A\text{)}.$$

Note, here:

- symbol 0 instead of $\varnothing$
- symbol 1 used (often dropped, definable as $0^{\star}$)
- no complementation operation $\overline{e}$

Language semantics $[\![\cdot]\!]_L$ *(Copi–Elgot–Wright, 1958)*

$\mathbf{0} \xmapsto{[\![\cdot]\!]_L}$ empty language $\varnothing$

$\mathbf{1} \xmapsto{[\![\cdot]\!]_L} \{\epsilon\}$ (ϵ the empty word)

$a \xmapsto{[\![\cdot]\!]_L} \{a\}$

Language semantics $[\![\cdot]\!]_L$ *(Copi–Elgot–Wright, 1958)*

$\mathbf{0} \xmapsto{[\![\cdot]\!]_L}$ empty language $\varnothing$

$\mathbf{1} \xmapsto{[\![\cdot]\!]_L} \{\epsilon\}$ (ϵ the empty word)

$a \xmapsto{[\![\cdot]\!]_L} \{a\}$

$e + f \xmapsto{[\![\cdot]\!]_L}$ union of $[\![e]\!]_L$ and $[\![f]\!]_L$

$e \cdot f \xmapsto{[\![\cdot]\!]_L}$ element-wise concatenation of $[\![e]\!]_L$ and $[\![f]\!]_L$

$e^* \xmapsto{[\![\cdot]\!]_L}$ set of words formed by concatenating words in $[\![e]\!]_L$ plus the empty word ϵ

Process semantics $[\![\cdot]\!]_P$ *(Milner, 1984)*

$\mathbf{0} \xmapsto{[\![\cdot]\!]_P}$ deadlock $\boldsymbol{\delta}$, no termination

$\mathbf{1} \xmapsto{[\![\cdot]\!]_P}$ empty process $\boldsymbol{\epsilon}$, then terminate

$a \xmapsto{[\![\cdot]\!]_P}$ atomic action a, then terminate

Process semantics $[\![\cdot]\!]_{\boldsymbol{P}}$ *(Milner, 1984)*

$\mathbf{0} \xmapsto{[\![\cdot]\!]_{\boldsymbol{P}}}$ deadlock $\boldsymbol{\delta}$, no termination

$\mathbf{1} \xmapsto{[\![\cdot]\!]_{\boldsymbol{P}}}$ empty process $\boldsymbol{\epsilon}$, then terminate

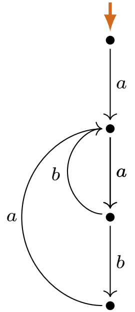
$a \xmapsto{[\![\cdot]\!]_{\boldsymbol{P}}}$ atomic action a, then terminate

$e + f \xmapsto{[\![\cdot]\!]_{\boldsymbol{P}}}$ alternative composition of $[\![e]\!]_{\boldsymbol{P}}$ and $[\![f]\!]_{\boldsymbol{P}}$

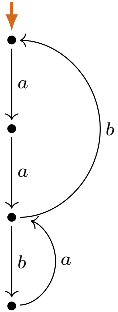
$e \cdot f \xmapsto{[\![\cdot]\!]_{\boldsymbol{P}}}$ sequential composition of $[\![e]\!]_{\boldsymbol{P}}$ and $[\![f]\!]_{\boldsymbol{P}}$

$e^* \xmapsto{[\![\cdot]\!]_{\boldsymbol{P}}}$ unbounded iteration of $[\![e]\!]_{\boldsymbol{P}}$, option to terminate

Process semantics $[\![\cdot]\!]_P$ (examples)

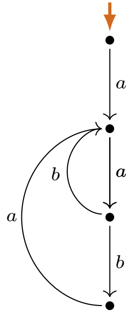


$$a(a(b+ba))^*0$$

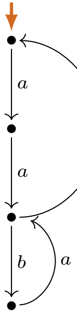


$$(aa(ba)^*b)^*0$$

Process semantics $[\![\cdot]\!]_P$ (examples)

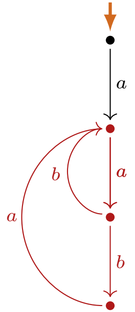


$$a \cdot (a \cdot (b + b \cdot a))^* \cdot 0$$

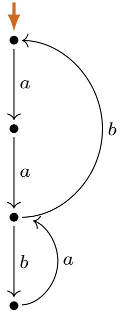


$$(a \cdot a \cdot (b \cdot a)^* \cdot b)^* \cdot 0$$

Process semantics $[\![\cdot]\!]_P$ (examples)

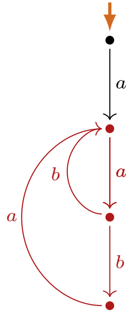


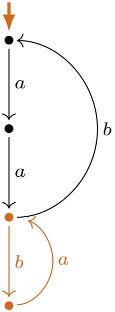
$a \cdot (a \cdot (b + b \cdot a))^* \cdot 0$



$(a \cdot a \cdot (b \cdot a)^* \cdot b)^* \cdot 0$

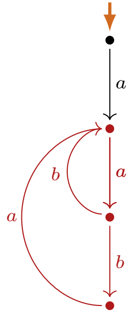
Process semantics $[\![\cdot]\!]_P$ (examples)



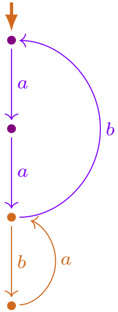


$$a \cdot (a \cdot (b + b \cdot a))^* \cdot 0 \qquad\qquad (a \cdot a \cdot (b \cdot a)^* \cdot b)^* \cdot 0$$

Process semantics $[\![\cdot]\!]_P$ (examples)

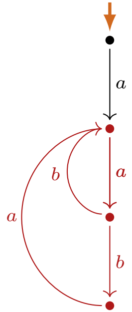


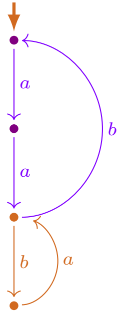
$$a \cdot (a \cdot (b + b \cdot a))^* \cdot 0$$



$$(a \cdot a \cdot (b \cdot a)^* \cdot b)^* \cdot 0$$

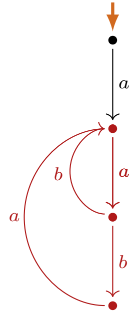
Process semantics $[\![\cdot]\!]_P$ (examples)

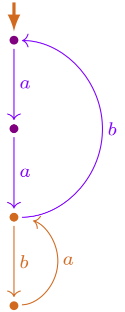




$[\![a \cdot (a \cdot (b + b \cdot a))^* \cdot 0]\!]_P$ $[\![(a \cdot a \cdot (b \cdot a)^* \cdot b)^* \cdot 0]\!]_P$

Process semantics $[\![\cdot]\!]_P$ (examples)

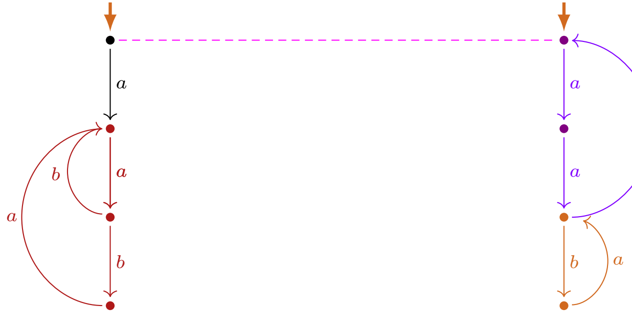




$$[\![a(a(b+ba))^*0]\!]_P$$

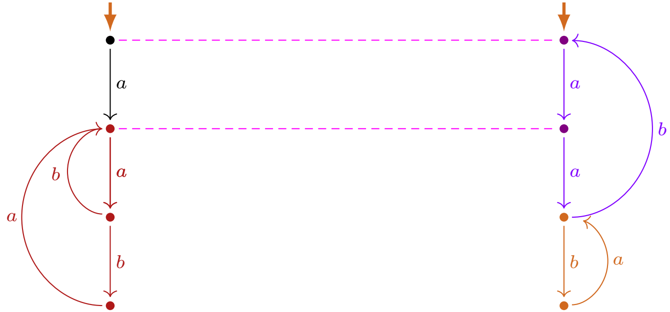
$$[\![(aa(ba)^*b)^*0]\!]_P$$

Process semantics $[\![\cdot]\!]_P$ (examples)



$[\![a(a(b+ba))^*0]\!]_P$ $[\![(aa(ba)^*b)^*0]\!]_P$

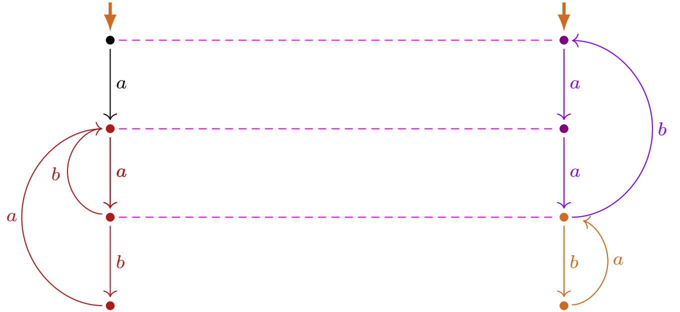
Process semantics $[\![\cdot]\!]_{P}$ (examples)



$[\![a(a(b+ba))^{*}0]\!]_{P}$

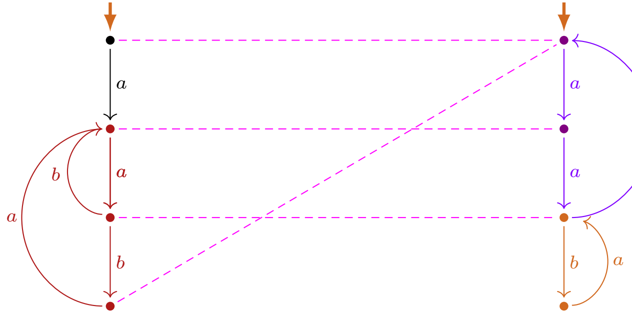
$[\![(aa(ba)^{*}b)^{*}0]\!]_{P}$

Process semantics $[\![\cdot]\!]_P$ (examples)



$[\![a(a(b+ba))^*0]\!]_P$ $[\![(aa(ba)^*b)^*0]\!]_P$

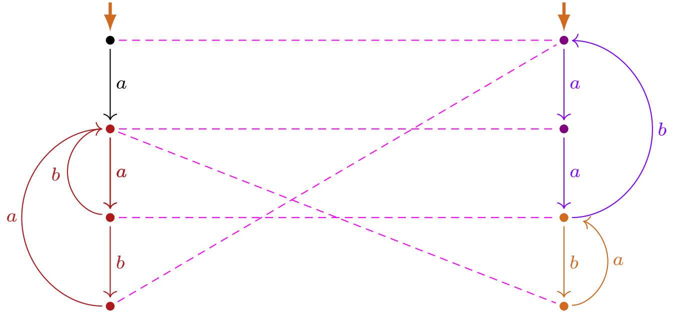
Process semantics $[\![\cdot]\!]_P$ (examples)



$[\![a(a(b+ba))^*0]\!]_P$

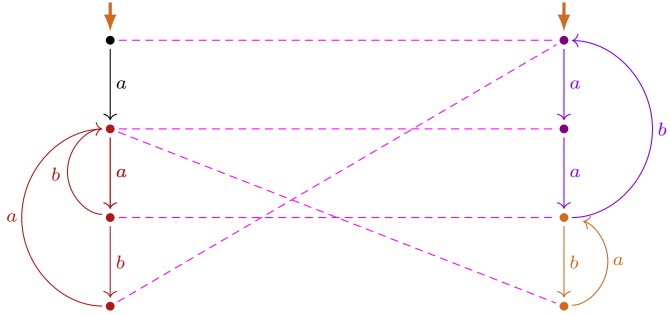
$[\![(aa(ba)^*b)^*0]\!]_P$

Process semantics $[\![\cdot]\!]_{\boldsymbol{P}}$ (examples)



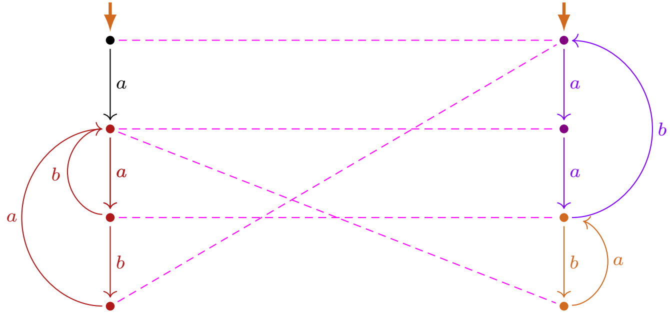
$[\![a(a(b+ba))^{*}0]\!]_{\boldsymbol{P}}$ $[\![(aa(ba)^{*}b)^{*}0]\!]_{\boldsymbol{P}}$

Process semantics $[\![\cdot]\!]_P$ (examples)



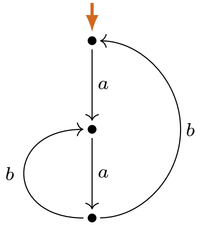
$$[\![a(a(b+ba))^*0]\!]_P \quad \underline{\leftrightarrow} \quad [\![(aa(ba)^*b)^*0]\!]_P$$

Process semantics $[\![\cdot]\!]_P$ (examples)

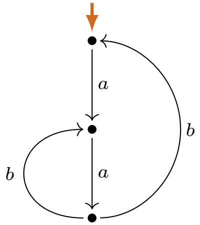


$$a(a(b+ba))^*0 \quad \underline{\leftrightarrow}_P \quad (aa(ba)^*b)^*0$$

Expressible process graphs (under bisimulation $\underline{\leftrightarrow}$)

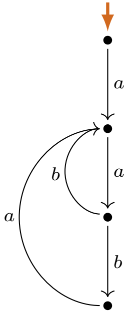


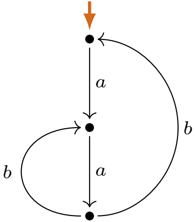
Expressible process graphs (under bisimulation $\underline{\leftrightarrow}$)



$$\notin im(\llbracket \cdot \rrbracket_{\boldsymbol{P}})$$

Expressible process graphs (under bisimulation $\leftrightarrows$)

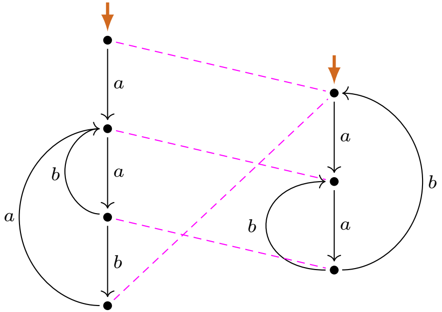




$\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\notin im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$

$\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible

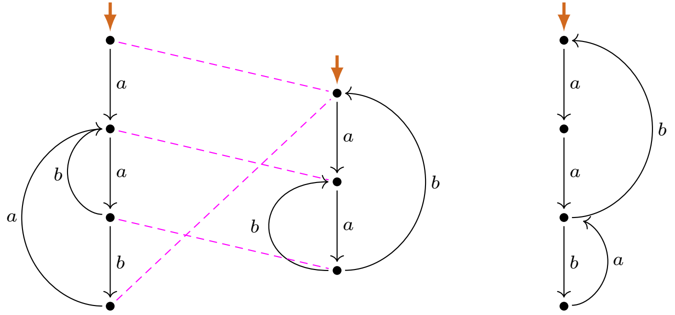
Expressible process graphs (under bisimulation $\underline{\leftrightarrow}$)



$\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\notin im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$

$\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible

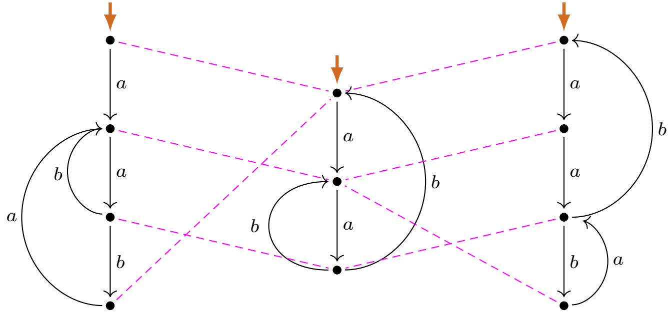
Expressible process graphs (under bisimulation $\underline{\leftrightarrow}$)



$\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\notin im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$

$\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible $\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible

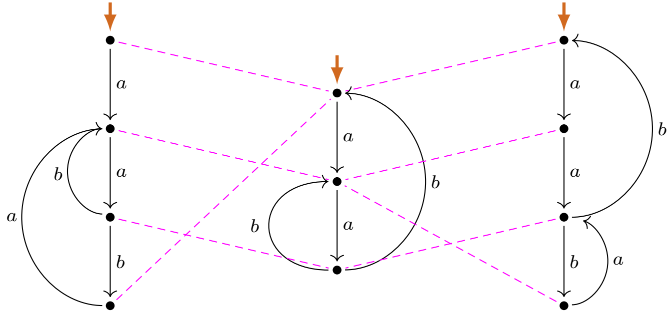
Expressible process graphs (under bisimulation $\leftrightarrow$)



$\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\notin im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$

$\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible $\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible

Expressible process graphs (under bisimulation $\underline{\leftrightarrow}$)



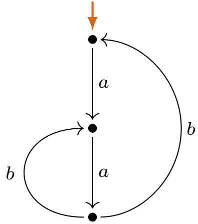
$\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible

$\notin im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$

$\in im(\llbracket\cdot\rrbracket_{\boldsymbol{P}})$ $\llbracket\cdot\rrbracket_{\boldsymbol{P}}$-expressible

Properties of $\boldsymbol{P}$

- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible.

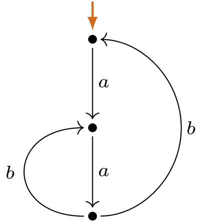


not $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible

$[\![\cdot]\!]_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$

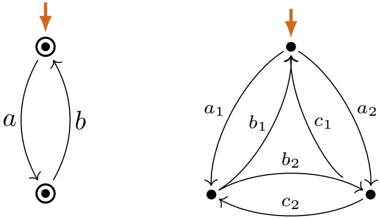
Properties of $\boldsymbol{P}$

- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible.
- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$.



not $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible

$[\![\cdot]\!]_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$



not $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible

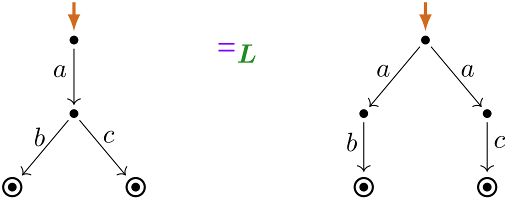
not $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$

Properties of P

- Not every finite-state process is $[\![\cdot]\!]_P$-expressible.
- Not every finite-state process is $[\![\cdot]\!]_P$-expressible modulo $\underline{\leftrightarrow}$.
- Fewer identities hold for $\underline{\leftrightarrow}_P$ than for $=_L$: $\underline{\leftrightarrow}_P \subsetneqq =_L$.

Properties of $\boldsymbol{P}$

- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible.
- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$.
- Fewer identities hold for $\underline{\leftrightarrow}_{\boldsymbol{P}}$ than for $=_{\boldsymbol{L}}$: $\underline{\leftrightarrow}_{\boldsymbol{P}} \subsetneqq =_{\boldsymbol{L}}$.



$$a \cdot (b + c) \quad =_{\boldsymbol{L}} \quad a \cdot b + a \cdot c$$

Properties of $\boldsymbol{P}$

- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible.
- Not every finite-state process is $[\![\cdot]\!]_{\boldsymbol{P}}$-expressible modulo $\underline{\leftrightarrow}$.
- Fewer identities hold for $\underline{\leftrightarrow}_{\boldsymbol{P}}$ than for $=_{\boldsymbol{L}}$: $\quad \underline{\leftrightarrow}_{\boldsymbol{P}} \subsetneqq =_{\boldsymbol{L}}$.

$$a \cdot (b + c) \quad \not\underline{\leftrightarrow}_{\boldsymbol{P}} \quad a \cdot b + a \cdot c$$

Complete axiomatization of $=_{\boldsymbol{L}}$ *(Aanderaa/Salomaa, 1965/66)*

Axioms :

(B1) $e + (f + g) = (e + f) + g$

(B2) $(e \cdot f) \cdot g = e \cdot (f \cdot g)$

(B3) $e + f = f + e$

(B4) $(e + f) \cdot g = e \cdot g + f \cdot g$

(B5) $e \cdot (f + g) = e \cdot f + e \cdot g$

(B6) $e + e = e$

(B7) $e \cdot 1 = e$

(B8) $e \cdot 0 = 0$

(B9) $e + 0 = e$

(B10) $e^* = 1 + e \cdot e^*$

(B11) $e^* = (1 + e)^*$

Inference rules : equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \text{FIX} \quad (\text{if } \underbrace{\{\epsilon\} \notin [\![f]\!]_{\boldsymbol{L}}}_{\textit{non-empty-word property}})$$

Sound and unsound axioms with respect to $\underline{\leftrightarrow}_P$

Axioms :

(B1)	$e + (f + g) = (e + f) + g$	(B7)	$e \cdot 1 = e$
(B2)	$(e \cdot f) \cdot g = e \cdot (f \cdot g)$	(B8)	$e \cdot 0 = 0$
(B3)	$e + f = f + e$	(B9)	$e + 0 = e$
(B4)	$(e + f) \cdot g = e \cdot g + f \cdot g$	(B10)	$e^* = 1 + e \cdot e^*$
(B5)	$e \cdot (f + g) = e \cdot f + e \cdot g$	(B11)	$e^* = (1 + e)^*$
(B6)	$e + e = e$		

Inference rules : equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \mathrm{FIX} \quad (\text{if } \underbrace{\{\epsilon\} \notin [\![f]\!]_L}_{\textit{non-empty-word property}})$$

Sound and unsound axioms with respect to $\underline{\leftrightarrow}_{P}$

Axioms :

(B1)	$e + (f + g) = (e + f) + g$	(B7)	$e \cdot 1 = e$
(B2)	$(e \cdot f) \cdot g = e \cdot (f \cdot g)$	(B8)	$e \cdot 0 = 0$
(B3)	$e + f = f + e$	(B9)	$e + 0 = e$
(B4)	$(e + f) \cdot g = e \cdot g + f \cdot g$	(B10)	$e^* = 1 + e \cdot e^*$
(B5)	$e \cdot (f + g) = e \cdot f + e \cdot g$	(B11)	$e^* = (1 + e)^*$
(B6)	$e + e = e$	(B8)′	$0 \cdot e = 0$

Inference rules : equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \text{FIX} \quad (\text{if } \underbrace{\{\epsilon\} \notin [\![f]\!]_L}_{\text{non-empty-word property}})$$

Adaptation for $\underline{\leftrightarrow}_{P}$ *(Milner, 1984)* (Mil = Mil⁻ + RSP*)

Axioms :

(B1)	$e + (f + g) = (e + f) + g$	(B7)	$e \cdot 1 = e$
(B2)	$(e \cdot f) \cdot g = e \cdot (f \cdot g)$		
(B3)	$e + f = f + e$	(B9)	$e + 0 = e$
(B4)	$(e + f) \cdot g = e \cdot g + f \cdot g$	(B10)	$e^* = 1 + e \cdot e^*$
		(B11)	$e^* = (1 + e)^*$
(B6)	$e + e = e$	(B8)′	$0 \cdot e = 0$

Inference rules : equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \mathrm{RSP}^* \quad (\text{if } \underbrace{\{\epsilon\} \notin [\![f]\!]_{L}}_{\textit{non-empty-word property}})$$

Adaptation for $\underline{\leftrightarrow}_{P}$ (Milner, 1984) (Mil = Mil$^-$ + RSP*)

Axioms :

(B1)	$e + (f + g) = (e + f) + g$	(B7)	$e \cdot 1 = e$
(B2)	$(e \cdot f) \cdot g = e \cdot (f \cdot g)$	(B8)′	$0 \cdot e = 0$
(B3)	$e + f = f + e$	(B9)	$e + 0 = e$
(B4)	$(e + f) \cdot g = e \cdot g + f \cdot g$	(B10)	$e^* = 1 + e \cdot e^*$
		(B11)	$e^* = (1 + e)^*$
(B6)	$e + e = e$		

Inference rules : equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \text{RSP}^* \ \Big(\text{if } \underbrace{\{\epsilon\} \notin [\![f]\!]_{L}}_{\text{non-empty-word property}}\Big)$$

Adaptation for $\underline{\leftrightarrow}_{P}$ *(Milner, 1984)* (Mil = Mil⁻ + RSP*)

Axioms :

(B1)	$e + (f + g) = (e + f) + g$	(B7)	$e \cdot 1 = e$
(B2)	$(e \cdot f) \cdot g = e \cdot (f \cdot g)$	(B8)′	$0 \cdot e = 0$
(B3)	$e + f = f + e$	(B9)	$e + 0 = e$
(B4)	$(e + f) \cdot g = e \cdot g + f \cdot g$	(B10)	$e^* = 1 + e \cdot e^*$
		(B11)	$e^* = (1 + e)^*$
(B6)	$e + e = e$		

Inference rules : equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \mathsf{RSP}^* \ \left(\text{if } \underbrace{\{\epsilon\} \notin [\![f]\!]_{L}}_{\textit{non-empty-word property}}\right)$$

Milner's questions

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

Milner's questions

Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

Milner's questions, and partial results

Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

Milner's questions, and partial results

Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

- definability by well-behaved specifications *(Baeten/Corradini, 2005)*

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

Milner's questions, and partial results

Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

- definability by well-behaved specifications *(Baeten/Corradini, 2005)*
- that is decidable (super–exponentially) *(Baeten/Corradini/G, 2007)*

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

Milner's questions, and partial results

Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

- definability by well-behaved specifications *(Baeten/Corradini, 2005)*
- that is decidable (super–exponentially) *(Baeten/Corradini/G, 2007)*

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

- Mil is complete for perpetual–loop expressions *(Fokkink, 1996)*
  - every iteration e^* occurs as part of a *'no-exit'* subexpression $e^* \cdot 0$

Milner's questions, and partial results

Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

- definability by well-behaved specifications *(Baeten/Corradini, 2005)*
- that is decidable (super–exponentially) *(Baeten/Corradini/G, 2007)*

Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

- Mil is complete for perpetual–loop expressions *(Fokkink, 1996)*
  - every iteration e^* occurs as part of a *'no-exit'* subexpression $e^* \cdot 0$
- Mil is complete when restricted to 1-return-less expressions *(Corradini, De Nicola, Labella, 2002)*

Milner's questions, and partial results

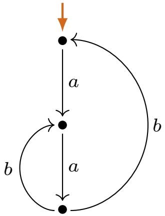
Q1. Recognition: Which structural property of finite process graphs characterizes $[\![\cdot]\!]_P$-expressibility modulo $\underline{\leftrightarrow}$?

- definability by well-behaved specifications *(Baeten/Corradini, 2005)*
- that is decidable (super–exponentially) *(Baeten/Corradini/G, 2007)*

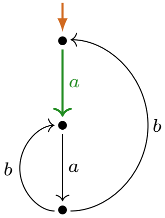
Q2. Complete axiomatization: Is Mil complete for $\underline{\leftrightarrow}_P$?

- Mil is complete for perpetual–loop expressions *(Fokkink, 1996)*
  - every iteration e^* occurs as part of a *'no-exit'* subexpression $e^* \cdot 0$
- Mil is complete when restricted to 1-return-less expressions *(Corradini, De Nicola, Labella, 2002)*
- Mil⁻ + one of two stronger rules (than RSP*) is complete *(G, 2006)*

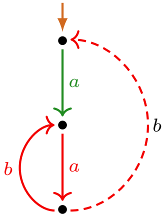
New approach: Loop Existence and Elimination (LEE)



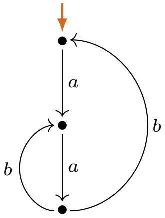
New approach: Loop Existence and Elimination (LEE)



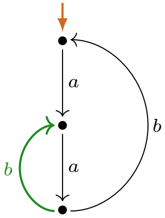
New approach: Loop Existence and Elimination (LEE)



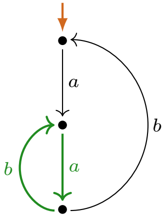
New approach: Loop Existence and Elimination (LEE)



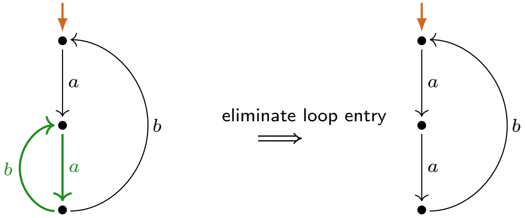
New approach: Loop Existence and Elimination (LEE)



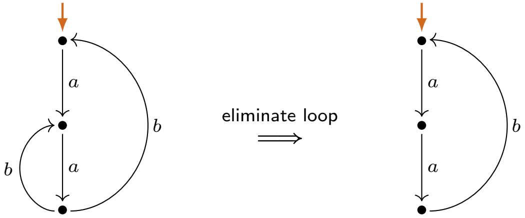
New approach: Loop Existence and Elimination (LEE)



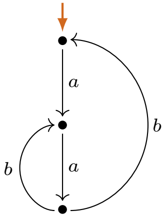
New approach: Loop Existence and Elimination (LEE)



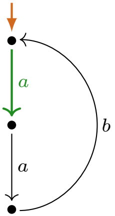
New approach: Loop Existence and Elimination (LEE)



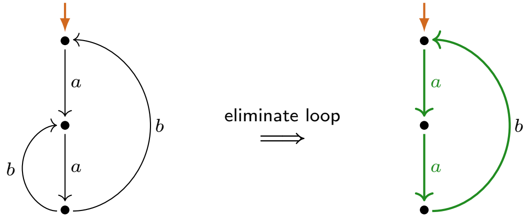
New approach: Loop Existence and Elimination (LEE)



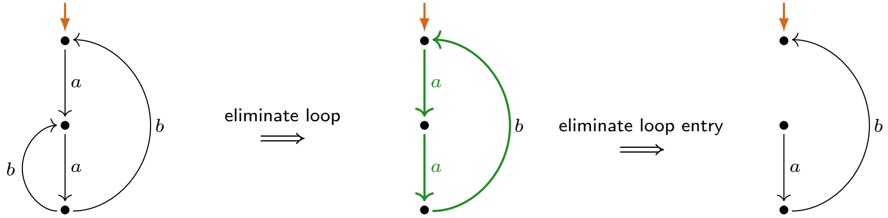
eliminate loop $\Longrightarrow$



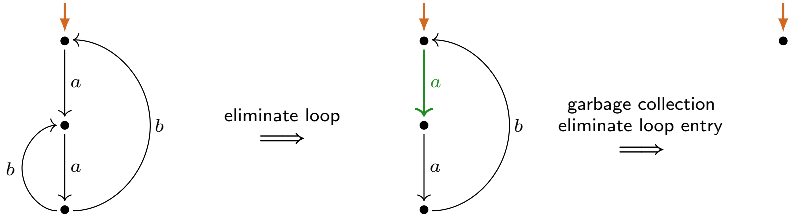
New approach: Loop Existence and Elimination (LEE)



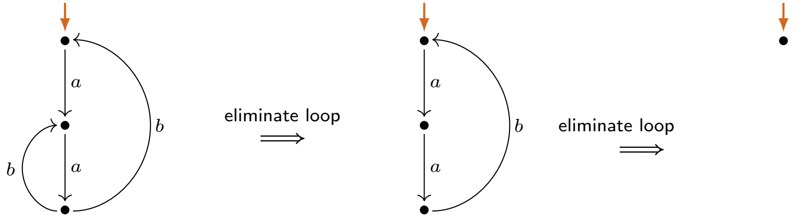
New approach: Loop Existence and Elimination (LEE)



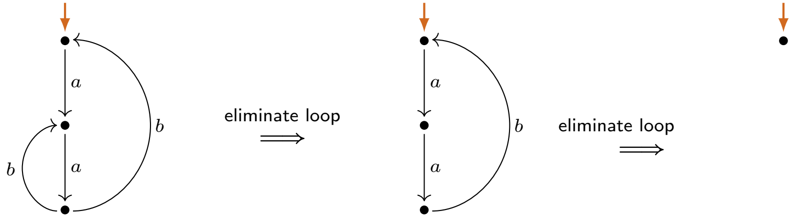
New approach: Loop Existence and Elimination (LEE)



New approach: Loop Existence and Elimination (LEE)



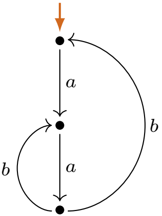
New approach: Loop Existence and Elimination (LEE)



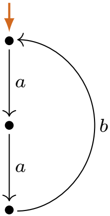
LEE:

repeated elimination of loops leads to a process graph without infinite trace

New approach: Loop Existence and Elimination (LEE)



eliminate loop $\Longrightarrow$

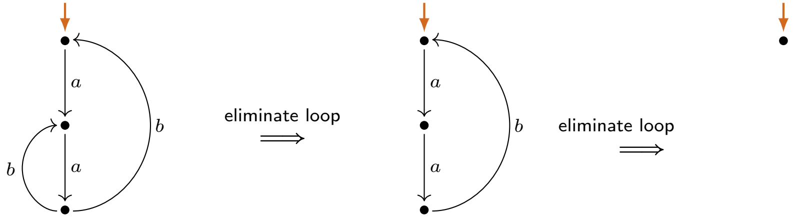


eliminate loop $\Longrightarrow$

LEE:

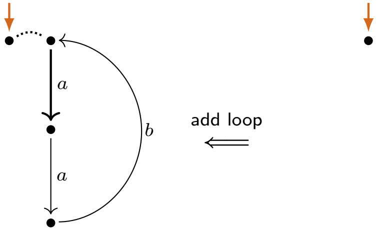
repeated elimination of loops leads to a process graph without infinite trace

New approach: Loop Existence and Elimination (LEE)

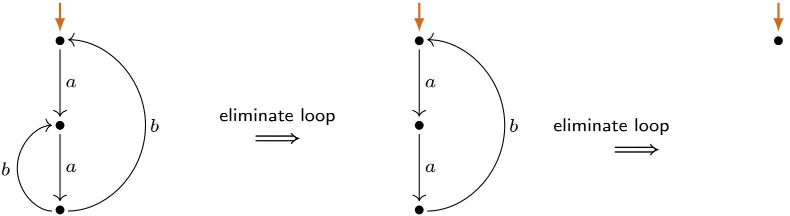


LEE:

repeated elimination of loops leads to a process graph without infinite trace

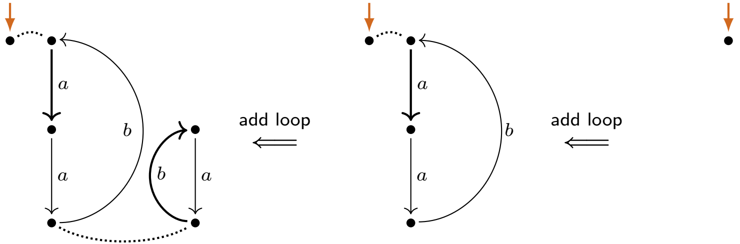


New approach: Loop Existence and Elimination (LEE)

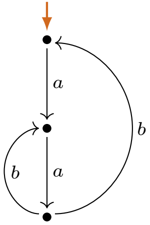


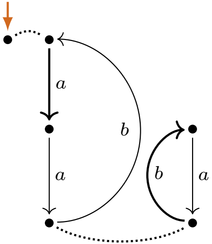
LEE:

repeated elimination of loops leads to a process graph without infinite trace



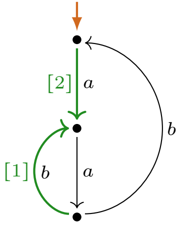
LEE-witness



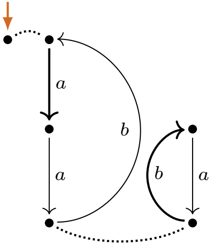


structured
LEE-witness

LEE-witness: structure constrained process graph

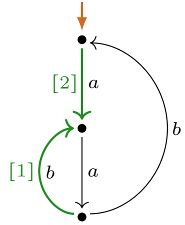


LEE-witness

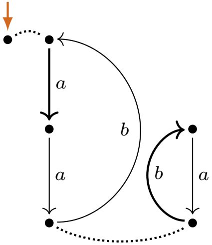


structured
LEE-witness

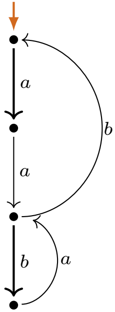
LEE-witness yields star expression



LEE-witness



structured
LEE-witness

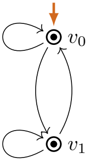


$[\![(aa(ba)^*b)^*0]\!]_{\boldsymbol{P}}$

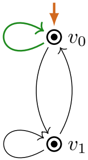
structure constraints (L'Aquila)

structure constraints (L'Aquila)

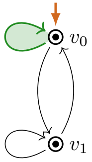
Loop elimination



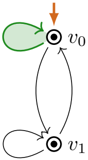
Loop elimination



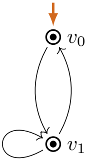
Loop elimination



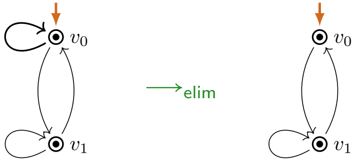
Loop elimination



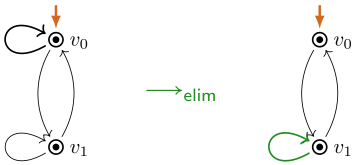
$\longrightarrow_{\text{elim}}$



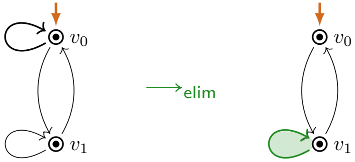
Loop elimination



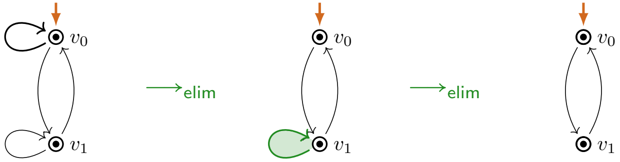
Loop elimination



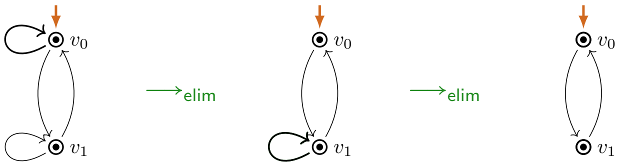
Loop elimination



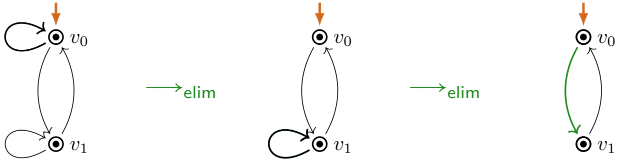
Loop elimination



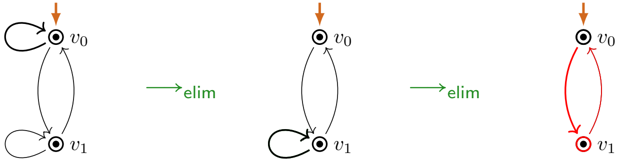
Loop elimination



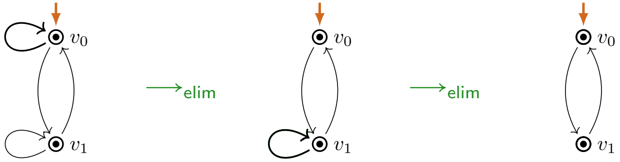
Loop elimination



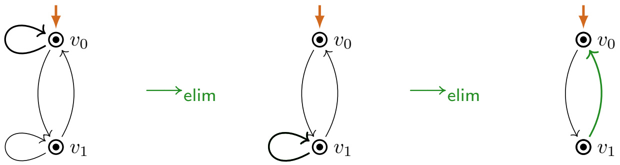
Loop elimination



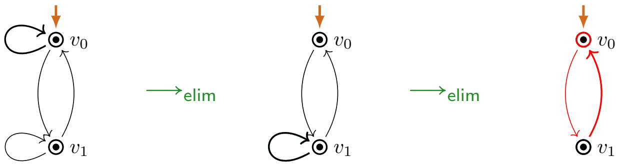
Loop elimination



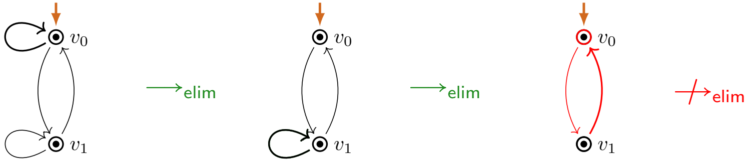
Loop elimination



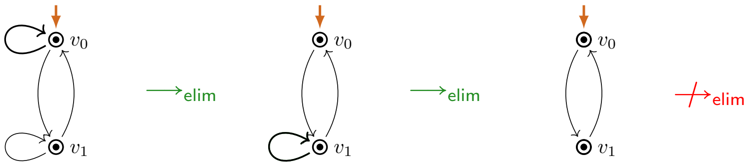
Loop elimination



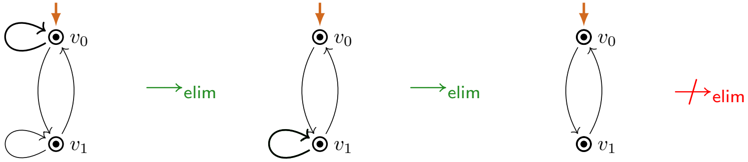
Loop elimination

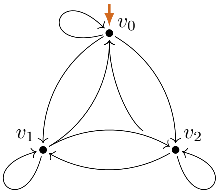


Loop elimination

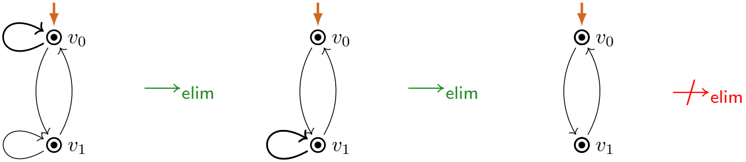


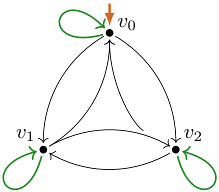
Loop elimination



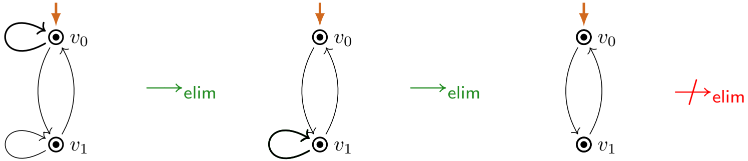


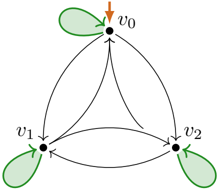
Loop elimination



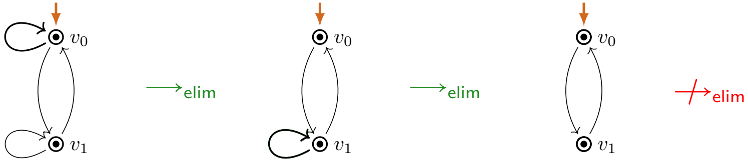


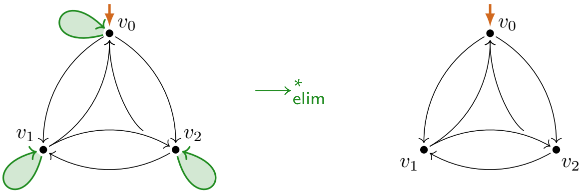
Loop elimination



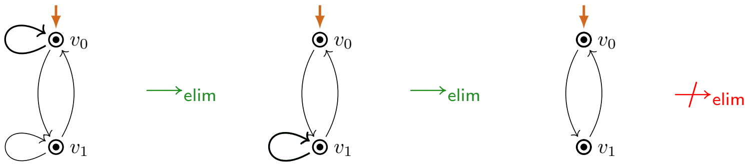


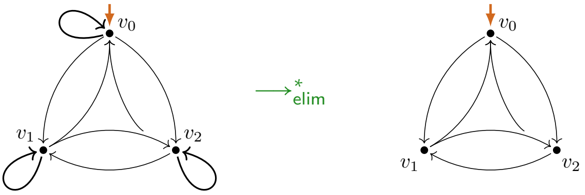
Loop elimination



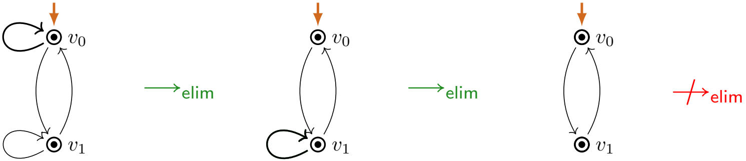


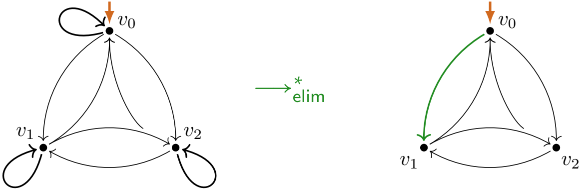
Loop elimination



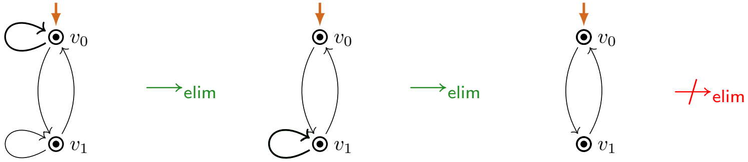


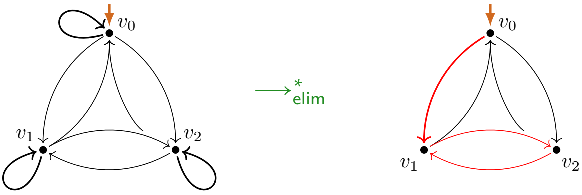
Loop elimination



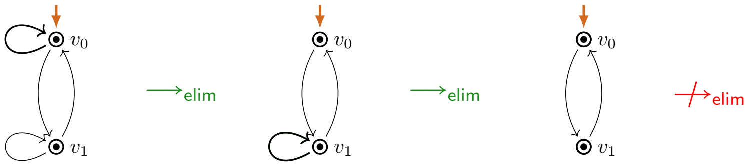


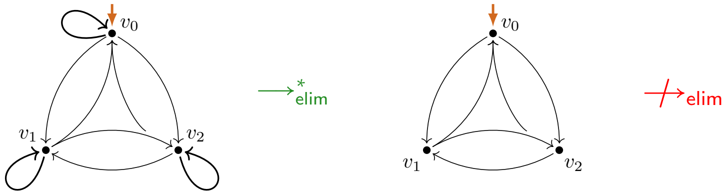
Loop elimination



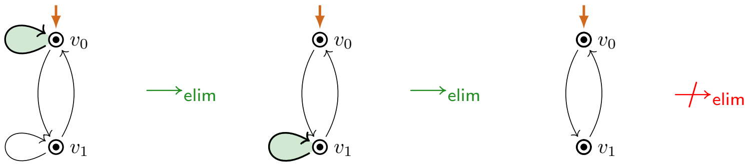


Loop elimination



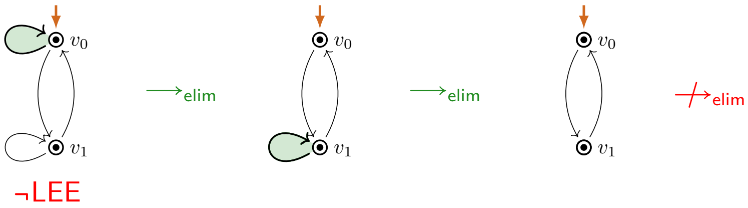


LEE fails



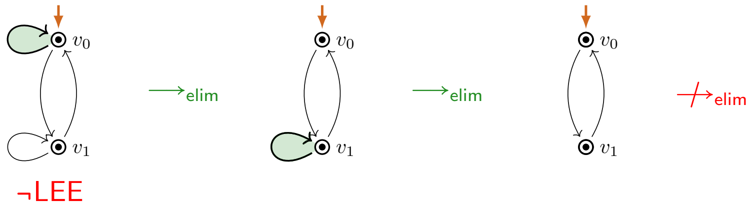
v_0
v_1
$\longrightarrow_{\text{elim}}$
v_0
v_1
$\longrightarrow_{\text{elim}}$
v_0
v_1
$\not\longrightarrow_{\text{elim}}$

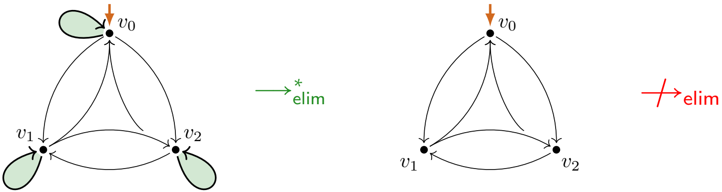
LEE fails



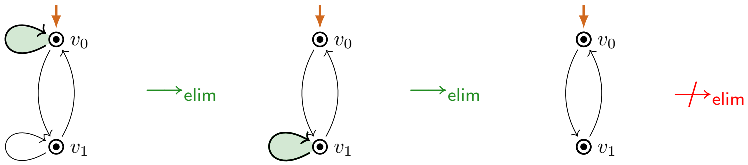
v_0
v_1
$\neg$LEE
$\longrightarrow_{\text{elim}}$
v_0
v_1
$\longrightarrow_{\text{elim}}$
v_0
v_1
$\not\longrightarrow_{\text{elim}}$

LEE fails

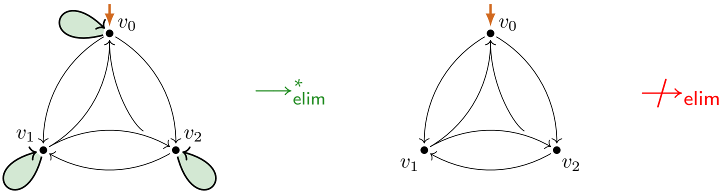

v_0
v_1
$\longrightarrow_{\text{elim}}$
v_0
v_1
$\longrightarrow_{\text{elim}}$
v_0
v_1
$\not\longrightarrow_{\text{elim}}$
$\neg$LEE


v_0
v_1
v_2
$\longrightarrow^{*}_{\text{elim}}$
v_0
v_1
v_2
$\not\longrightarrow_{\text{elim}}$

LEE fails



¬LEE

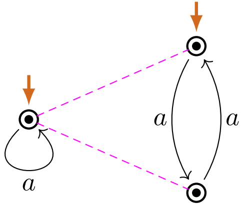


¬LEE

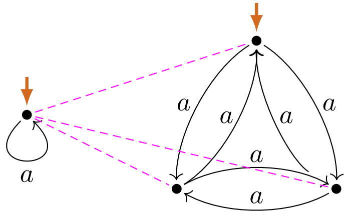
LEE under bisimulation

Observation

- LEE is not invariant under bisimulation.
- LEE is not preserved by converse functional bisimulation.

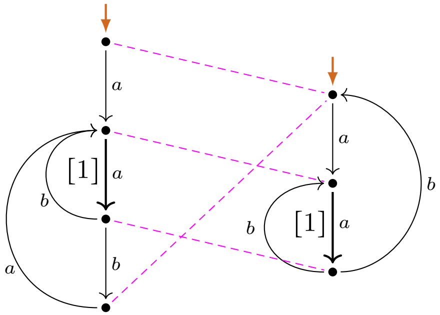


LEE ¬LEE

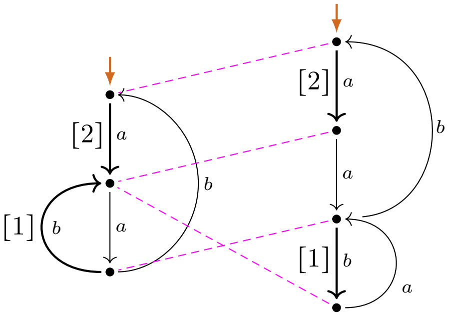


LEE ¬LEE

LEE is preserved under bisimulation collapse



$[\![a(a(b+ba))^*0]\!]_{\boldsymbol{P}}$



$[\![(aa(ba)^*b)^*0]\!]_{\boldsymbol{P}}$

Goals

- Completeness proof
  - is a large project: report (now ~ 250 pages, ~ 2 years)
  - writing up a crucial step:
    - pseudo-collapse LEE-witnesses with 1-transitions
- Interpretations of equational theories
  - Characterize interpretations with respect to what kinds of properties they permit to transfer between equational theories

Goals

- Completeness proof
  - is a large project: report (now ~ 250 pages, ~ 2 years)
  - writing up a crucial step:
    - pseudo-collapse LEE-witnesses with 1-transitions
- Interpretations of equational theories
  - Characterize interpretations with respect to what kinds of properties they permit to transfer between equational theories
- Recognition problem
  - polynomial if no 1-transitions (testing for property LEE)
- Cost models for λ-calculus