The Graph Structure of Process Interpretations of Regular Expressions

Clemens Grabmayer

https://clegra.github.io

Department of Computer Science

GRAN SASSO SCIENCE INSTITUTE

SCHOOL OF ADVANCED STUDIES

Scuola Universitaria Superiore

L'Aquila, Italy

IFIP 1.6 Working Group Meeting

Nancy

July 1, 2024

Overview

- regular expressions (unary/binary star/1-free-under-star $(*/\!\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- loop existence and elimination (LEE)
  - defined by loop elimination rewrite system, its completion
  - describes interpretations of $(*/\!\not{1})$ reg. expr.s (extraction possible)
  - LEE-witnesses: labelings of process graphs with LEE
  - LEE is preserved under bisimulation collapse (stepwise collapse)
- 1-LEE = sharing via 1-transitions facilitates LEE
- LEE/1-LEE characterize image of $P^{\bullet}$ (restricted/unrestricted)
  - where $P^{\bullet}$ a compact (sharing-increased) refinement of P
- outlook on work-to-do

Overview

- regular expressions (unary/binary star/1-free-under-star $(*/\cancel{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- loop existence and elimination (LEE)
  - defined by loop elimination rewrite system, its completion
  - describes interpretations of $(*/\cancel{1})$ reg. expr.s (extraction possible)
  - LEE-witnesses: labelings of process graphs with LEE
  - LEE is preserved under bisimulation collapse (stepwise collapse)
- 1-LEE = sharing via 1-transitions facilitates LEE
  - describes interpretations of all reg. expr.s (extraction possible)
  - not preserved under bisimulation collapse (approximation possible)
- LEE/1-LEE characterize image of $P^{\bullet}$ (restricted/unrestricted)
  - where $P^{\bullet}$ a compact (sharing-increased) refinement of P
  - via refined extraction using LEE/1-LEE
- outlook on work-to-do

Regular Expressions

Definition (~ *Copi–Elgot–Wright, 1958* **)**

Regular expressions over alphabet A with unary Kleene star:

$$e, e_1, e_2 ::= \mathbf{0} \mid \boldsymbol{a} \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^* \qquad \text{(for } \boldsymbol{a} \in A\text{).}$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$

Regular Expressions

Definition (~ *Copi–Elgot–Wright, 1958*)

Regular expressions over alphabet A with unary Kleene star:

$$e, e_1, e_2 \ ::= \ \mathbf{0} \ | \ \boldsymbol{a} \ | \ e_1 + e_2 \ | \ e_1 \cdot e_2 \ | \ e^* \qquad \text{(for } \boldsymbol{a} \in A\text{).}$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$
- with unary star *: $\mathbf{1}$ is definable as $\mathbf{0}^*$

Regular Expressions

Definition (~ *Kleene, 1951*, ~ *Copi–Elgot–Wright, 1958*)

Regular expressions over alphabet A with unary / binary Kleene star:

$$e, e_1, e_2 \ ::= \ \mathbf{0} \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^{*} \qquad \text{(for } a \in A\text{)}.$$

$$e, e_1, e_2 \ ::= \ \mathbf{0} \mid \mathbf{1} \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e_1{}^{\circledast} e_2 \qquad \text{(for } a \in A\text{)}.$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$
- with unary star *: $\mathbf{1}$ is definable as $\mathbf{0}^{*}$
- with binary star $^{\circledast}$: $\mathbf{1}$ is not definable (in its absence)

Regular Expressions

Definition (~ *Kleene, 1951*, ~ *Copi–Elgot–Wright, 1958*)

Regular expressions over alphabet A with unary / binary Kleene star:

$$e, e_1, e_2 ::= \mathbf{0} \mid \mathbf{1} \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^{*} \qquad (\text{for } a \in A).$$

$$e, e_1, e_2 ::= \mathbf{0} \mid \mathbf{1} \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e_1{}^{\circledast} e_2 \qquad (\text{for } a \in A).$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$
- with unary star *: $\mathbf{1}$ is definable as $\mathbf{0}^{*}$
- with binary star $^{\circledast}$: $\mathbf{1}$ is not definable (in its absence)

Regular Expressions (1-free)

Definition (~ *Kleene, 1951*, ~ *Copi–Elgot–Wright, 1958*)

Regular expressions over alphabet A with unary / binary Kleene star:

$$e, e_1, e_2 \;::=\; \mathbf{0} \mid \mathbf{1} \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^{*} \qquad (\text{for } a \in A).$$

$$e, e_1, e_2 \;::=\; \mathbf{0} \mid \mathbf{1} \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e_1{}^{\circledast} e_2 \qquad (\text{for } a \in A).$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$
- with unary star *: $\mathbf{1}$ is definable as $\mathbf{0}^{*}$
- with binary star $^{\circledast}$: $\mathbf{1}$ is not definable (in its absence)

Definition (for process interpretation)

1-free regular expressions over alphabet A with binary Kleene star:

$$f, f_1, f_2 \;::=\; \mathbf{0} \mid a \mid f_1 + f_2 \mid f_1 \cdot f_2 \mid f_1{}^{\circledast} f_2 \qquad (\text{for } a \in A).$$

Regular Expressions (1-free)

Definition (~ *Kleene, 1951*, ~ *Copi–Elgot–Wright, 1958*)

Regular expressions over alphabet A with unary / binary Kleene star:

$$e, e_1, e_2 \;::=\; \mathbf{0} \mid \mathbf{1} \mid \boldsymbol{a} \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^{*} \qquad (\text{for } \boldsymbol{a} \in A).$$

$$e, e_1, e_2 \;::=\; \mathbf{0} \mid \mathbf{1} \mid \boldsymbol{a} \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e_1{}^{\circledast} e_2 \qquad (\text{for } \boldsymbol{a} \in A).$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$
- with unary star *: $\mathbf{1}$ is definable as $\mathbf{0}^{*}$
- with binary star $^{\circledast}$: $\mathbf{1}$ is not definable (in its absence)

Definition (for process interpretation)

1-free regular expressions over alphabet A with unary/binary Kleene star:

$$f, f_1, f_2 \;::=\; \mathbf{0} \mid \boldsymbol{a} \mid f_1 + f_2 \mid f_1 \cdot f_2 \mid (f_1{}^{*}) \cdot f_2 \qquad (\text{for } \boldsymbol{a} \in A),$$

$$f, f_1, f_2 \;::=\; \mathbf{0} \mid \boldsymbol{a} \mid f_1 + f_2 \mid f_1 \cdot f_2 \mid f_1{}^{\circledast} f_2 \qquad (\text{for } \boldsymbol{a} \in A).$$

Regular Expressions (under-star-/1-free)

Definition (~ *Kleene, 1951*, ~ *Copi–Elgot–Wright, 1958*)

Regular expressions over alphabet A with unary / binary Kleene star:

$$e, e_1, e_2 \ ::= \ \mathbf{0} \mid \mathbf{1} \mid \boldsymbol{a} \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^{*} \qquad (\text{for } \boldsymbol{a} \in A).$$

$$e, e_1, e_2 \ ::= \ \mathbf{0} \mid \mathbf{1} \mid \boldsymbol{a} \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e_1{}^{\circledast} e_2 \qquad (\text{for } \boldsymbol{a} \in A).$$

- symbol $\mathbf{0}$ instead of $\varnothing$, symbol $\mathbf{1}$ instead of $\{\epsilon\}$
- with unary star *: $\mathbf{1}$ is definable as $\mathbf{0}^{*}$
- with binary star $^{\circledast}$: $\mathbf{1}$ is not definable (in its absence)

Definition (for process interpretation)

The set $\mathit{RExp}^{(\cancel{1})}(A)$ of 1-free regular expressions over A is defined by:

$$f, f_1, f_2 \ ::= \ \mathbf{0} \mid \boldsymbol{a} \mid f_1 + f_2 \mid f_1 \cdot f_2 \mid f_1{}^{\star} \cdot f_2 \qquad (\text{for } \boldsymbol{a} \in A),$$

the set $\mathit{RExp}^{(*/\cancel{1})}(A)$ of under-star-1-free regular expressions over A by:

$$uf, uf_1, uf_2 \ ::= \ \mathbf{0} \mid \mathbf{1} \mid \boldsymbol{a} \mid uf_1 + uf_2 \mid uf_1 \cdot uf_2 \mid f^{\star} \qquad (\text{for } \boldsymbol{a} \in A).$$

Process interpretation P of regular expressions *(Milner, 1984)*

$\mathbf{0} \xmapsto{P}$ deadlock $\boldsymbol{\delta}$, no termination

$\mathbf{1} \xmapsto{P}$ empty-step process $\boldsymbol{\epsilon}$, then terminate

$a \xmapsto{P}$ atomic action a, then terminate

Process interpretation P of regular expressions *(Milner, 1984)*

$\mathbf{0} \xmapsto{P}$ deadlock δ, no termination

$\mathbf{1} \xmapsto{P}$ empty-step process ϵ, then terminate

$a \xmapsto{P}$ atomic action a, then terminate

$e_1 + e_2 \xmapsto{P}$ (*choice*) execute $P(e_1)$ or $P(e_2)$

$e_1 \cdot e_2 \xmapsto{P}$ (*sequentialization*) execute $P(e_1)$, then $P(e_2)$

$e^* \xmapsto{P}$ (*iteration*) repeat (terminate or execute $P(e)$)

Process interpretation P of regular expressions *(Milner, 1984)*

$\mathbf{0} \xrightarrow{P}$ deadlock δ, no termination

$\mathbf{1} \xrightarrow{P}$ empty-step process ϵ, then terminate

$a \xrightarrow{P}$ atomic action a, then terminate

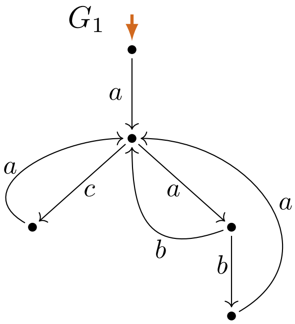
$e_1 + e_2 \xrightarrow{P}$ (*choice*) execute $P(e_1)$ or $P(e_2)$

$e_1 \cdot e_2 \xrightarrow{P}$ (*sequentialization*) execute $P(e_1)$, then $P(e_2)$

$e^* \xrightarrow{P}$ (*iteration*) repeat (terminate or execute $P(e)$)

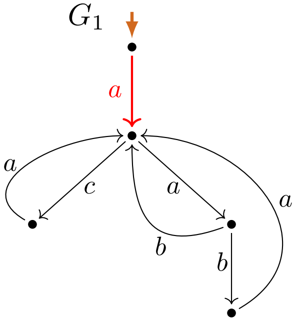
$[\![e]\!]_P := [P(e)]_{\underline{\leftrightarrow}}$ (bisimilarity equivalence class of process $P(e)$)

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



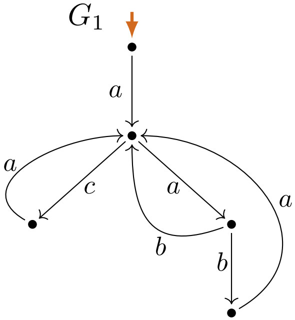
$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^*)\cdot 0}^{f}\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



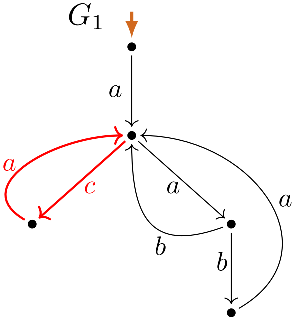
$$P\Big(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^{f} \Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



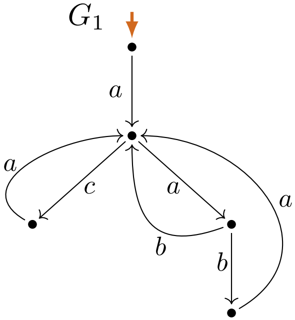
$$P\big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^*)\cdot 0}^{f}\big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



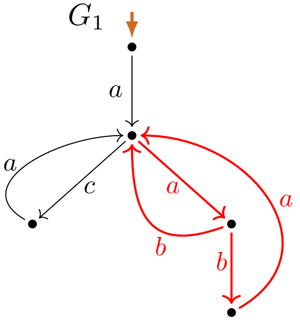
$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b+b\cdot a))^*)\cdot 0}^{f}\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



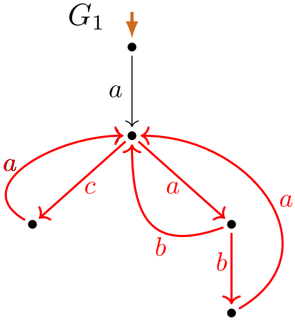
$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^{*})\cdot 0}^{f}\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



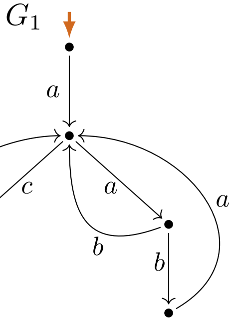
$$P\Big(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{*}) \cdot 0}^{f} \Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



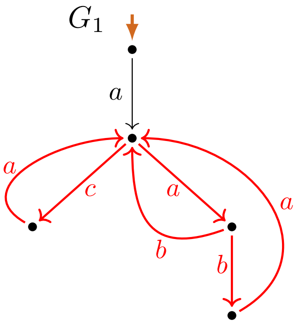
$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^*)\cdot 0}^{f}\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



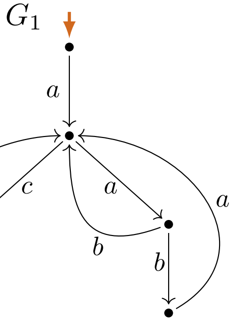
$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{*})\cdot 0}^{f}\Big)$$
$$P\Big(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{\circledast}0\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{*})\cdot 0}^{f}\Big)$$
$$P\Big(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{\circledast}0\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)

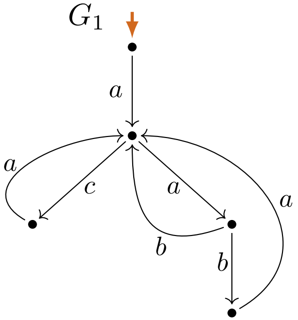


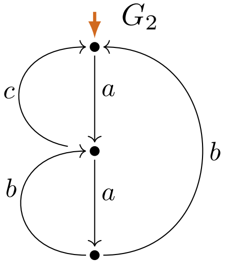
$$P\Big(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^{f} \Big)$$

$$P\Big(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\circledast} 0 \Big)$$

$$G_1 \in [\![f]\!]_P$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



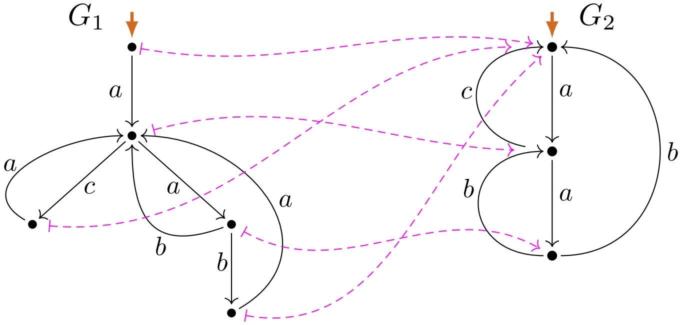


$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^*)\cdot 0}^{f}\Big)$$

$$P\Big(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{\circledast}0\Big)$$

$$G_1 \in [\![f]\!]_P$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)

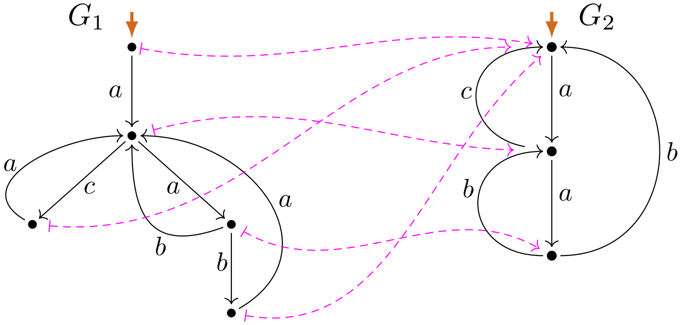


$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{*})\cdot 0}^{f}\Big)$$

$$P\Big(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{\circledast}0\Big)$$

$$G_1 \in [\![f]\!]_P$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^*)\cdot 0}^{f}\Big)$$

$$P\Big(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^{\circledast}0\Big)$$

$$G_1 \in [\![f]\!]_P$$

$$G_2 \in [\![f]\!]_P$$

Process interpretation P (formally)

Definition (Transition system specification $\mathcal{T}$)

$$\frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} e_i'}{e_1 + e_2 \xrightarrow{a} e_i'} \, (i \in \{1,2\})$$

Process interpretation P (formally)

Definition (Transition system specification $\mathcal{T}$)

$$\frac{}{a \xrightarrow{a} \mathbf{1}} \qquad \frac{e_i \xrightarrow{a} e_i'}{e_1 + e_2 \xrightarrow{a} e_i'} \; (i \in \{1,2\})$$

$$\frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Process interpretation P (formally)

Definition (Transition system specification $\mathcal{T}$)

$$\frac{}{\mathbf{1}{\Downarrow}} \qquad \frac{e_i{\Downarrow}}{(e_1 + e_2){\Downarrow}} \; (i \in \{1,2\}) \qquad \frac{e_1{\Downarrow} \qquad e_2{\Downarrow}}{(e_1 \cdot e_2){\Downarrow}} \qquad \frac{}{(e^*){\Downarrow}}$$

$$\frac{}{a \xrightarrow{a} \mathbf{1}} \qquad \frac{e_i \xrightarrow{a} e_i'}{e_1 + e_2 \xrightarrow{a} e_i'} \; (i \in \{1,2\})$$

$$\frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Process interpretation P (formally)

Definition (Transition system specification $\mathcal{T}$)

$$\frac{}{\mathbf{1}{\Downarrow}} \qquad \frac{e_i{\Downarrow}}{(e_1 + e_2){\Downarrow}} \; (i \in \{1,2\}) \qquad \frac{e_1{\Downarrow} \quad e_2{\Downarrow}}{(e_1 \cdot e_2){\Downarrow}} \qquad \frac{}{(e^*){\Downarrow}}$$

$$\frac{}{a \xrightarrow{a} \mathbf{1}} \qquad \frac{e_i \xrightarrow{a} e_i'}{e_1 + e_2 \xrightarrow{a} e_i'} \; (i \in \{1,2\})$$

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \qquad \frac{e_1{\Downarrow} \quad e_2 \xrightarrow{a} e_2'}{e_1 \cdot e_2 \xrightarrow{a} e_2'} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Process interpretation P (formally)

Definition (Transition system specification $\mathcal{T}$)

$$\frac{}{1{\Downarrow}} \qquad \frac{e_i{\Downarrow}}{(e_1 + e_2){\Downarrow}}\ (i \in \{1,2\}) \qquad \frac{e_1{\Downarrow} \quad e_2{\Downarrow}}{(e_1 \cdot e_2){\Downarrow}} \qquad \frac{}{(e^*){\Downarrow}}$$

$$\frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} e_i'}{e_1 + e_2 \xrightarrow{a} e_i'}\ (i \in \{1,2\})$$

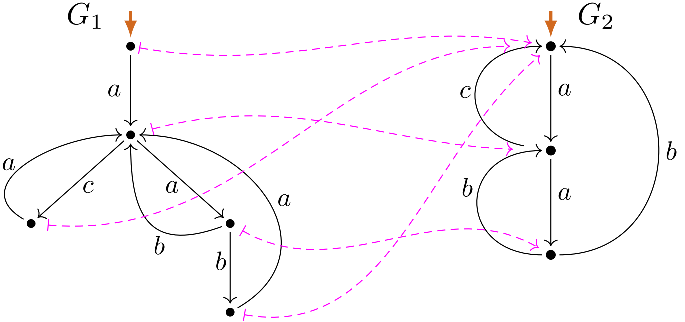
$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \qquad \frac{e_1{\Downarrow} \quad e_2 \xrightarrow{a} e_2'}{e_1 \cdot e_2 \xrightarrow{a} e_2'} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Definition

The process (graph) interpretation $P(e)$ of a regular expression e:

$P(e)$:= labeled transition graph generated by e by derivations in $\mathcal{T}$.

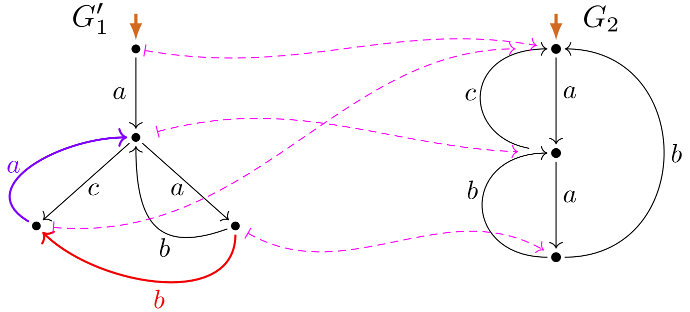
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



$$P\Big(\overbrace{a\cdot((c\cdot a+a\cdot(b+b\cdot a)^*)\cdot 0)}^{f}\Big)$$

$G_1 \in [\![f]\!]_P$ $G_2 \in [\![f]\!]_P$

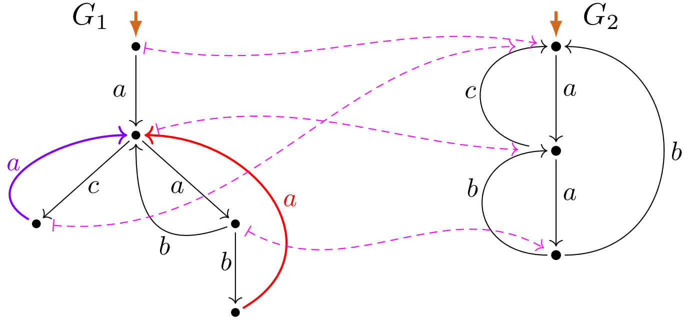
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, formally)



$$P\Big(\overbrace{a \cdot ((c \cdot a + a \cdot (b + b \cdot a)^*) \cdot 0)}^{f} \Big)$$

$G'_1 \in [\![f]\!]_P$ $\qquad$ $G_2 \in [\![f]\!]_P$

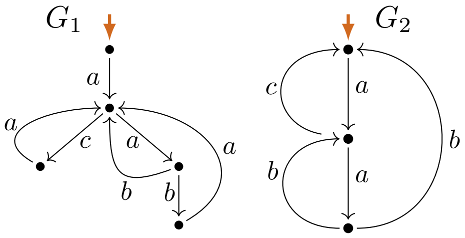
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, formally)



$$P\Big(\overbrace{a \cdot ((c \cdot a + a \cdot (b + b \cdot (a + a))^*) \cdot 0)}^{f}\Big)$$

$G_1 \in [\![f]\!]_P$ $G_2 \in [\![f]\!]_P$

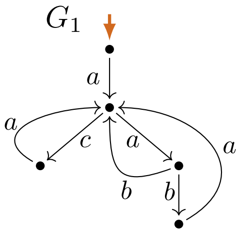
P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples)

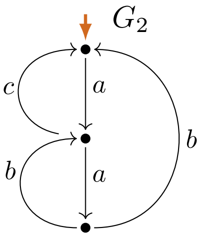


P-expressible

$[\![\cdot]\!]_P$-expressible $[\![\cdot]\!]_P$-expressible

P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples)

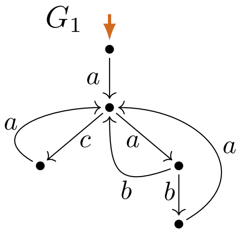


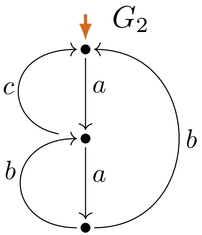


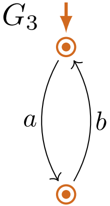
P-expressible ?

$[\![\cdot]\!]_P$-expressible $[\![\cdot]\!]_P$-expressible

P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples)

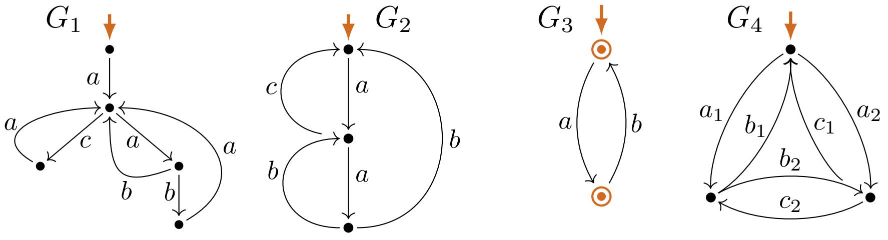






G_1	G_2	G_3
P-expressible	**?**	**not** P-expressible
$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	**not** $[\![\cdot]\!]_P$-expressible

P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples)

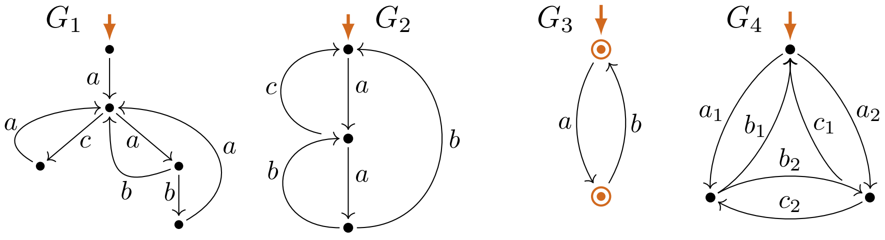


P-expressible
$[\![\cdot]\!]_P$-expressible

?
$[\![\cdot]\!]_P$-expressible

not P-expressible
not $[\![\cdot]\!]_P$-expressible

P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples)



G_1	G_2	G_3, G_4
P-expressible	**?**	**not** P-expressible
$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	**not** $[\![\cdot]\!]_P$-expressible

Q: How can P-expressibility and $[\![\cdot]\!]_P$-expressibility be characterized?

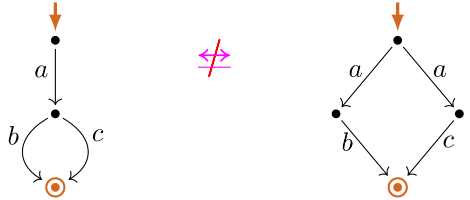
Properties of P, $[\![\cdot]\!]_P$, and $=_{[\![\cdot]\!]_P}$

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

$P(a \cdot (b + c))$ $\quad$ $P(a \cdot b + a \cdot c)$

Properties of P, $[\![\cdot]\!]_P$, and $=_{[\![\cdot]\!]_P}$

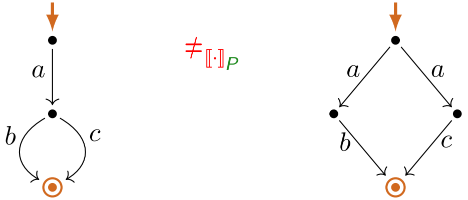
- Not every finite-state process is P-expressible.
- Not every finite-state process is $[\![\cdot]\!]_P$-expressible
  (= P-expressible modulo $\underline{\leftrightarrow}$).
- Fewer identities hold for $=_{[\![\cdot]\!]_P}$ than for $=_{[\![\cdot]\!]_L}$: .



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

Properties of P, $[\![\cdot]\!]_P$, and $=_{[\![\cdot]\!]_P}$

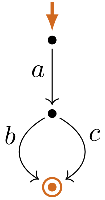
- Not every finite-state process is P-expressible.
- Not every finite-state process is $[\![\cdot]\!]_P$-expressible
  (= P-expressible modulo $\underline{\leftrightarrow}$).
- Fewer identities hold for $=_{[\![\cdot]\!]_P}$ than for $=_{[\![\cdot]\!]_L}$: .



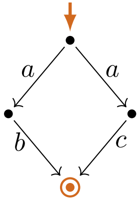
$$a \cdot (b + c) \quad \neq_{[\![\cdot]\!]_P} \quad a \cdot b + a \cdot c$$

Properties of P, $[\![\cdot]\!]_P$, and $=_{[\![\cdot]\!]_P}$

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



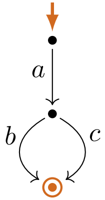
$=_{[\![\cdot]\!]_L}$



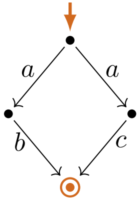
$a \cdot (b + c) \quad =_{[\![\cdot]\!]_L} \quad a \cdot b + a \cdot c$

Properties of P, $[\![\cdot]\!]_P$, and $=_{[\![\cdot]\!]_P}$

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



$\neq_{[\![\cdot]\!]_P}$



$$a \cdot (b + c) \quad \neq_{[\![\cdot]\!]_P} \quad a \cdot b + a \cdot c$$

Milner's questions (1984)

(Q1) Complete axiomatization:

Is the axiom system suggested by Milner complete for $=_{[\![\cdot]\!]_P}$*?*

(Q2) $[\![\cdot]\!]_P$-Expressibility:

What structural property characterizes process graphs that are $[\![\cdot]\!]_P$*-expressible?*

Milner's questions (1984)

(Q1) Complete axiomatization:

Is the axiom system suggested by Milner complete for $=_{[\![\cdot]\!]_P}$?

(Q2) $[\![\cdot]\!]_P$-Expressibility:

What structural property characterizes process graphs that are $[\![\cdot]\!]_P$-expressible?

- is decidable (Baeten/Corradini/G, 2007)

Milner's questions (1984)

(Q1) Complete axiomatization:
Is the axiom system suggested by Milner complete for $=_{[\![\cdot]\!]_P}$?

(Q2) $[\![\cdot]\!]_P$-Expressibility:
What structural property characterizes process graphs that are $[\![\cdot]\!]_P$-expressible?

- is decidable (Baeten/Corradini/G, 2007)
- partial new answer (G/Fokkink, 2020):
  - bisimulation collapse has loop existence & elimination property (LEE) if expressible by under-star-1-free regular expression

Milner's questions (1984)

(Q1) Complete axiomatization:

Is the axiom system suggested by Milner complete for $=_{[\![\cdot]\!]_P}$?

- series of partial completeness results for:
  - exitless iterations (Fokkink, 1998)
  - with a stronger fixed-point rule (G, 2006)
  - under-star 1-free, and without 0 (Corradini/de Nicola/Labella, 2004)
  - with 0 but under-star-1-free (G/Fokkink, 2020)

(Q2) $[\![\cdot]\!]_P$-Expressibility:

What structural property characterizes process graphs that are $[\![\cdot]\!]_P$-expressible?

- is decidable (Baeten/Corradini/G, 2007)
- partial new answer (G/Fokkink, 2020):
  - bisimulation collapse has loop existence & elimination property (LEE) if expressible by under-star-1-free regular expression

Milner's questions (1984)

(Q1) Complete axiomatization:

Is the axiom system suggested by Milner complete for $=_{[\![\cdot]\!]_P}$?

- Yes! (G, 2022, proof summary, employing LEE and crystallization)
- series of partial completeness results for:
  - exitless iterations (Fokkink, 1998)
  - with a stronger fixed-point rule (G, 2006)
  - under-star 1-free, and without 0 (Corradini/de Nicola/Labella, 2004)
  - with 0 but under-star-1-free (G/Fokkink, 2020)

(Q2) $[\![\cdot]\!]_P$-Expressibility:

What structural property characterizes process graphs that are $[\![\cdot]\!]_P$-expressible?

- is decidable (Baeten/Corradini/G, 2007)
- partial new answer (G/Fokkink, 2020):
  - bisimulation collapse has loop existence & elimination property (LEE) if expressible by under-star-1-free regular expression

Loop graphs (interpretations of innermost iterations without 1)

Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.

Loop graphs (interpretations of innermost iterations without 1)

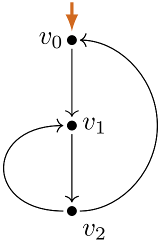
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



Loop graphs (interpretations of innermost iterations without 1)

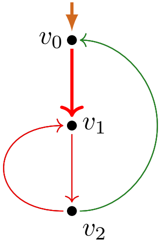
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1), ~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

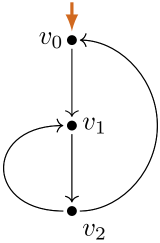
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1), ~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

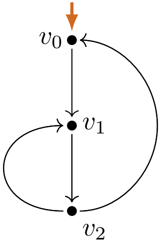
Definition

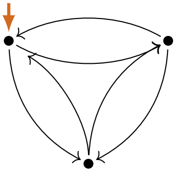
A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.





(L1), ~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

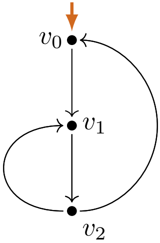
Definition

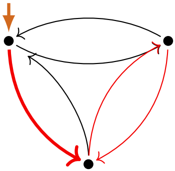
A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.





(L1), ~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

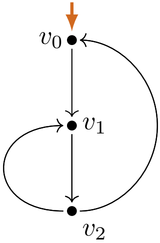
Definition

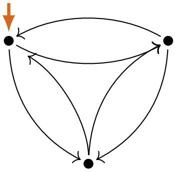
A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.





(L1), ~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

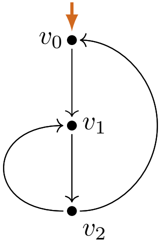
Definition

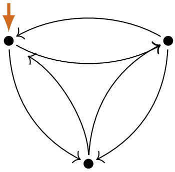
A process graph is a loop graph if:

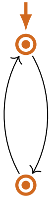
(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.







(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

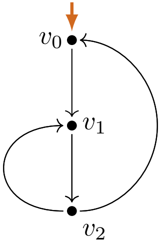
Definition

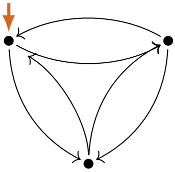
A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.





(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

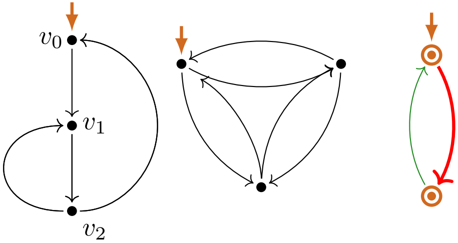
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~   (L1),(L2),~~(L3)~~

Loop graphs (interpretations of innermost iterations without 1)

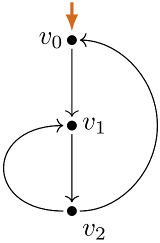
Definition

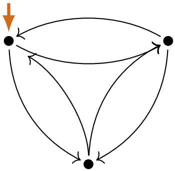
A process graph is a loop graph if:

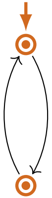
(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.







(L1),~~(L2)~~

(L1),(L2),~~(L3)~~

Loop graphs (interpretations of innermost iterations without 1)

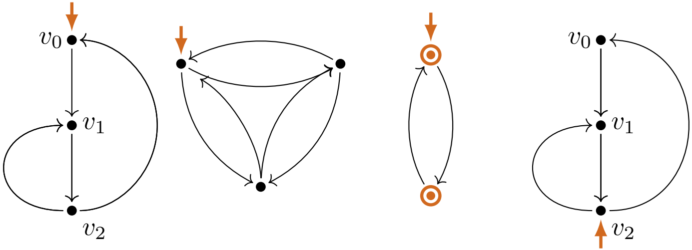
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~

Loop graphs (interpretations of innermost iterations without 1)

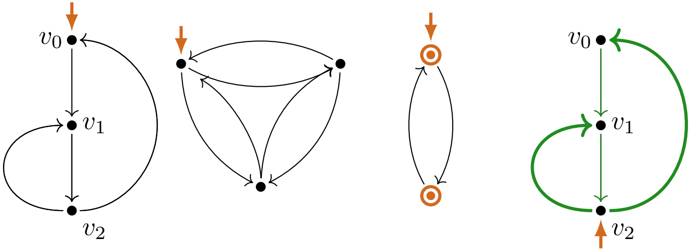
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~

Loop graphs (interpretations of innermost iterations without 1)

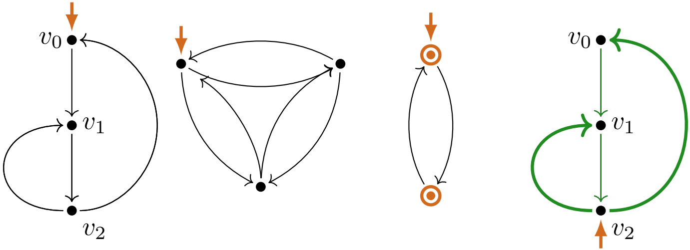
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart

Loop graphs (interpretations of innermost iterations without 1)

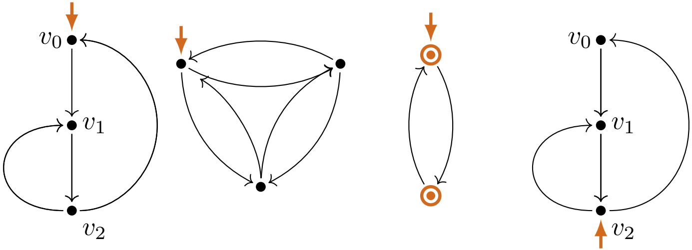
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart

Loop graphs (interpretations of innermost iterations without 1)

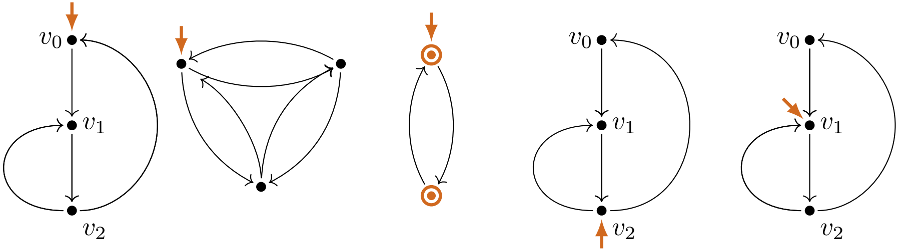
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart

Loop graphs (interpretations of innermost iterations without 1)

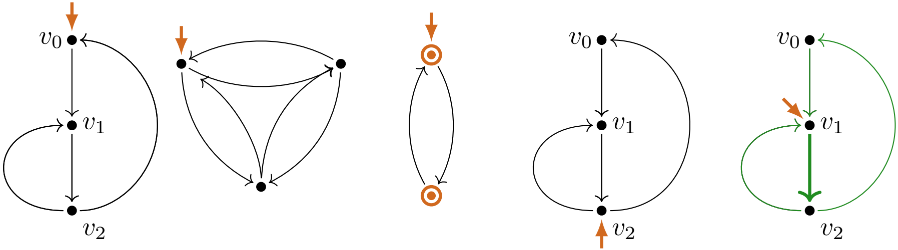
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart

Loop graphs (interpretations of innermost iterations without 1)

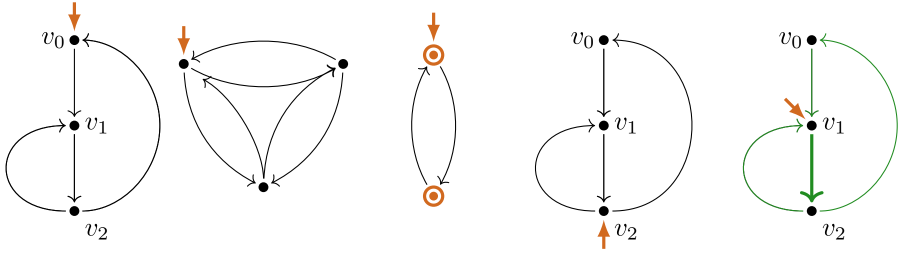
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart loop chart

Loop graphs (interpretations of innermost iterations without 1)

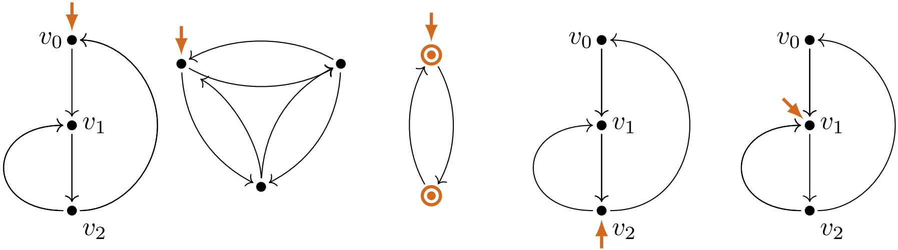
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.



(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart loop chart

Loop graphs (interpretations of innermost iterations without 1)

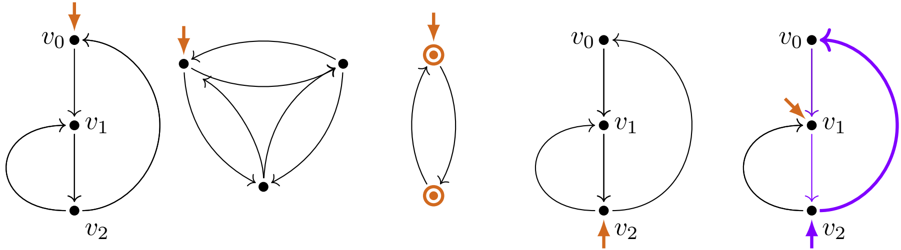
Definition

A process graph is a loop graph if:

(L1) There is an infinite path from the start vertex.

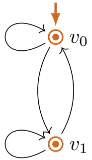
(L2) Every infinite path from the start vertex returns to it.

(L3) Termination is only possible at the start vertex.

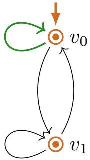


(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart loop subchart

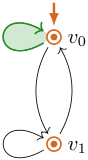
Loop elimination



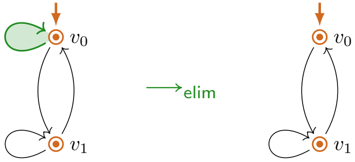
Loop elimination



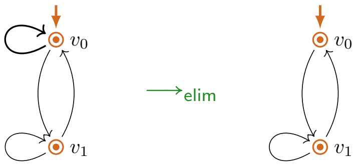
Loop elimination



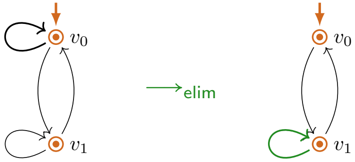
Loop elimination



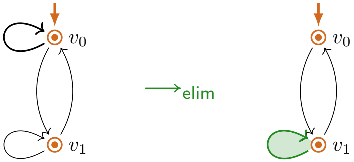
Loop elimination



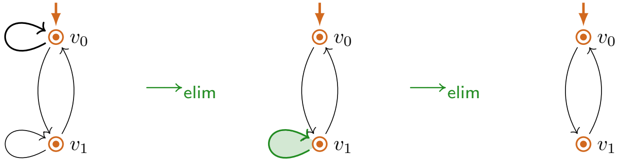
Loop elimination



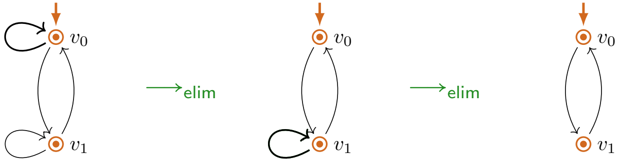
Loop elimination



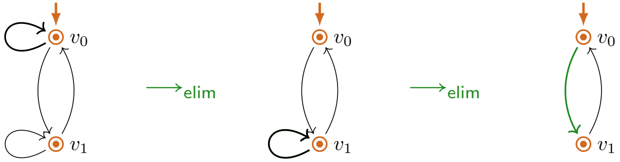
Loop elimination



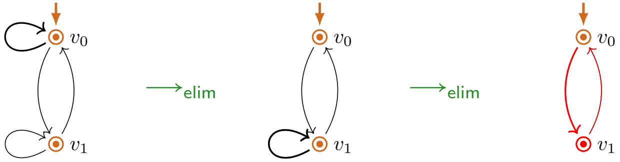
Loop elimination



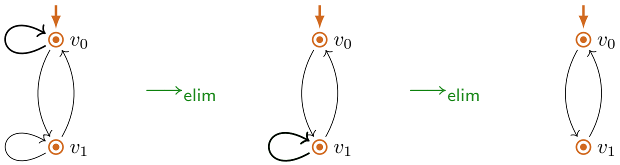
Loop elimination



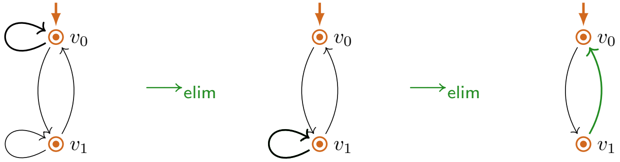
Loop elimination



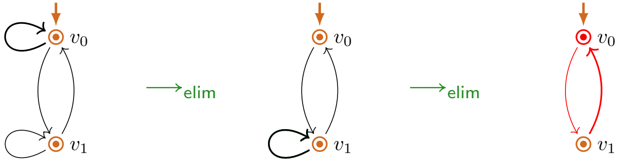
Loop elimination



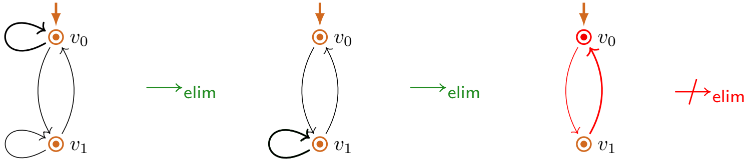
Loop elimination



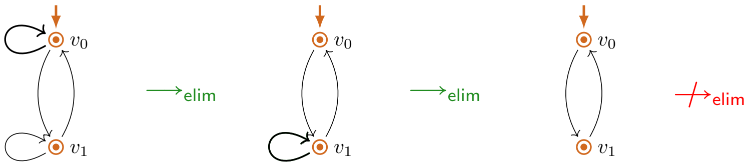
Loop elimination



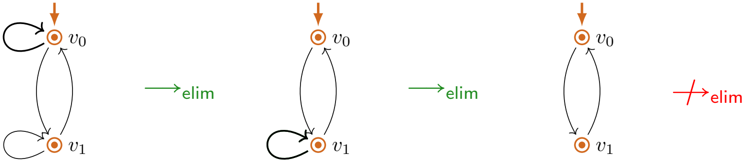
Loop elimination

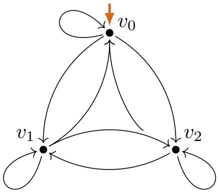


Loop elimination

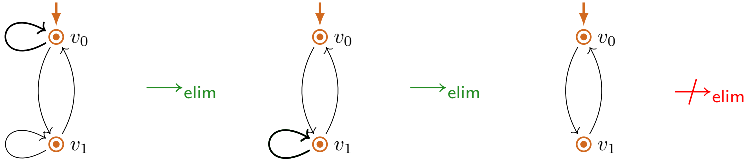


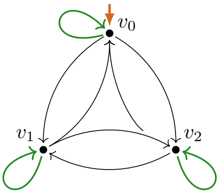
Loop elimination



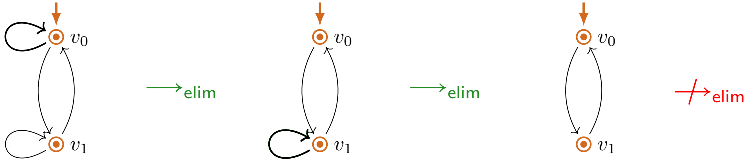


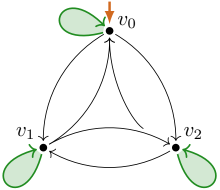
Loop elimination



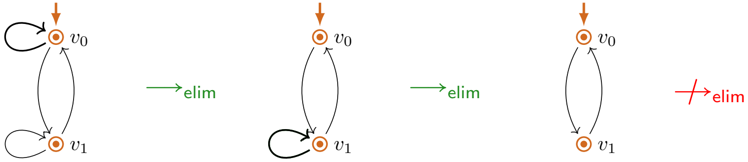


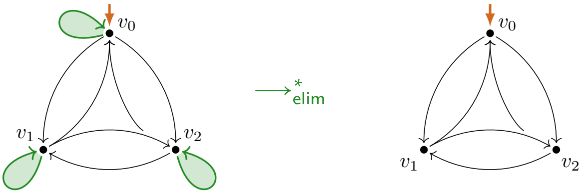
Loop elimination



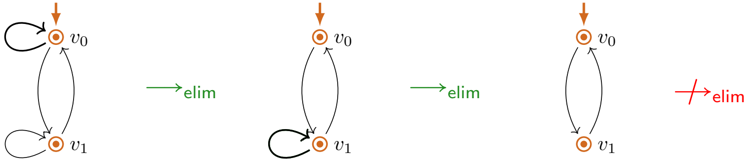


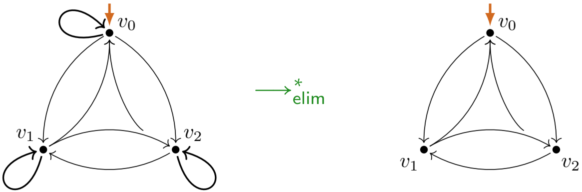
Loop elimination



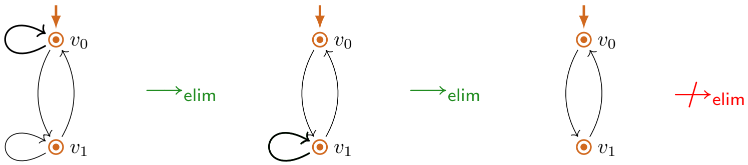


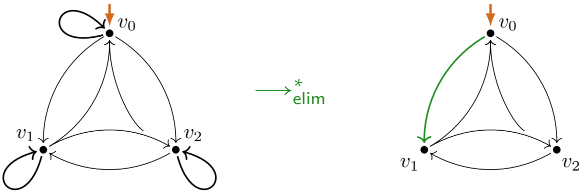
Loop elimination



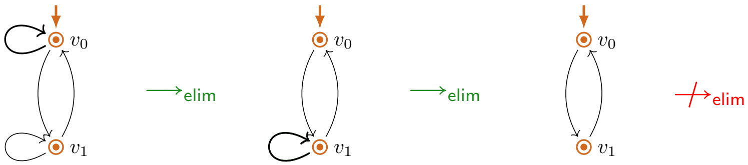


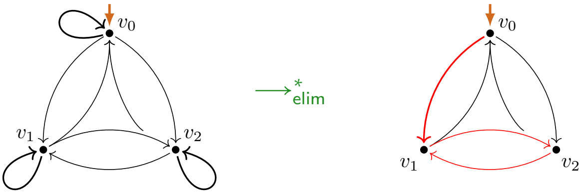
Loop elimination



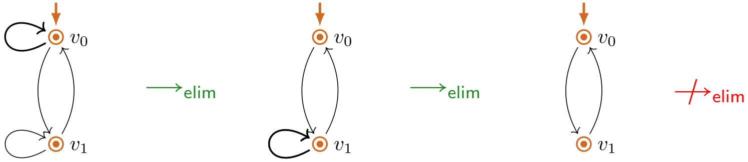


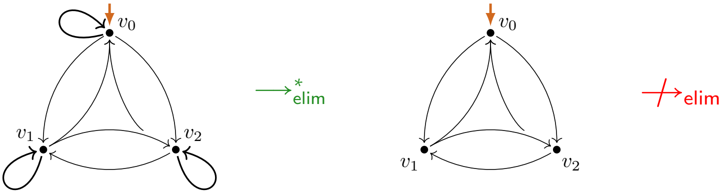
Loop elimination





Loop elimination





Loop elimination

$\xrightarrow{}_{\text{elim}}$: eliminate a transition-induced loop by:

- removing the loop-entry transition(s)
- garbage collection

$\xrightarrow{}_{\text{prune}}$: remove a transition to a deadlocking state

Lemma

(i) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is terminating.*

Loop elimination

$\longrightarrow_{\text{elim}}$: eliminate a transition-induced loop by:

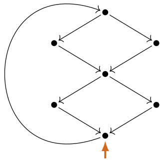
- removing the loop-entry transition(s)
- garbage collection

$\longrightarrow_{\text{prune}}$: remove a transition to a deadlocking state

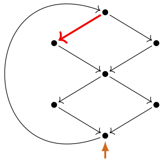
Lemma

(i) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is terminating.*

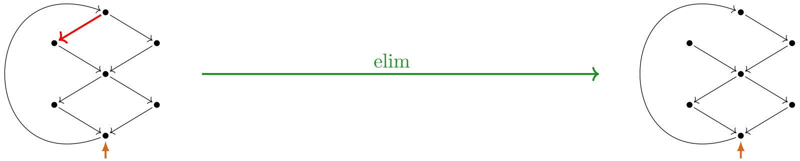
'Critical pair': bi-loop elimination



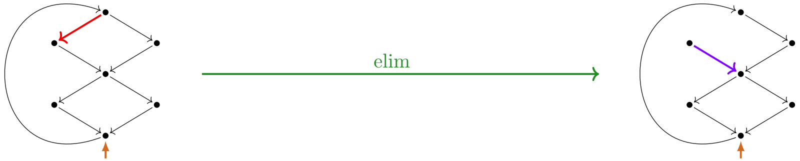
‘Critical pair’: bi-loop elimination



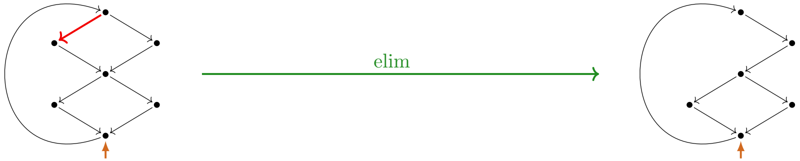
'Critical pair': bi-loop elimination



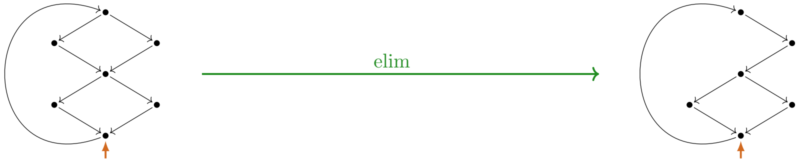
'Critical pair': bi-loop elimination



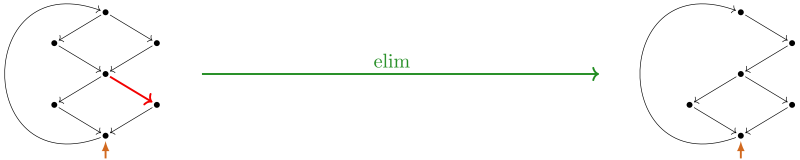
'Critical pair': bi-loop elimination



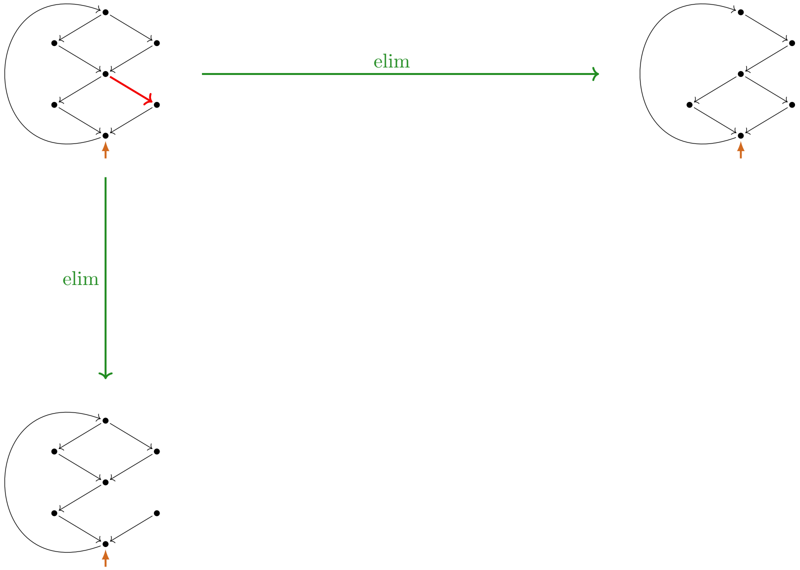
'Critical pair': bi-loop elimination



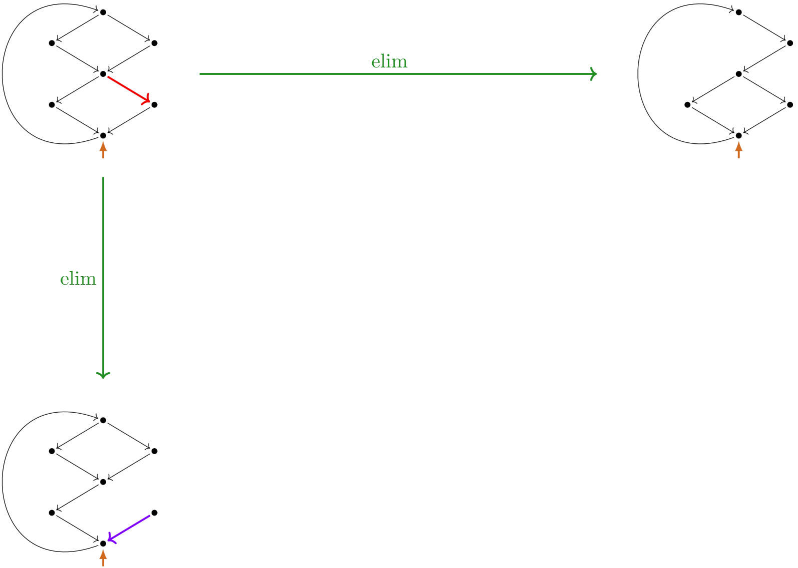
'Critical pair': bi-loop elimination



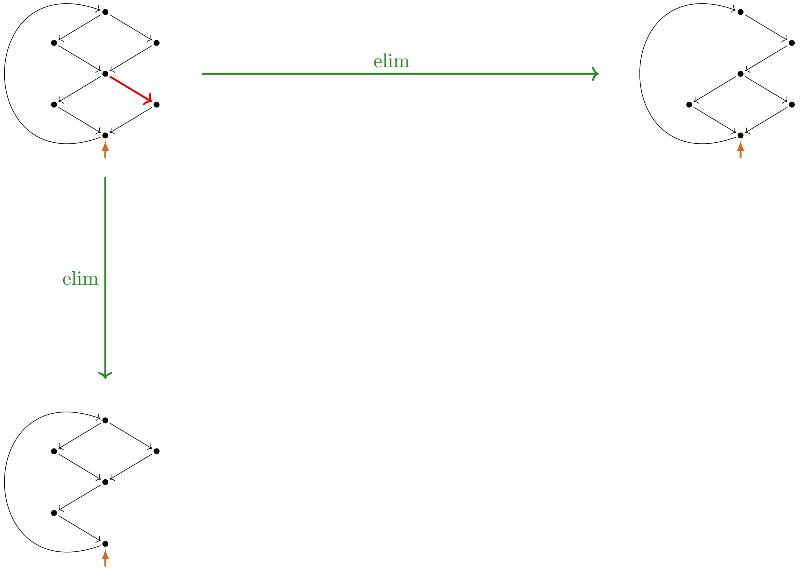
'Critical pair': bi-loop elimination



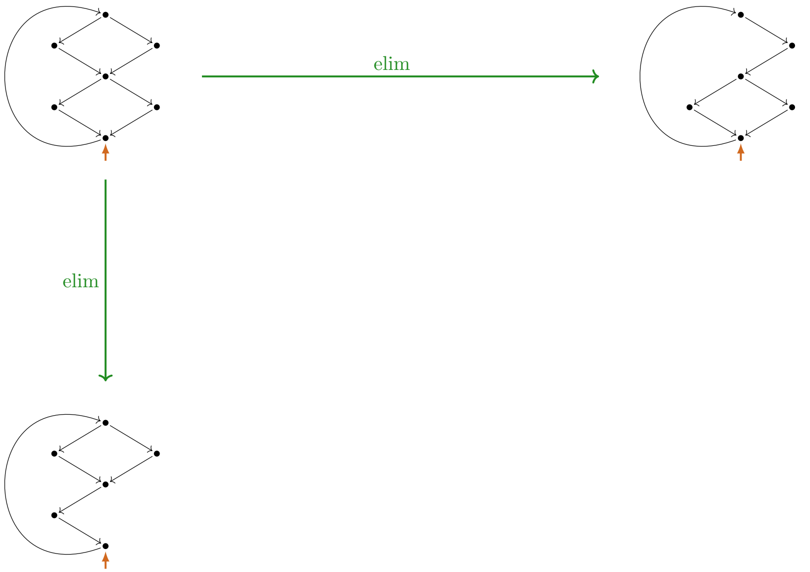
'Critical pair': bi-loop elimination



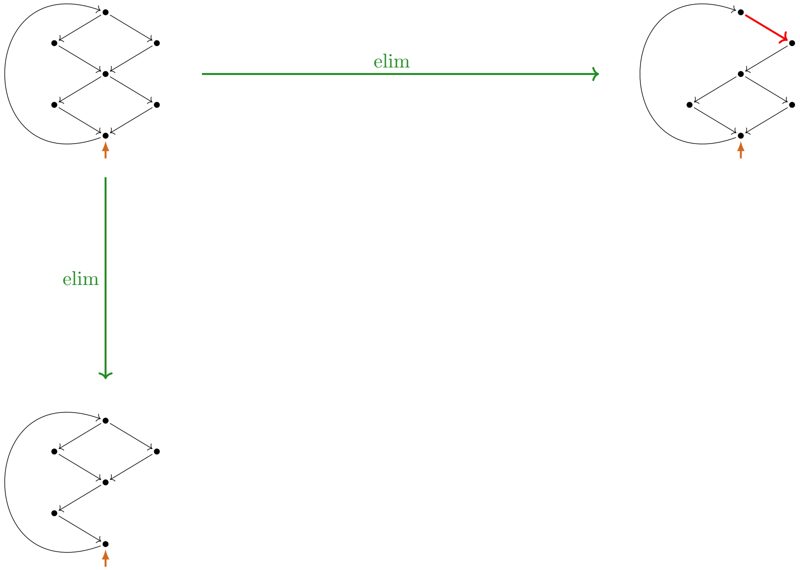
'Critical pair': bi-loop elimination



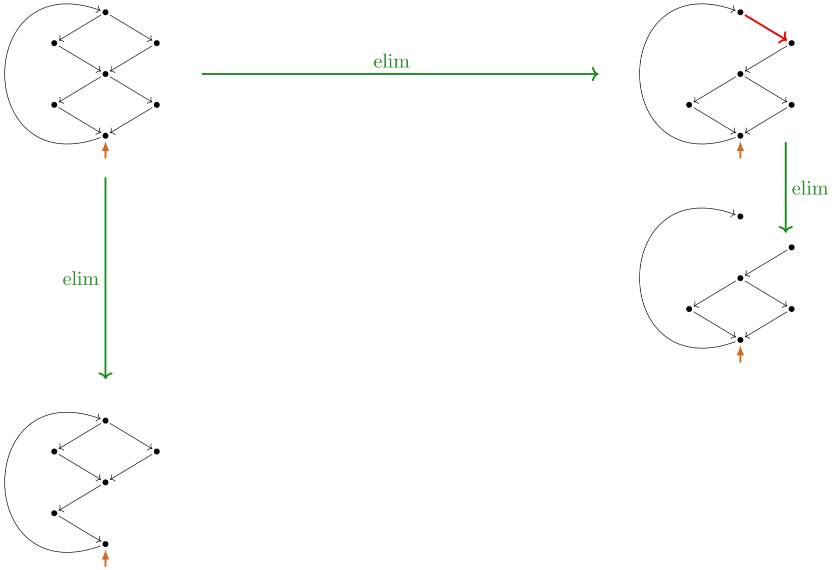
'Critical pair': bi-loop elimination



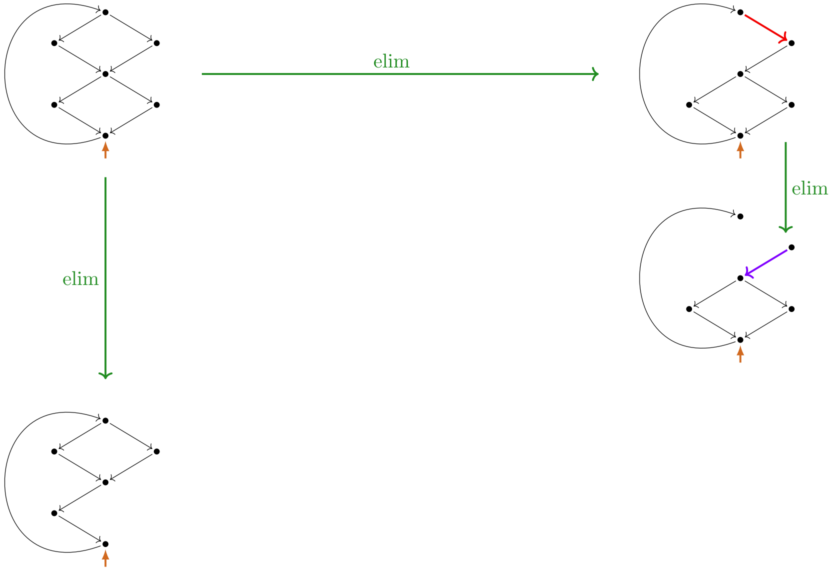
'Critical pair': bi-loop elimination



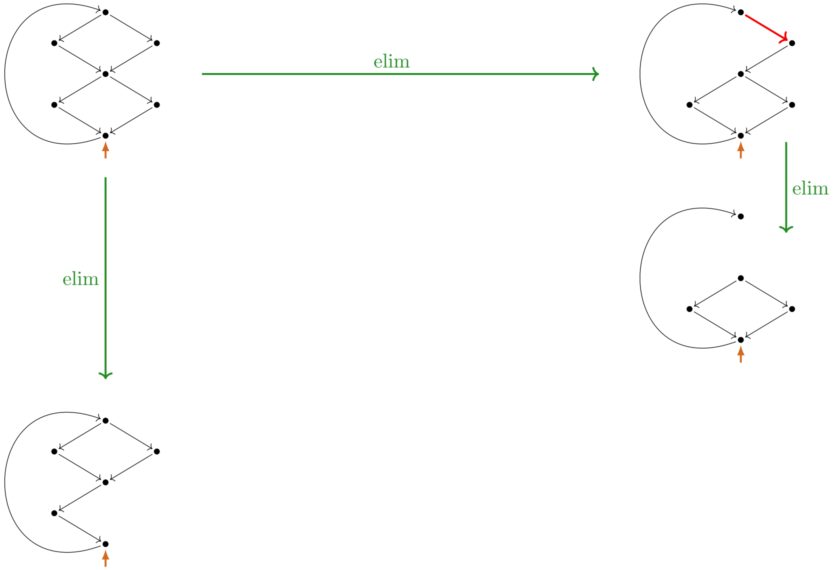
'Critical pair': bi-loop elimination



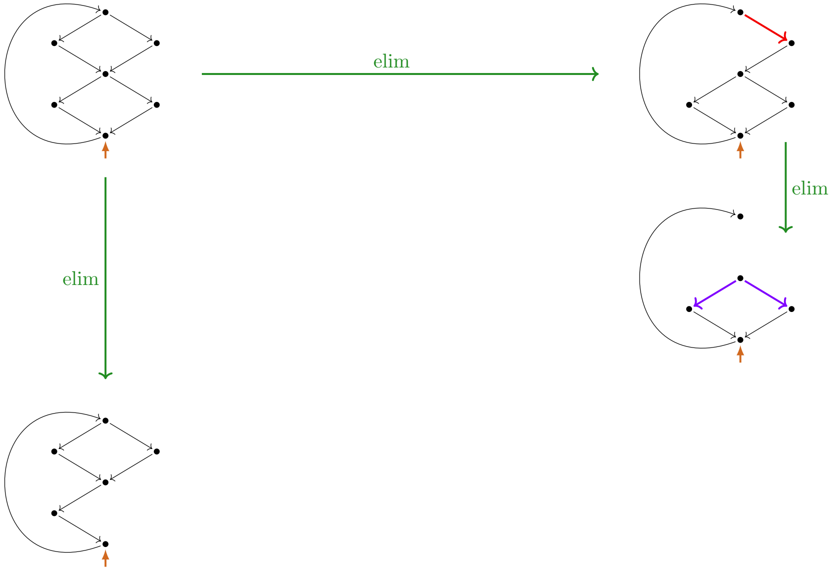
'Critical pair': bi-loop elimination



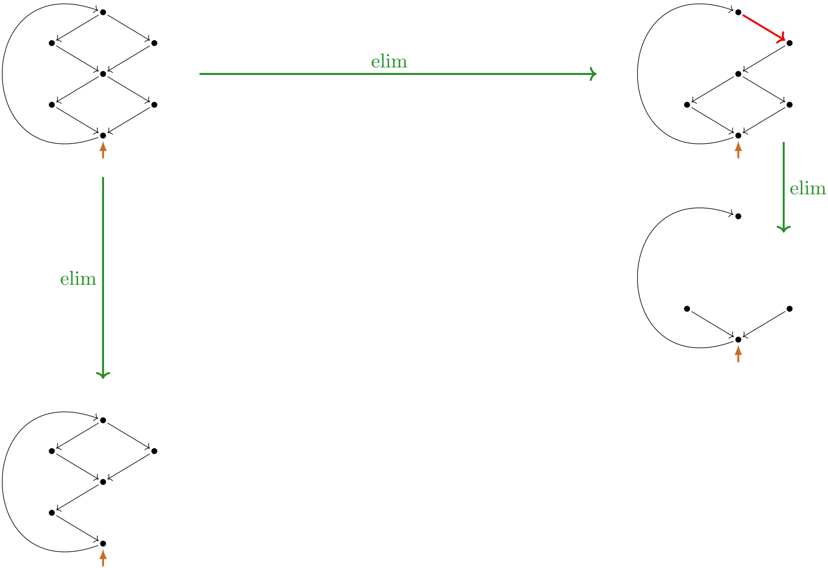
'Critical pair': bi-loop elimination



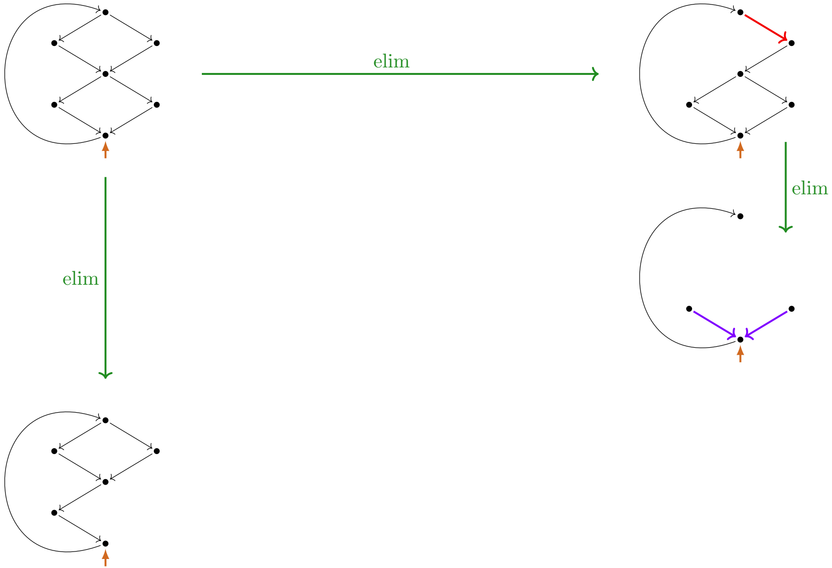
'Critical pair': bi-loop elimination



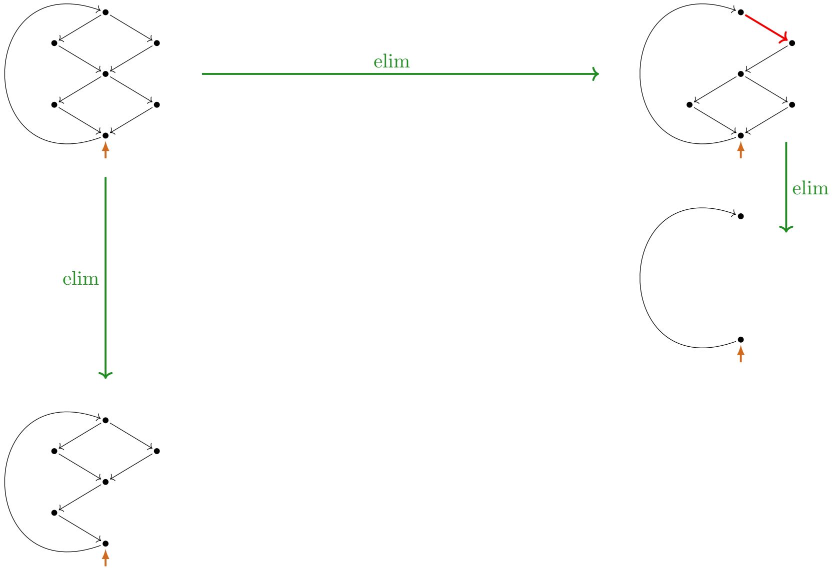
‘Critical pair’: bi-loop elimination



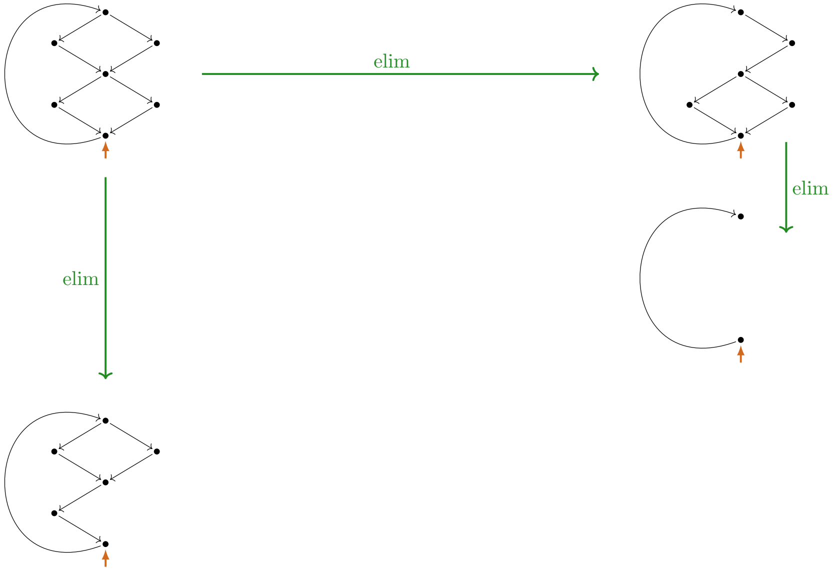
'Critical pair': bi-loop elimination



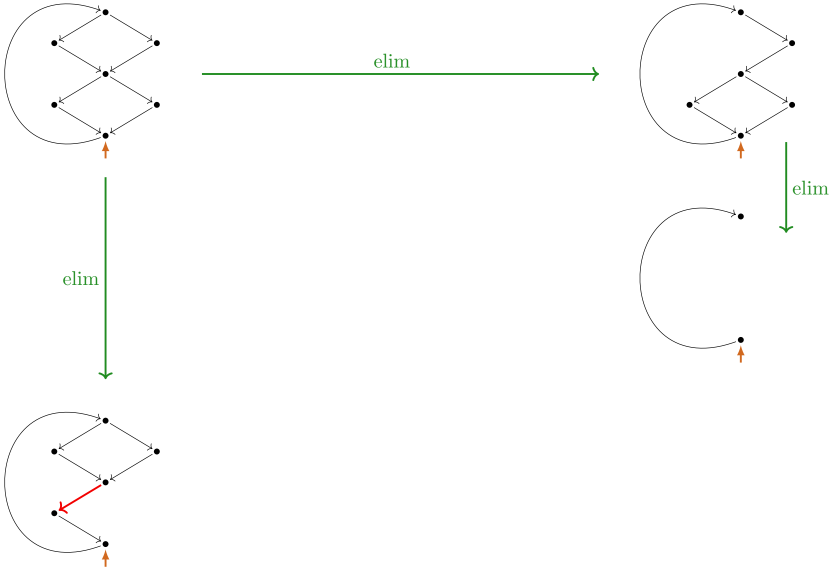
'Critical pair': bi-loop elimination



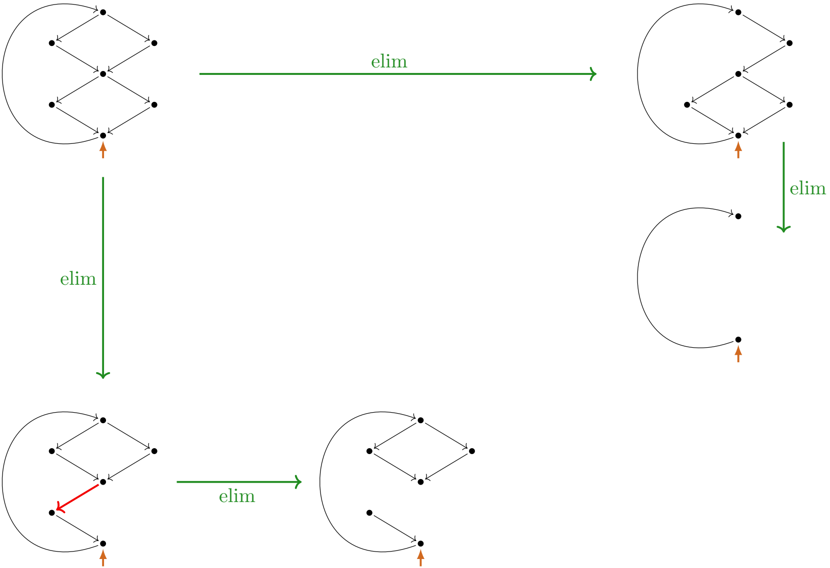
'Critical pair': bi-loop elimination



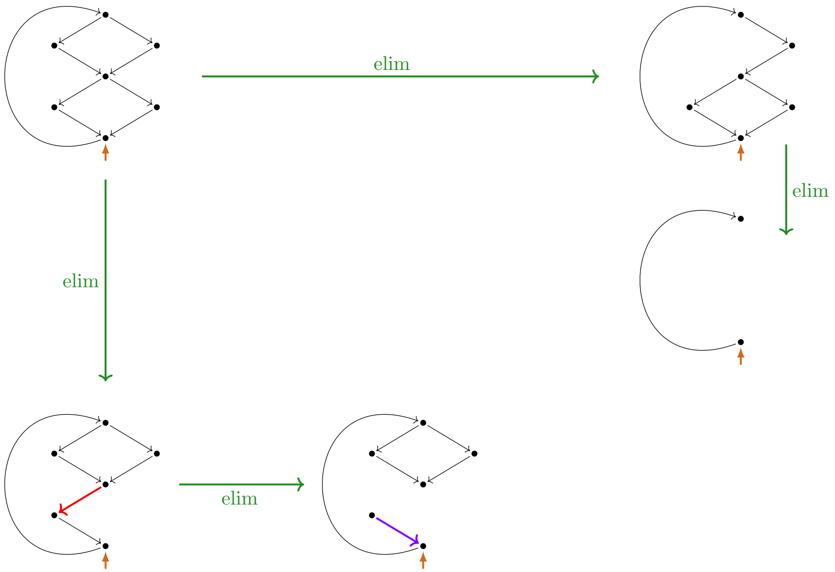
'Critical pair': bi-loop elimination



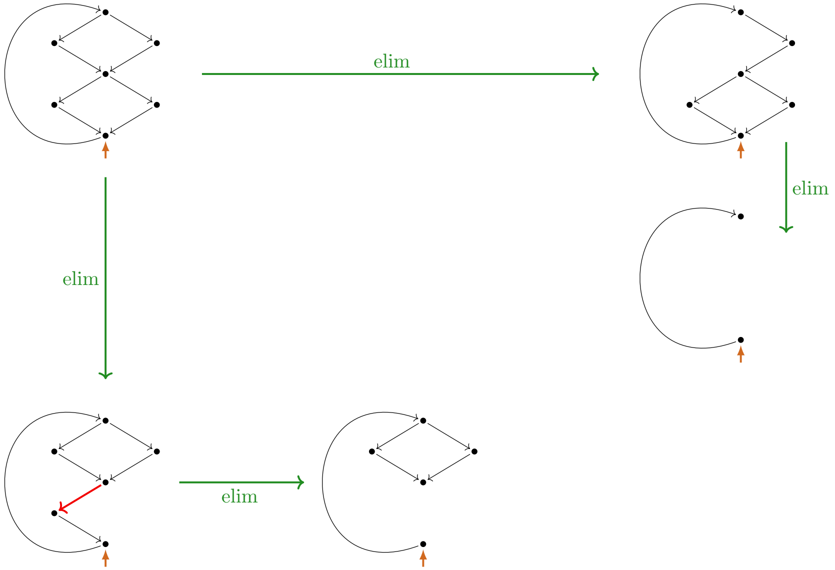
'Critical pair': bi-loop elimination



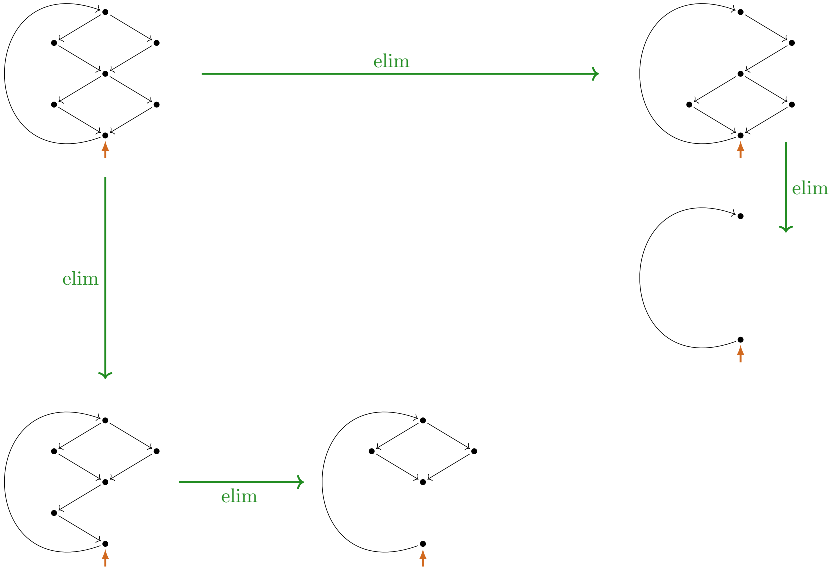
‘Critical pair’: bi-loop elimination



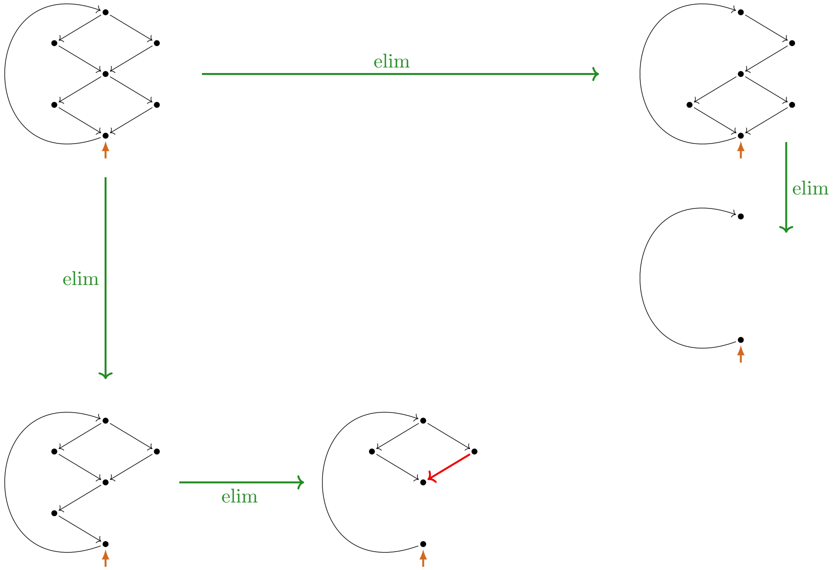
'Critical pair': bi-loop elimination



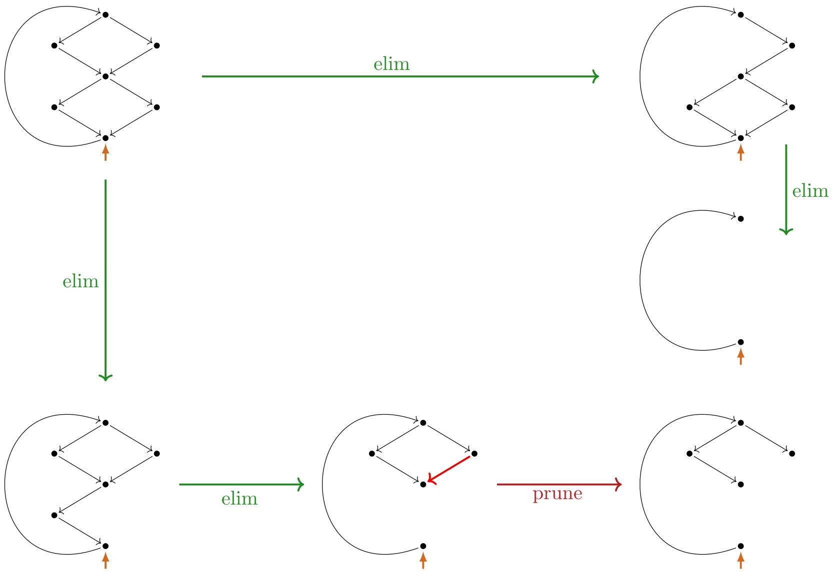
'Critical pair': bi-loop elimination



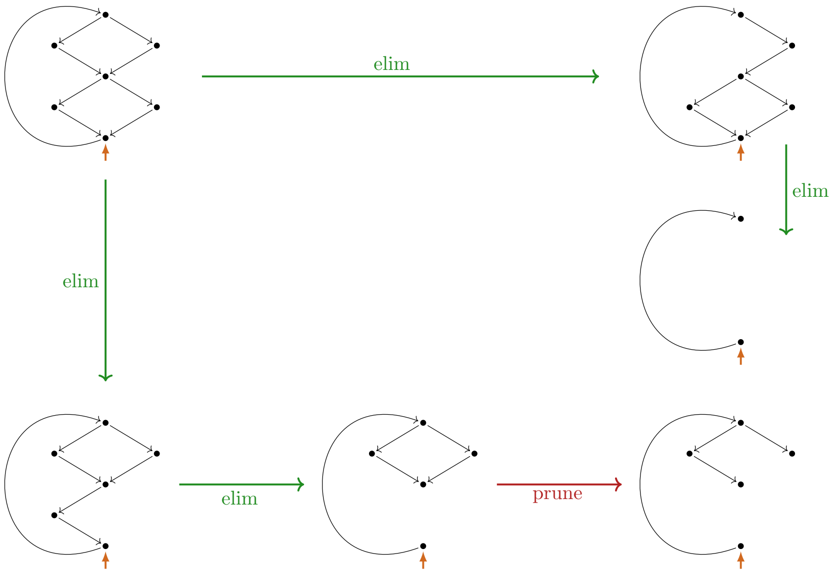
'Critical pair': bi-loop elimination



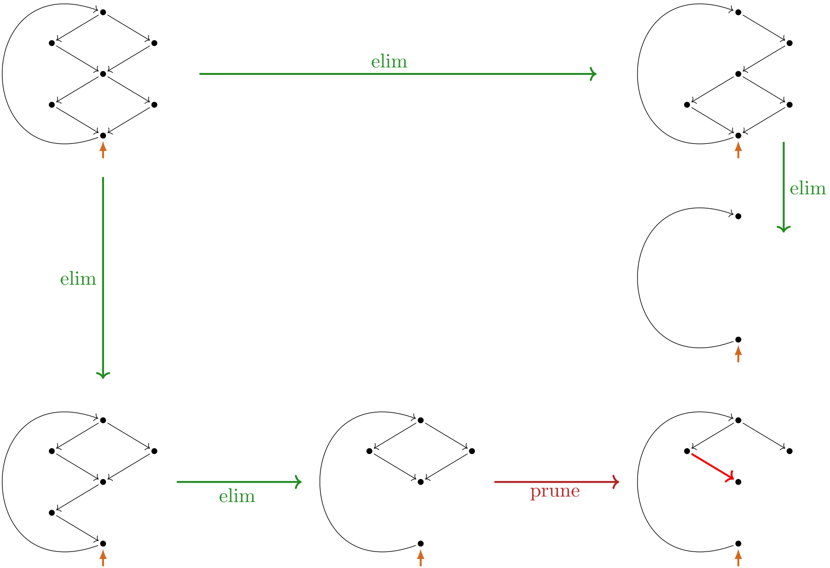
'Critical pair': bi-loop elimination



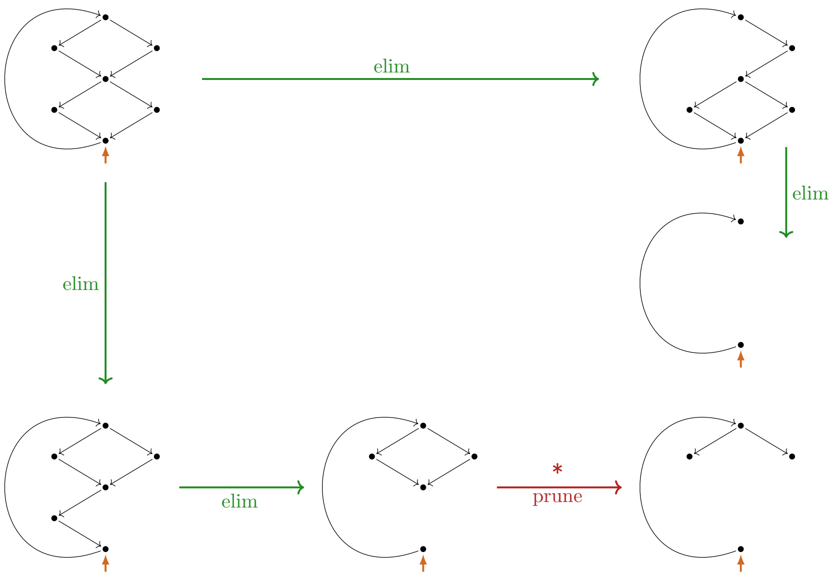
‘Critical pair’: bi-loop elimination



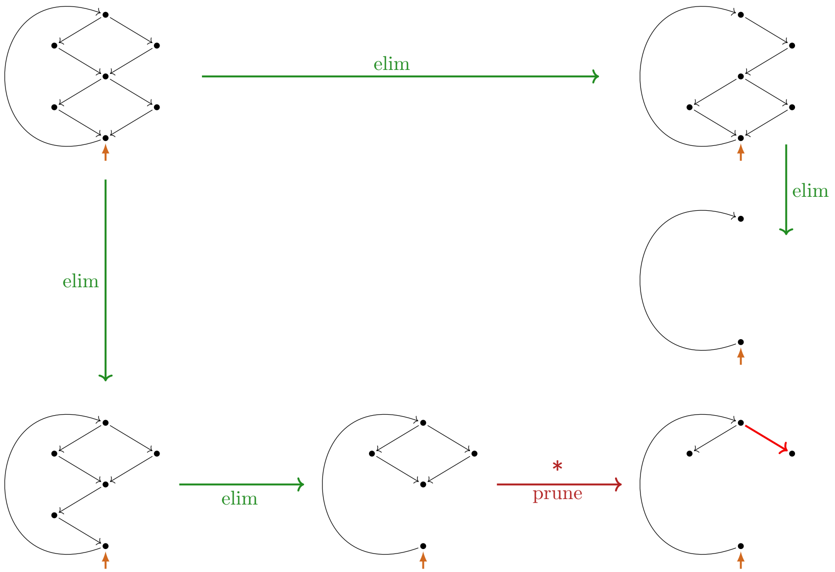
'Critical pair': bi-loop elimination



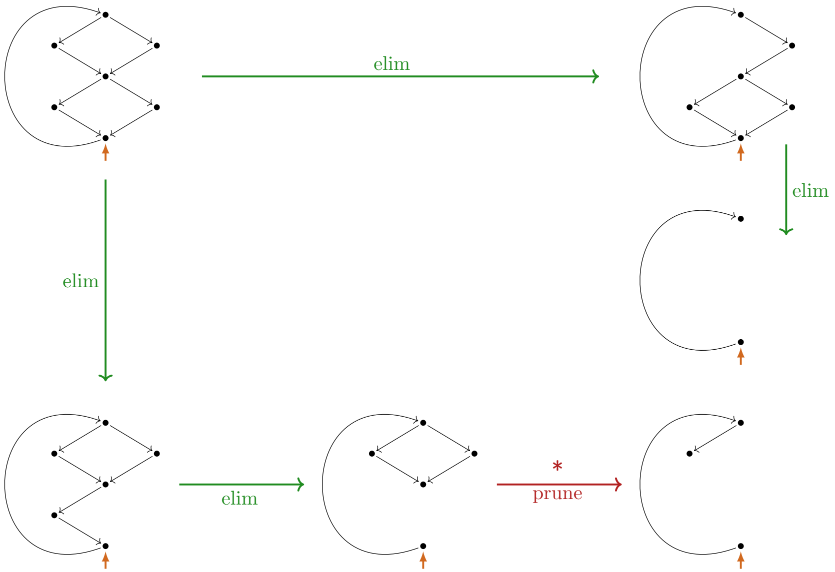
'Critical pair': bi-loop elimination



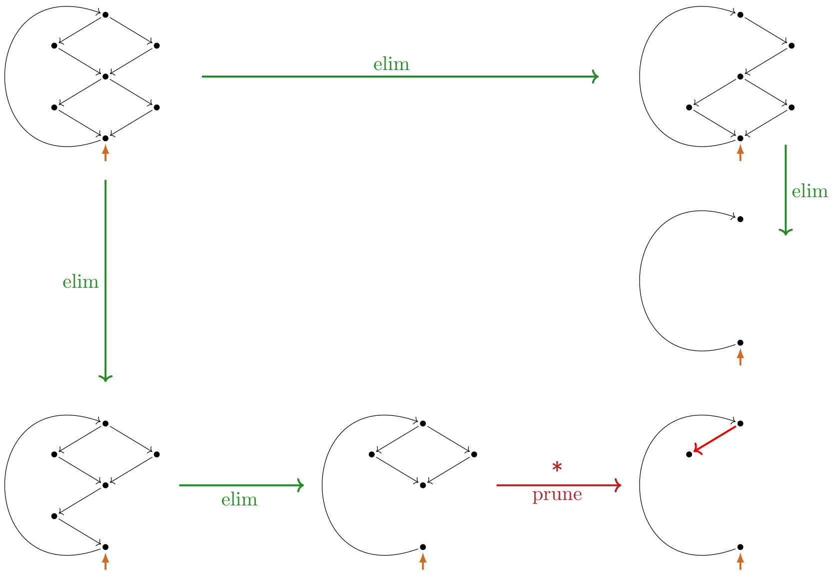
'Critical pair': bi-loop elimination



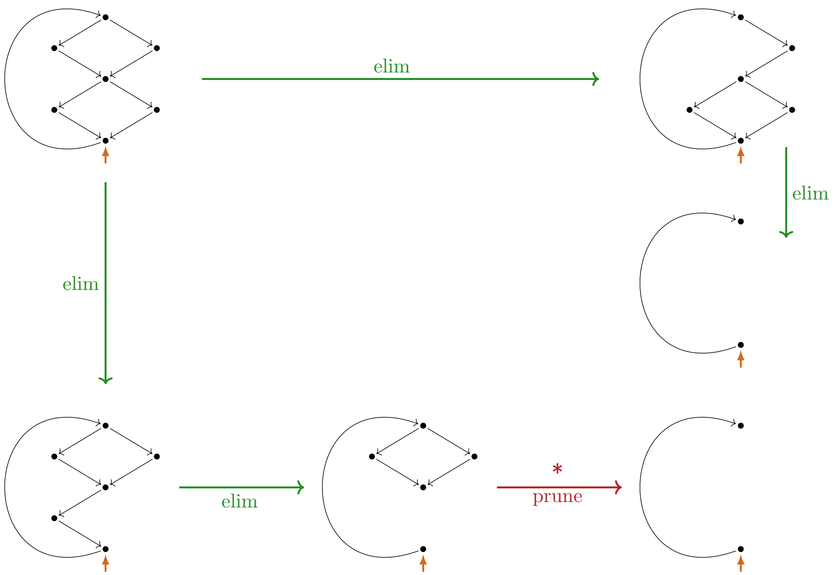
'Critical pair': bi-loop elimination



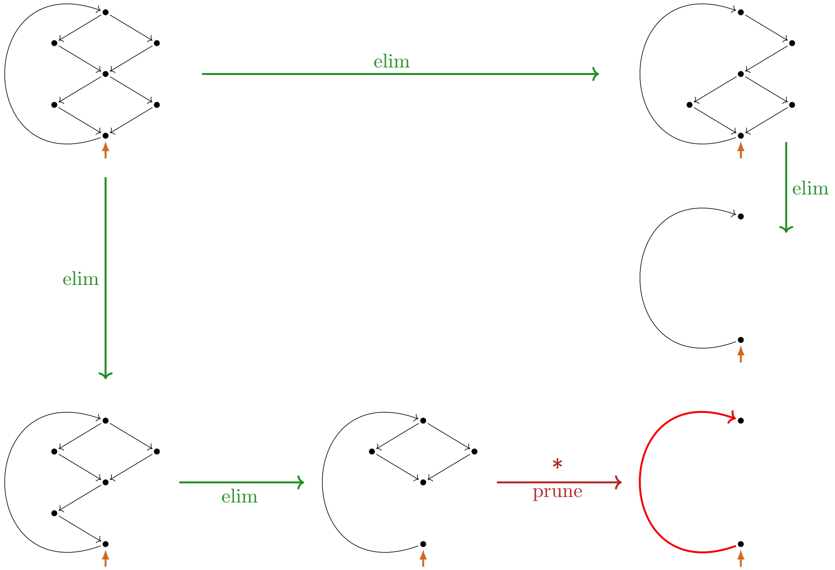
'Critical pair': bi-loop elimination



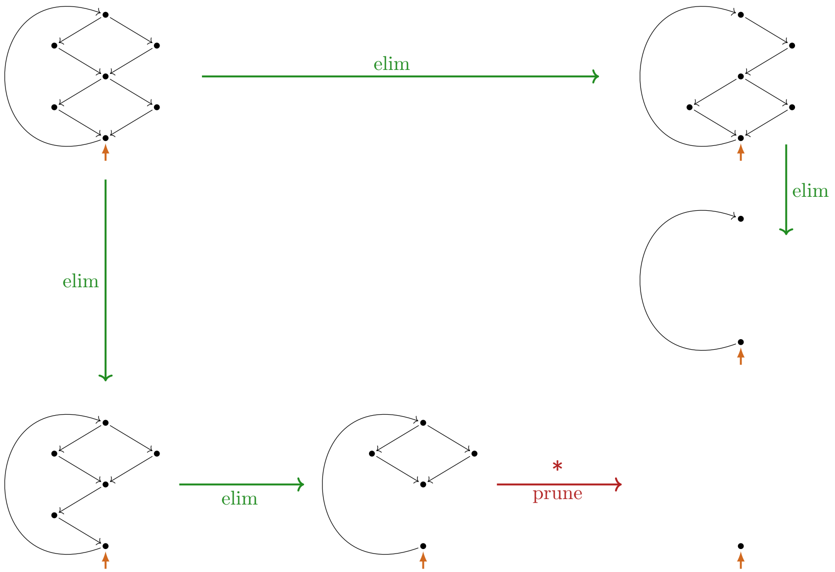
'Critical pair': bi-loop elimination



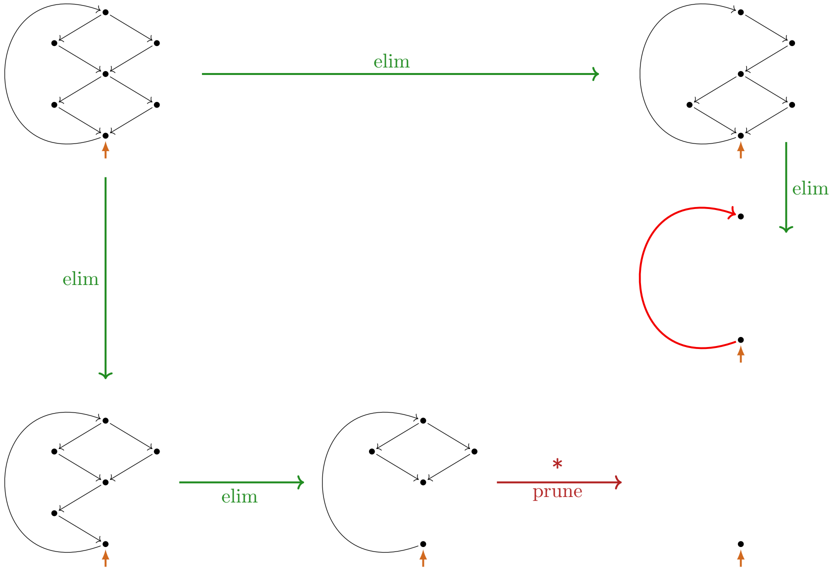
'Critical pair': bi-loop elimination



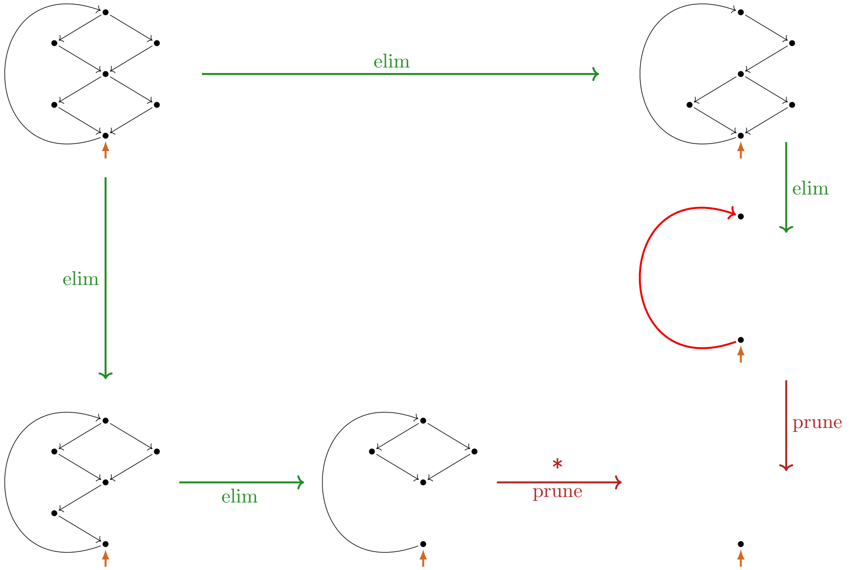
'Critical pair': bi-loop elimination



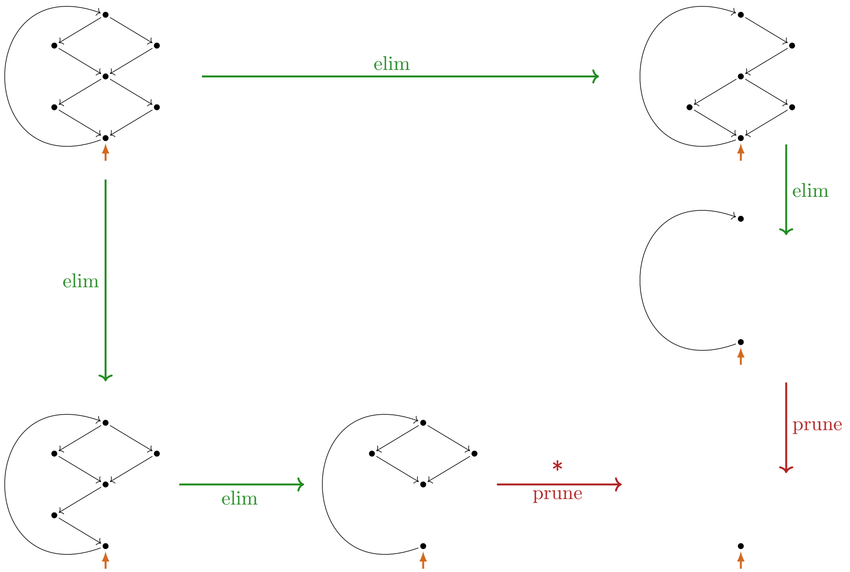
'Critical pair': bi-loop elimination



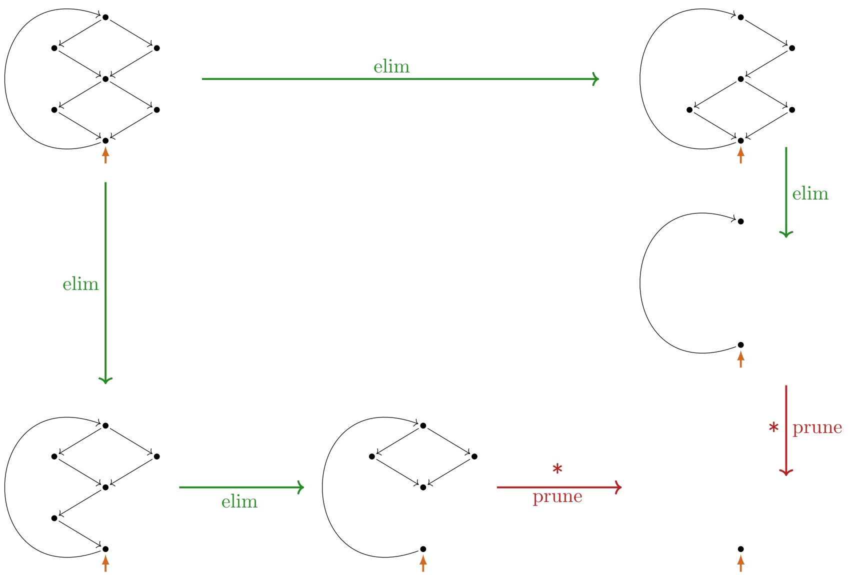
'Critical pair': bi-loop elimination



'Critical pair': bi-loop elimination



'Critical pair': bi-loop elimination



Loop elimination, and properties

$\longrightarrow_{\text{elim}}$: eliminate a transition-induced loop by:

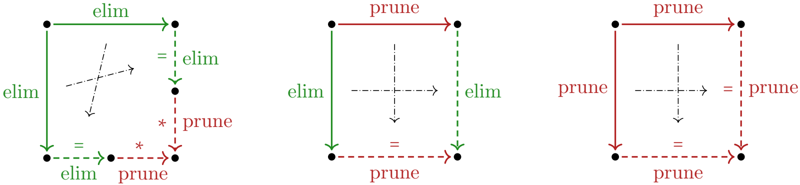
- removing the loop-entry transition(s)
- garbage collection

$\longrightarrow_{\text{prune}}$: remove a transition to a deadlocking state

Lemma

(i) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is terminating.*

(ii) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is decreasing, and so due to (i) locally confluent.*



Loop elimination, and properties

$\longrightarrow_{\text{elim}}$: eliminate a transition-induced loop by:

- removing the loop-entry transition(s)
- garbage collection

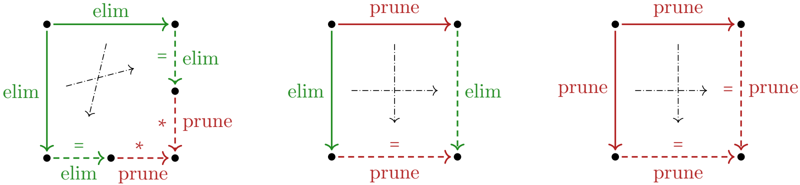
$\longrightarrow_{\text{prune}}$: remove a transition to a deadlocking state

Lemma

(i) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is terminating.*

(ii) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is decreasing, and so due to (i) locally confluent.*

(iii) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is confluent.*



Structure property LEE

Definition

A process graph G satisfies LEE (*loop existence and elimination*) if:

$$\exists G_0 \left(G \longrightarrow^{*}_{\text{elim}} G_0 \not\longrightarrow_{\text{elim}} \right.$$
$$\left. \wedge\; G_0 \text{ has no infinite trace} \right).$$

Structure property LEE

Definition

A process graph G satisfies LEE (*loop existence and elimination*) if:

$$\exists G_0 \left(G \longrightarrow^{*}_{\text{elim}} G_0 \not\longrightarrow_{\text{elim}} \wedge\ G_0 \text{ has no infinite trace} \right).$$

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

(ii) There is an $\longrightarrow_{\text{elim}}$ normal form without an infinite trace.

Structure property LEE

Definition

A process graph G satisfies LEE (*loop existence and elimination*) if:

$$\exists G_0 \left(G \longrightarrow^{*}_{\text{elim}} G_0 \not\longrightarrow_{\text{elim}} \wedge\ G_0 \text{ has no infinite trace} \right).$$

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

(ii) There is an $\longrightarrow_{\text{elim}}$ normal form without an infinite trace.

(iii) There is an $\longrightarrow_{\text{elim},\text{prune}}$ normal form without an infinite trace.

Structure property LEE

Definition

A process graph G satisfies LEE (*loop existence and elimination*) if:

$$\exists G_0 \left(G \longrightarrow^*_{\text{elim}} G_0 \not\longrightarrow_{\text{elim}} \wedge\ G_0 \text{ has no infinite trace} \right).$$

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

(ii) *There is an $\longrightarrow_{\text{elim}}$ normal form without an infinite trace.*

(iii) *There is an $\longrightarrow_{\text{elim},\text{prune}}$ normal form without an infinite trace.*

(iv) *Every $\longrightarrow_{\text{elim}}$ normal form is without an infinite trace.*

(v) *Every $\longrightarrow_{\text{elim},\text{prune}}$ normal form is without an infinite trace.*

Structure property LEE

Definition

A process graph G satisfies LEE (*loop existence and elimination*) if:

$$\exists G_0 \left(G \longrightarrow^{*}_{\mathsf{elim}} G_0 \not\longrightarrow_{\mathsf{elim}} \right.$$
$$\left. \wedge \; G_0 \text{ has no infinite trace} \right).$$

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

(ii) *There is an $\longrightarrow_{\mathsf{elim}}$ normal form without an infinite trace.*

(iii) *There is an $\longrightarrow_{\mathsf{elim},\mathsf{prune}}$ normal form without an infinite trace.*

(iv) *Every $\longrightarrow_{\mathsf{elim}}$ normal form is without an infinite trace.*

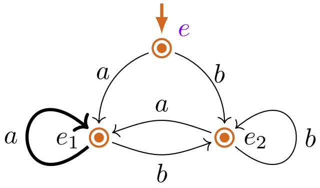
(v) *Every $\longrightarrow_{\mathsf{elim},\mathsf{prune}}$ normal form is without an infinite trace.*

Theorem (efficient decidability)

The problem of deciding LEE(G) for process graphs G is in PTIME.

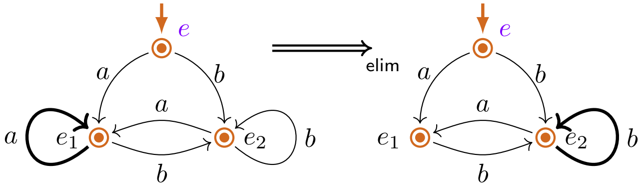
Failure of LEE in general (example)

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



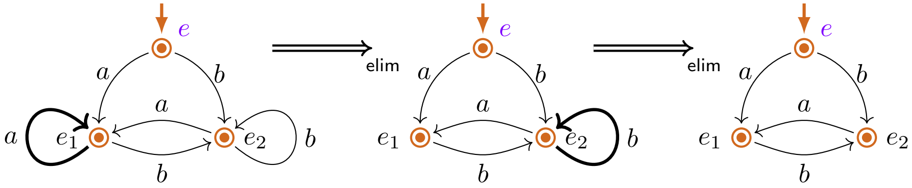
Failure of LEE in general (example)

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



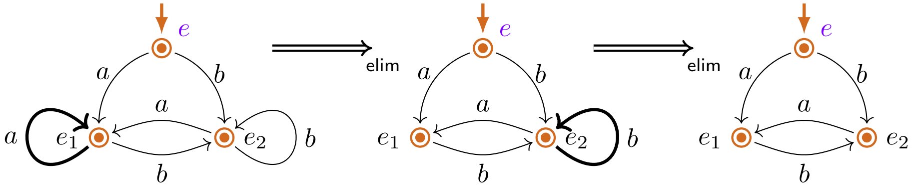
Failure of LEE in general (example)

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



Failure of LEE in general (example)

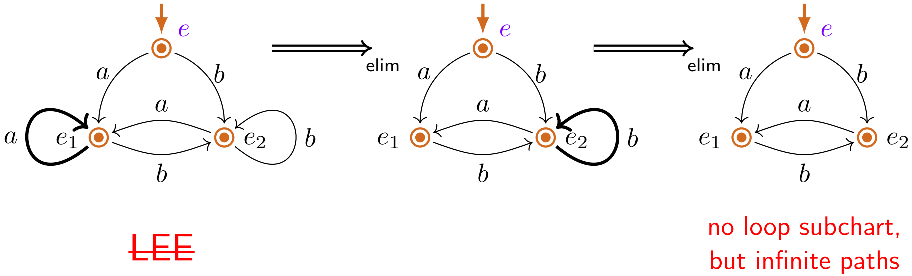
$P((a^* \cdot b^*)^*)$



no loop subchart,
but infinite paths

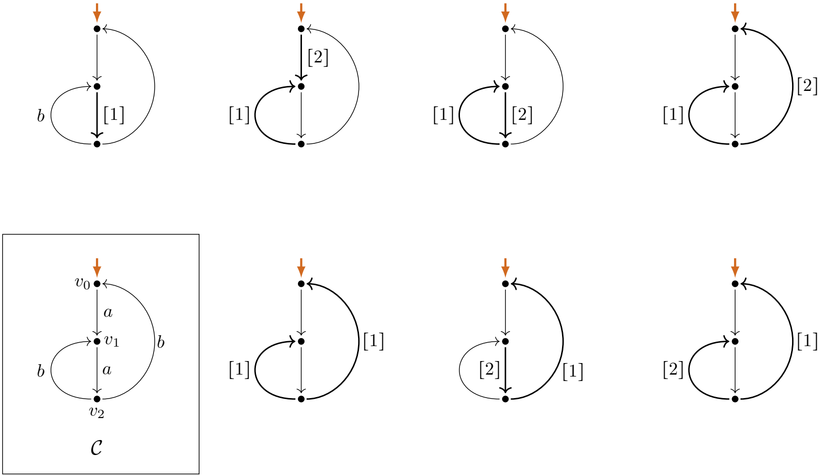
Failure of LEE in general (example)

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

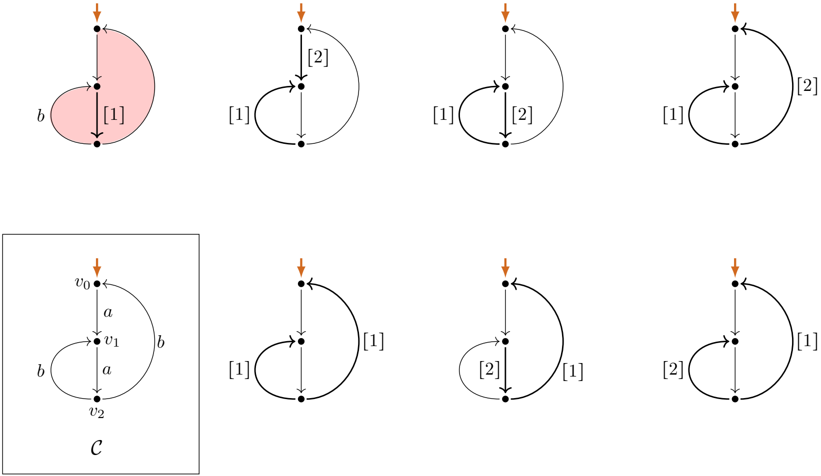


~~LEE~~

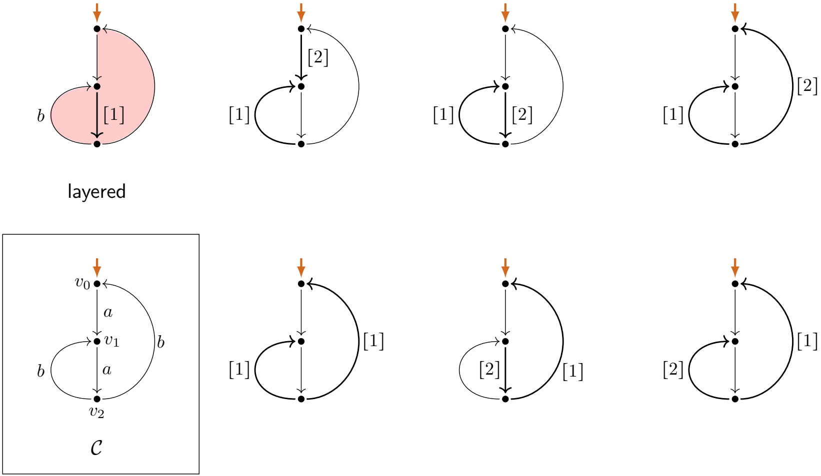
7 LEE-witnesses



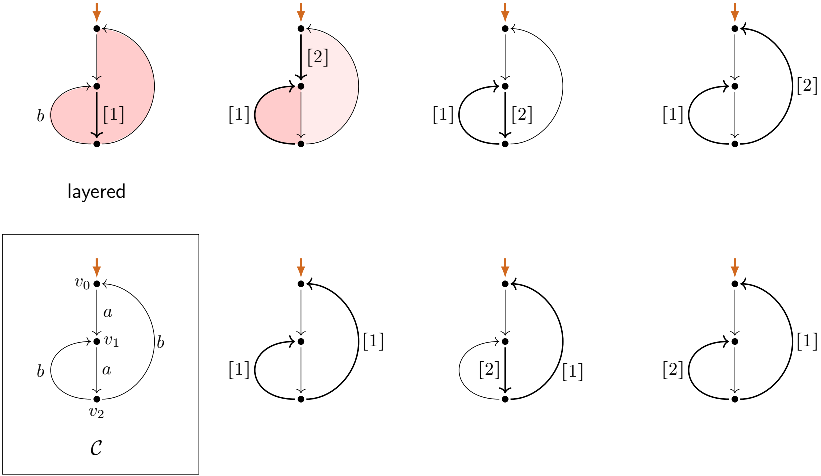
7 LEE-witnesses



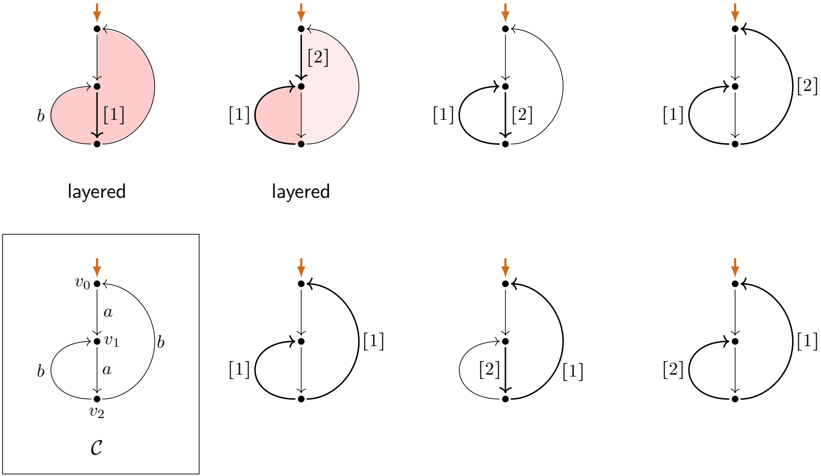
7 LEE-witnesses



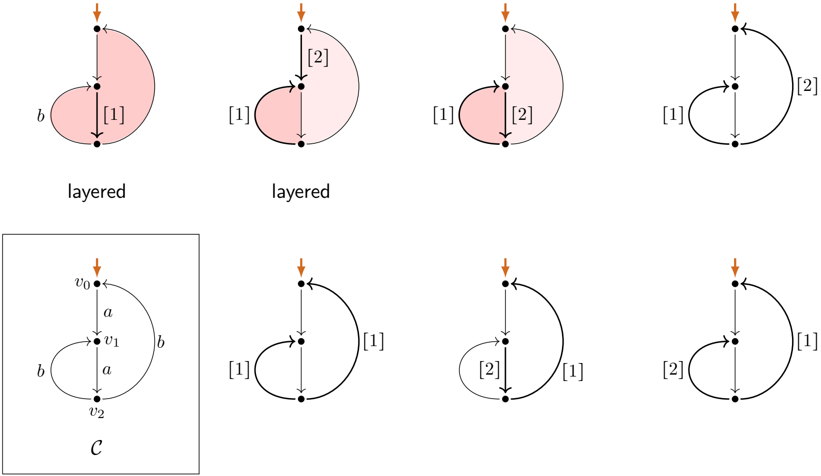
7 LEE-witnesses



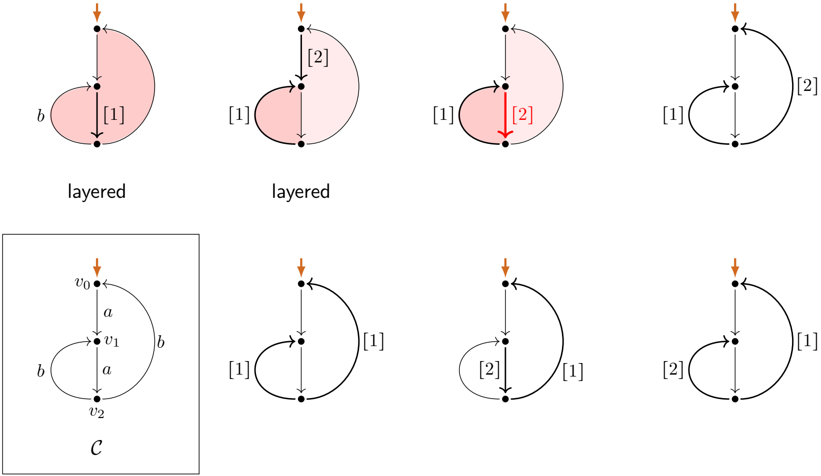
7 LEE-witnesses



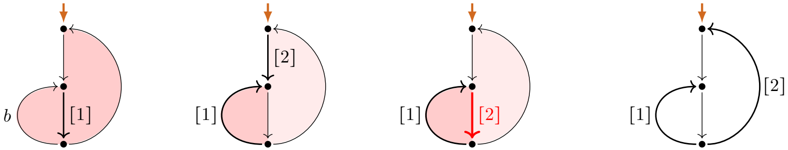
7 LEE-witnesses



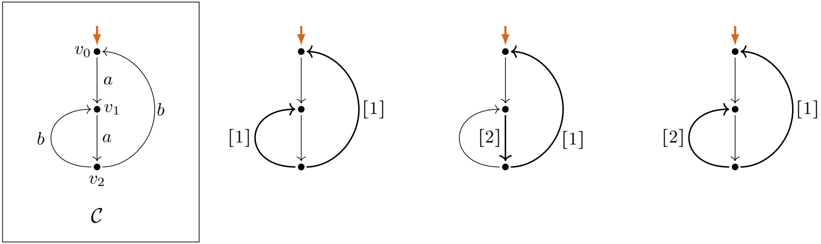
7 LEE-witnesses



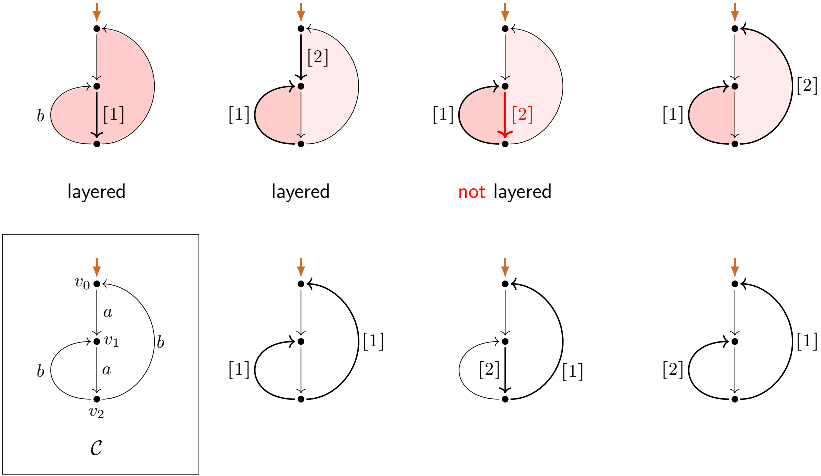
7 LEE-witnesses



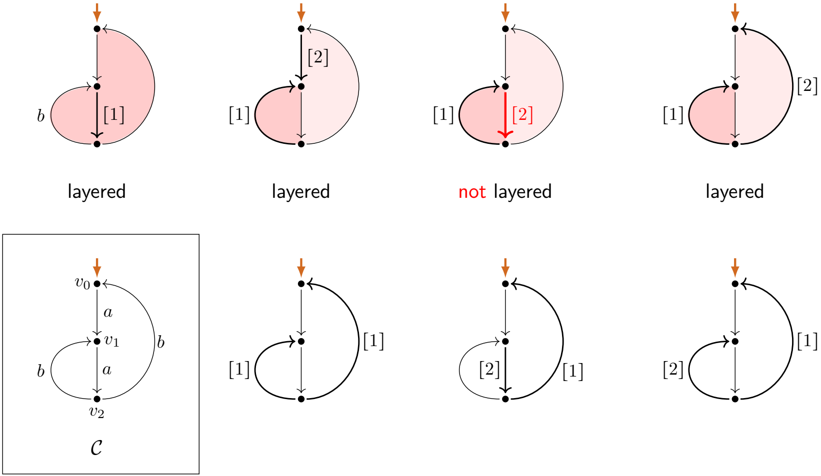
layered layered not layered



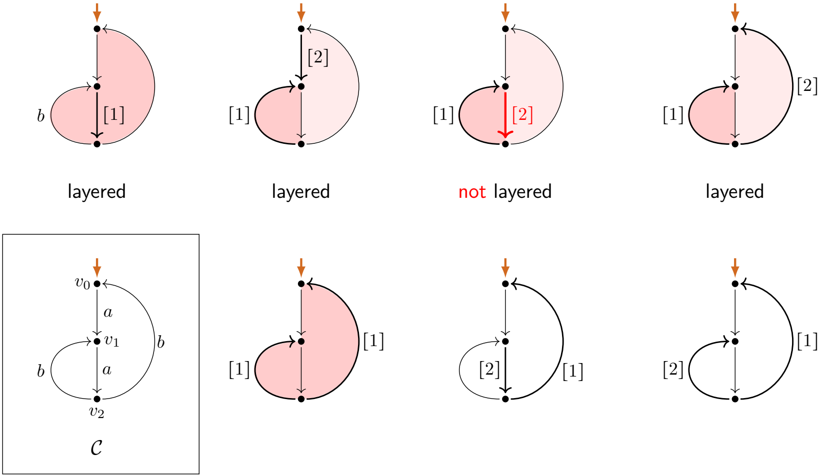
7 LEE-witnesses



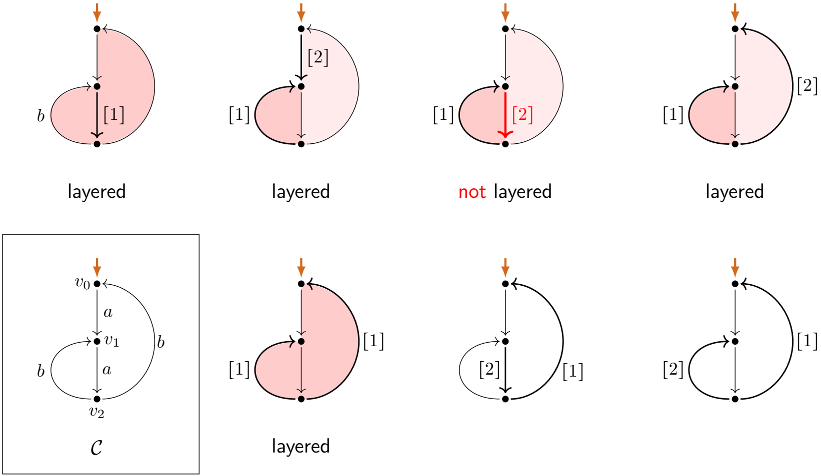
7 LEE-witnesses



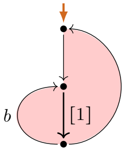
7 LEE-witnesses



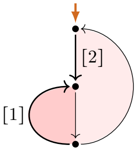
7 LEE-witnesses



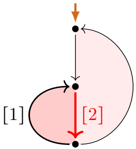
7 LEE-witnesses



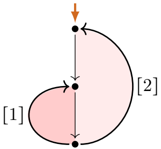
layered



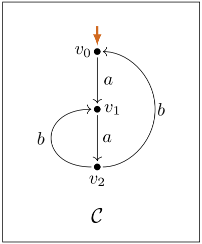
layered

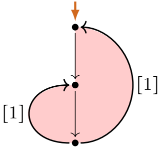


not layered

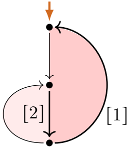


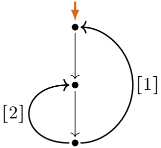
layered



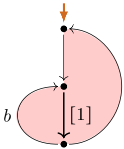


layered

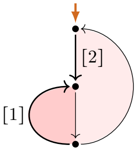




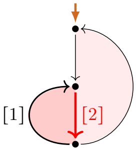
7 LEE-witnesses



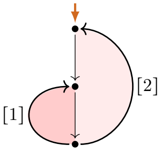
layered



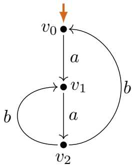
layered



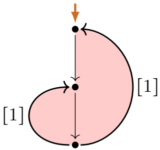
not layered



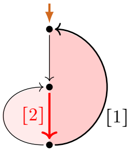
layered

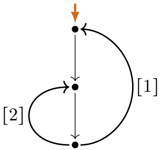


$\mathcal{C}$

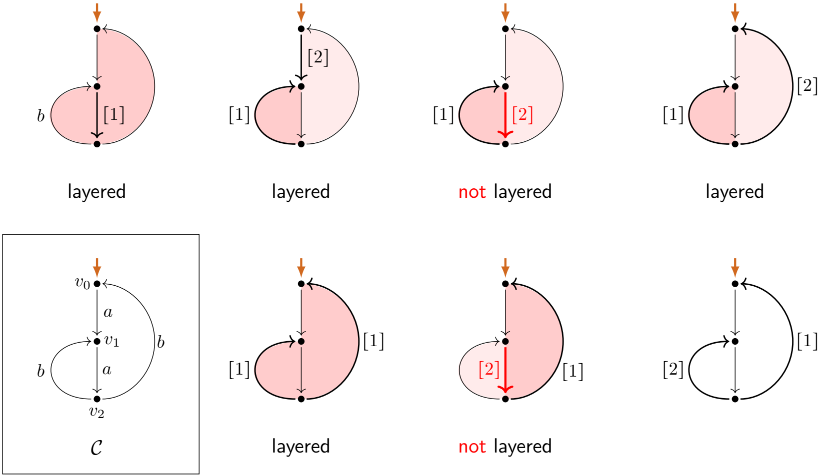


layered

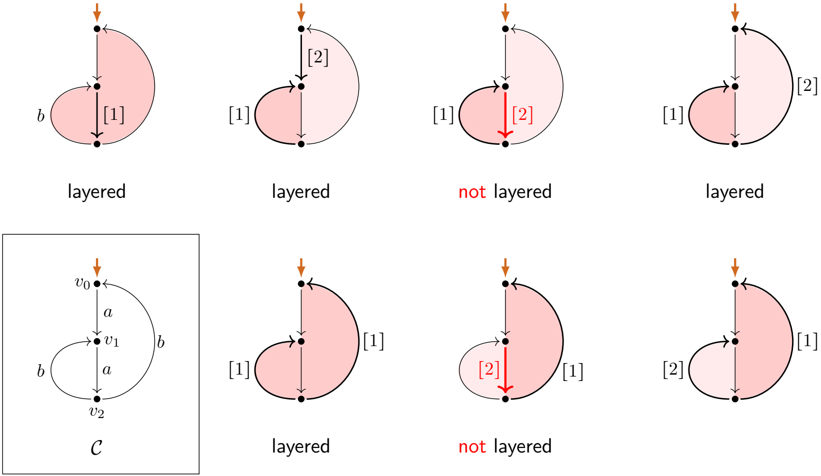




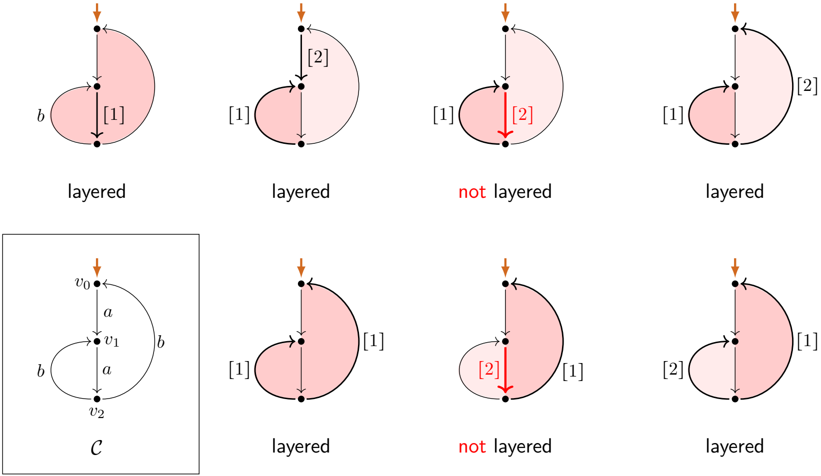
7 LEE-witnesses



7 LEE-witnesses



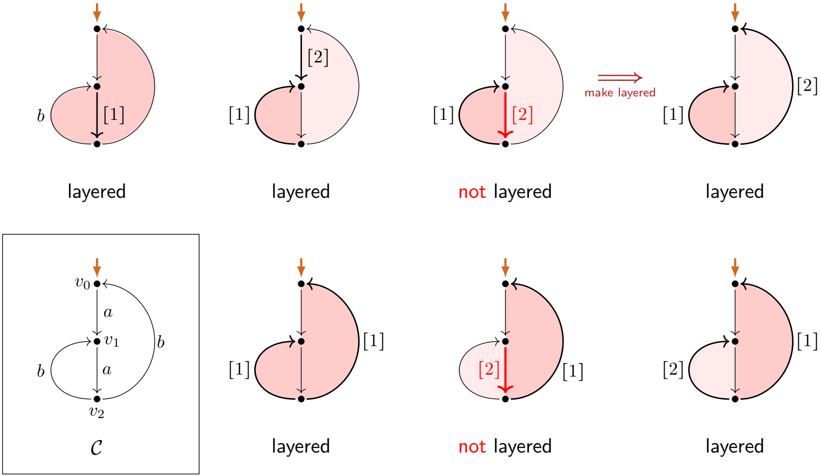
7 LEE-witnesses



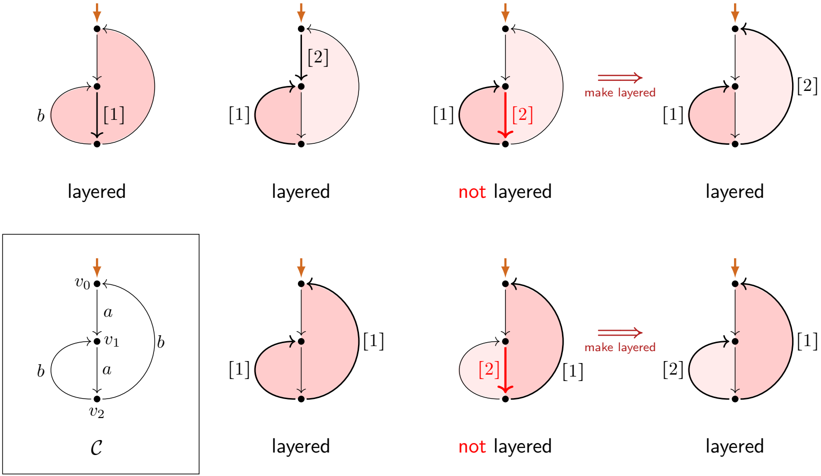
layered layered not layered layered

layered not layered layered

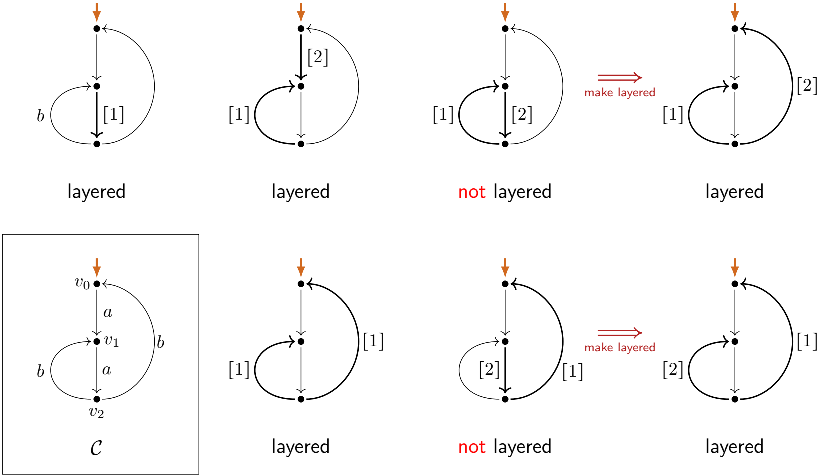
7 LEE-witnesses



7 LEE-witnesses



7 LEE-witnesses



Interpretation/extraction correspondences with LEE

($\Leftarrow$ G/Fokkink 2020, G 2021)

(Int)$_{P}^{(*/\cancel{1})}$: *$P^{\bullet}$-$(*/\cancel{1})$-expressible graphs have structural property* LEE

Process **i**nterpretations $P(e)$
of $(*/\cancel{1})$ regular expressions e
are finite process graphs that satisfy LEE.

(Extr)$_{P}$: LEE *implies* $[\![\cdot]\!]_{P}$*-expressibility*

From every finite process graph G with LEE
a regular expression e can be **e**xtracted
such that $G \leftrightarrow P(e)$.

Interpretation/extraction correspondences with LEE

($\Leftarrow$ G/Fokkink 2020, G 2021)

(Int)$_P^{(*/\cancel{1})}$: $P^\bullet$-$(*/\cancel{1})$*-expressible graphs have structural property* LEE

Process **i**nterpretations $P(e)$
of $(*/\cancel{1})$ regular expressions e
are finite process graphs that satisfy LEE.

(Extr)$_P$: LEE *implies* $[\![\cdot]\!]_P$*-expressibility*

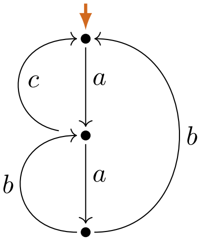
From every finite process graph G with LEE
a regular expression e can be **e**xtracted
such that $G \leftrightarroweq P(e)$.

(Coll): LEE *is preserved under collapse*

The class of finite process graphs with LEE
is closed under bisimulation **c**ollapse.

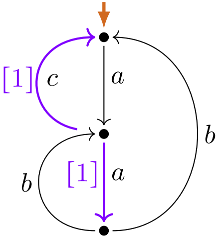
Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

G_2



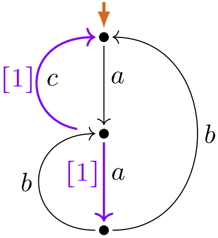
Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

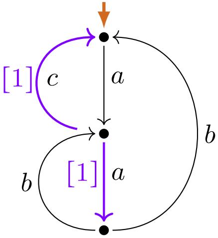
$\widehat{G_2}$



$$(\qquad\qquad\qquad\qquad)^* \cdot 0$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$

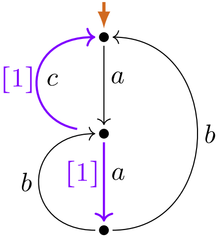


$\downarrow a$

$$(\qquad\qquad)^* \cdot 0$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



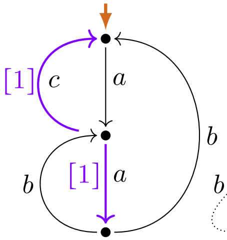
$$(a \cdot 1) \cdot \Big(\Big(\qquad\qquad \Big)^{*} \cdot 0\Big)$$

$$\downarrow a$$

$$\Big(\qquad\qquad \Big)^{*} \cdot 0$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$(a \cdot 1) \cdot ((\qquad)^* \cdot 0)$

a

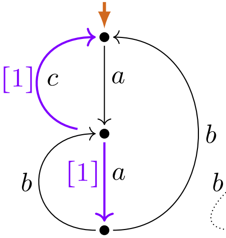
$(\qquad)^* \cdot 0$

b

b

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$(a \cdot 1) \cdot ((\qquad\qquad)^* \cdot 0)$

a

$(\qquad\qquad)^* \cdot 0$

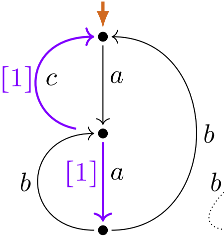
b

b

$(b \cdot 1 + b \cdot (a \cdot 1)) \cdot ((\qquad\qquad)^*) \cdot 0)$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



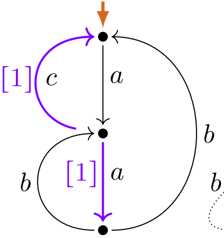
$(a \cdot 1) \cdot ((\qquad)^* \cdot 0)$

$(\qquad)^* \cdot 0$

$(b \cdot 1 + b \cdot (a \cdot 1)) \cdot ((\qquad)^*) \cdot 0)$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$$(a \cdot 1) \cdot \big(\big(\qquad\qquad \big)^* \cdot 0 \big)$$

$$\downarrow a$$

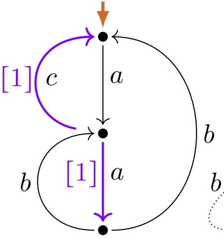
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0$$

$$\downarrow a$$

$$(b \cdot 1 + b \cdot (a \cdot 1)) \cdot \big(\big(\qquad\qquad \big)^* \big) \cdot 0 \big)$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$$(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \Big)$$

$$\downarrow a$$

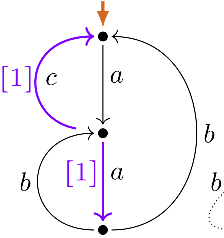
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0$$

$$\downarrow a$$

$$(b \cdot 1 + b \cdot (a \cdot 1)) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \big) \cdot 0 \Big)$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

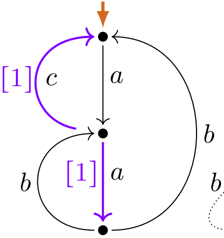
$\widehat{G_2}$



$$\begin{tikzcd}
\overbrace{(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\big)^{*} \cdot 0\Big)}^{e} \arrow[d, dotted, "a"] \\
\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\big)^{*} \cdot 0 \arrow[d, dotted, "a"] \arrow[u, bend left, dotted, "c"] \\
(b \cdot 1 + b \cdot (a \cdot 1)) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\big)^{*}\big) \cdot 0\Big) \arrow[u, bend left, dotted, "b"] \arrow[uu, bend right, dotted, "b"]
\end{tikzcd}$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$P(e) \underline{\rightarrow} G_2$

$$\overbrace{(a\cdot 1)\cdot\Big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\Big)}^{e}$$

$$\Big\downarrow a$$

$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

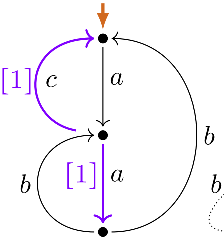
$$\Big\downarrow a$$

$$(b\cdot 1+b\cdot(a\cdot 1))\cdot\Big(\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0\Big)$$

c b b

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$$P(e) \mathrel{\underline{\rightarrow}} G_2$$

$$\overbrace{(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \Big)}^{e}$$

$$\downarrow a$$

$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0$$

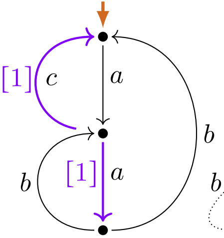
$$\downarrow a$$

$$(b \cdot 1 + b \cdot (a \cdot 1)) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0 \Big)$$

c b b

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$$P(e) \rightrightarrows G_2$$

$$\overbrace{(a\cdot 1)\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\big)}^{e}$$

$$\downarrow a$$

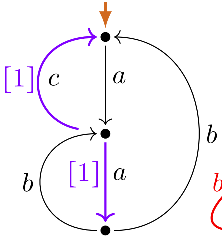
$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

$$(b\cdot 1+b\cdot(a\cdot 1))\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*})\cdot 0\big)$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$P(e) \rightleftharpoons G_2$

$$\overbrace{(a\cdot 1)\cdot\Big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\Big)}^{e}$$

$$\downarrow a$$

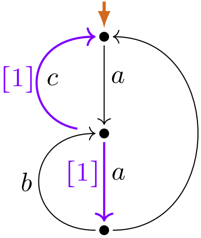
$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

$$(b\cdot 1+b\cdot(a\cdot 1))\cdot\Big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*})\cdot 0\Big)$$

Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



$$P(e) \mathrel{\underline{\to}} G_2 \not\simeq P(e)$$

$$\overbrace{(a\cdot 1)\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\big)}^{e}$$

$$\downarrow a$$

$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

$$(b\cdot 1+b\cdot(a\cdot 1))\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0\big)$$

Interpretation of extracted expression

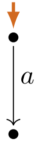
G_2'

$P(e) = G_2'$

$$\overbrace{(a\cdot 1)\cdot\Big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\Big)}^{e}$$

Interpretation of extracted expression

G_2'



$$P(e) = G_2'$$

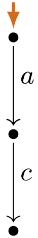
$$\overbrace{(a \cdot 1) \cdot \big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \big)}^{e}$$

$$\Big\downarrow a$$

$$(1 \cdot 1) \cdot \big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \big)$$

Interpretation of extracted expression

G_2'



$P(e) = G_2'$

$$\overbrace{(a \cdot 1) \cdot \big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \big)}^{e}$$

$$\big\downarrow a$$

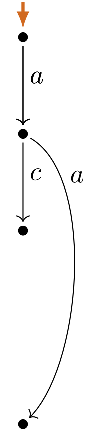
$$(1 \cdot 1) \cdot \big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \big)$$

$$\big\downarrow c$$

$$\big((1 \cdot (a \cdot 1)) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \big) \cdot 0$$

Interpretation of extracted expression

G_2'



$P(e) = G_2'$

$$\overbrace{(a\cdot 1)\cdot\big((c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1)))^{*}\cdot 0\big)}^{e}$$

$$\downarrow a$$

$$(1\cdot 1)\cdot\big((c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1)))^{*}\cdot 0\big)$$

$$\downarrow c$$

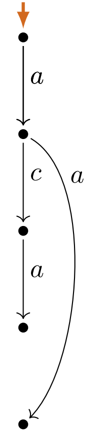
$$\big((1\cdot(a\cdot 1))\cdot(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1)))^{*}\big)\cdot 0$$

$$\xrightarrow{a}$$

$$\big((1\cdot(b\cdot 1+b\cdot(a\cdot 1)))\cdot(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1)))^{*}\big)\cdot 0$$

Interpretation of extracted expression

G_2'



$\textcolor{green}{P}(\textcolor{orange}{e}) = G_2'$

$$\overbrace{(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0 \Big)}^{e}$$

$$\downarrow a$$

$$(1 \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0 \Big)$$

$$\downarrow c$$

$$\Big((1 \cdot (\textcolor{red}{a} \cdot 1)) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \Big) \cdot 0$$

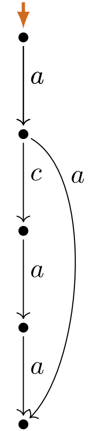
$$\textcolor{red}{\downarrow a}$$

$$\Big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \Big) \cdot 0 \Big)$$

$$\Big((1 \cdot (b \cdot 1 + b \cdot (a \cdot 1))) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \Big) \cdot 0$$

Interpretation of extracted expression

G'_2

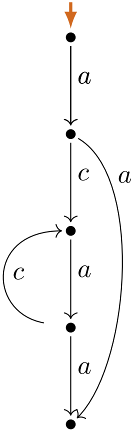


$P(e) = G'_2$

$$
\begin{array}{c}
\overbrace{(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \Big)}^{e} \\
\big\downarrow a \\
(1 \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \Big) \\
\big\downarrow c \\
\Big((1 \cdot (a \cdot 1)) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0 \\
\big\downarrow a \\
\Big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0 \Big) \\
\big\downarrow a \\
\Big((1 \cdot (b \cdot 1 + b \cdot (a \cdot 1))) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0
\end{array}
$$

Interpretation of extracted expression

G_2'



$P(e) = G_2'$

$$\overbrace{(a \cdot 1) \cdot \big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0 \big)}^{e}$$

$$\Big\downarrow a$$

$$(1 \cdot 1) \cdot \big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0 \big)$$

$$\Big\downarrow c$$

$$\big((1 \cdot (a \cdot 1)) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \big) \cdot 0$$

$$\Big\downarrow a$$

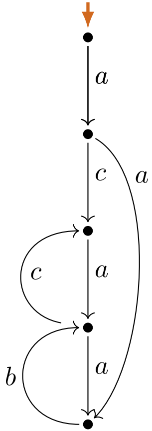
$$\big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \big) \cdot 0 \big)$$

$$\Big\downarrow a$$

$$\big((1 \cdot (b \cdot 1 + b \cdot (a \cdot 1))) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \big) \cdot 0$$

Interpretation of extracted expression

G'_2



$P(e) = G'_2$

$$\overbrace{(a\cdot 1)\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\big)}^{e}$$

$$\downarrow a$$

$$(1\cdot 1)\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\big)$$

$$\downarrow c$$

$$\big((1\cdot(a\cdot 1))\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0$$

$$\downarrow a$$

$$\big((1\cdot 1)\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0\big)$$

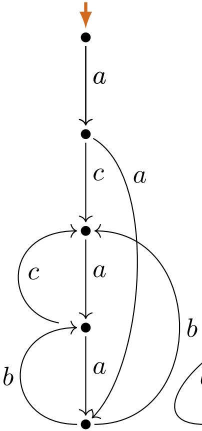
$$\downarrow a$$

$$\big((1\cdot(b\cdot 1+b\cdot(a\cdot 1)))\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0$$

c b a

Interpretation of extracted expression

G_2'



$P(e) = G_2'$

$$\overbrace{(a\cdot 1)\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\big)}^{e}$$

$$\Big\downarrow a$$

$$(1\cdot 1)\cdot\big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\big)$$

$$\Big\downarrow c$$

$$\big((1\cdot(a\cdot 1))\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0$$

$$\Big\downarrow a$$

$$\big((1\cdot 1)\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0\big)$$

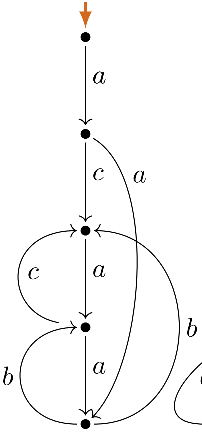
$$\Big\downarrow a$$

$$\big((1\cdot(b\cdot 1+b\cdot(a\cdot 1)))\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0$$

a c b b

Interpretation of extracted expression

G'_2



$$P(e) = G'_2 \rightarrow G_2$$

$$\overbrace{(a \cdot 1) \cdot \left(\left(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\right)^{*} \cdot 0\right)}^{e}$$

$$\big\downarrow a$$

$$(1 \cdot 1) \cdot \left(\left(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\right)^{*} \cdot 0\right)$$

$$\big\downarrow c$$

$$\left((1 \cdot (a \cdot 1)) \cdot \left(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\right)^{*}\right) \cdot 0$$

$$\big\downarrow a$$

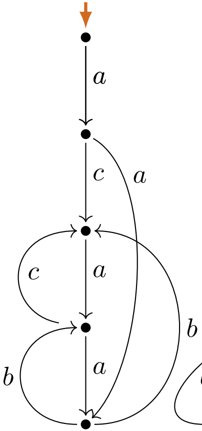
$$\left((1 \cdot 1) \cdot \left(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\right)^{*}\right) \cdot 0\Big)$$

$$\big\downarrow a$$

$$\left((1 \cdot (b \cdot 1 + b \cdot (a \cdot 1))) \cdot \left(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\right)^{*}\right) \cdot 0$$

Interpretation of extracted expression

G_2'



$$P(e) = G_2' \rightarrow G_2 \not\simeq G_2'$$

$$\overbrace{(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \Big)}^{e}$$

$$\downarrow a$$

$$(1 \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0 \Big)$$

$$\downarrow c$$

$$\Big((1 \cdot (a \cdot 1)) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0$$

$$\downarrow a$$

$$\Big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0 \Big)$$

$$\downarrow a$$

$$\Big((1 \cdot (b \cdot 1 + b \cdot (a \cdot 1))) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0$$

LEE under bisimulation?

LEE under bisimulation

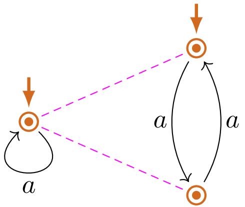
Observation

- LEE is not invariant under bisimulation.

LEE under bisimulation

Observation

- LEE is not invariant under bisimulation.

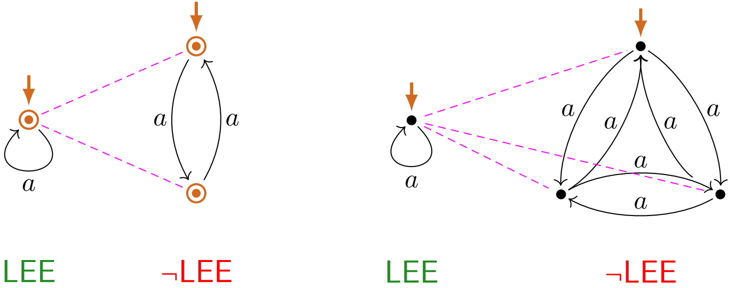


LEE ¬LEE

LEE under bisimulation

Observation

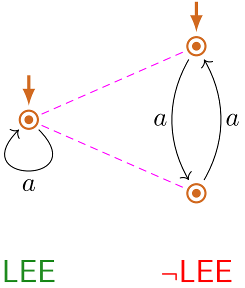
- LEE is not invariant under bisimulation.

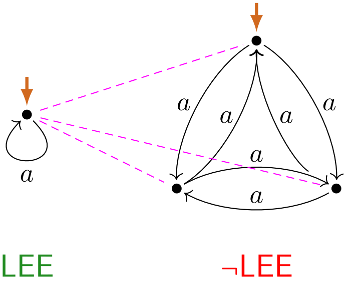


LEE under bisimulation

Observation

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





LEE under functional bisimulation

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\to}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

LEE under functional bisimulation

Lemma

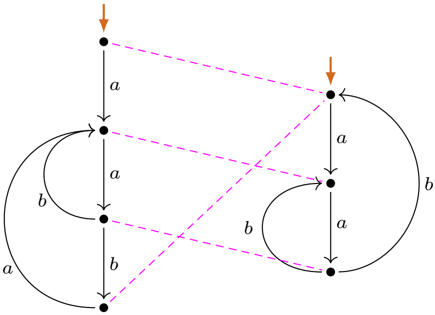
(i) LEE is preserved by functional bisimulations:

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\to}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

Proof (Idea).

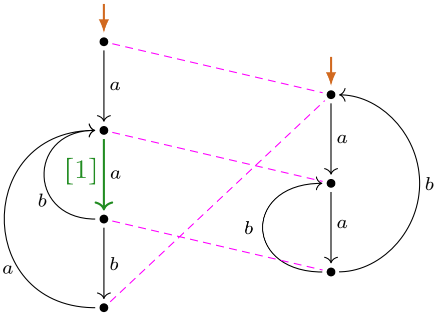
Use loop elimination in G_1 to carry out loop elimination in G_2.

Collapsing LEE-witnesses



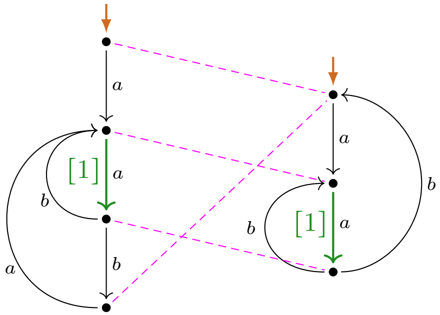
$P(a(a(b + ba))^* \cdot 0)$

Collapsing LEE-witnesses



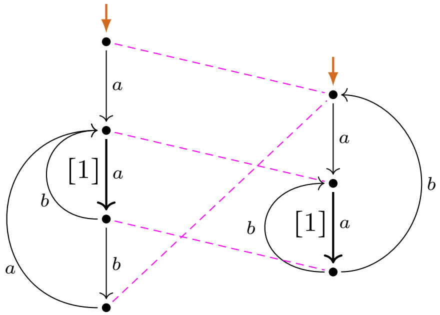
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



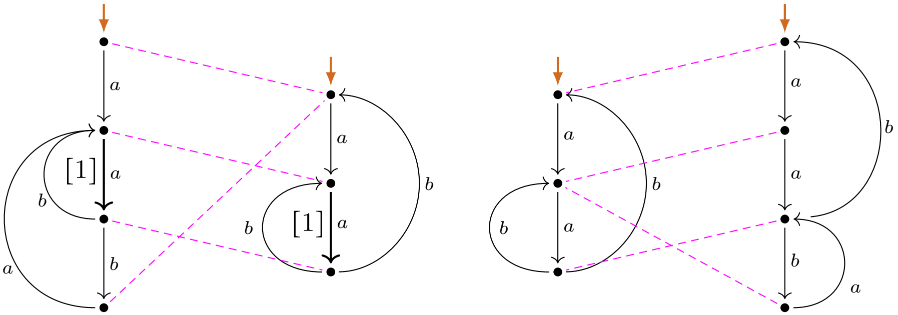
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



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

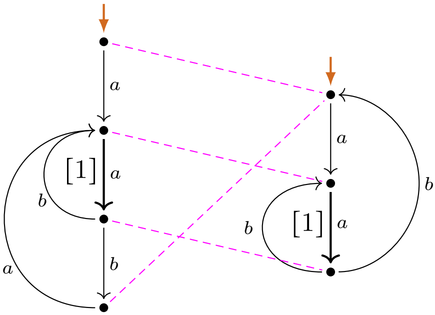
Collapsing LEE-witnesses

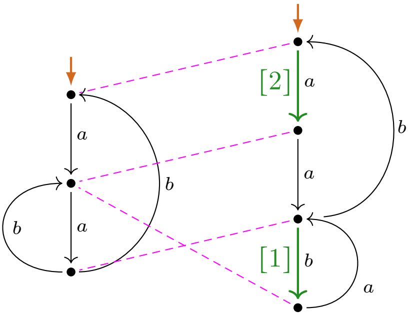


$P(a(a(b+ba))^* \cdot 0)$

$P((aa(ba)^* \cdot b)^* \cdot 0)$

Collapsing LEE-witnesses

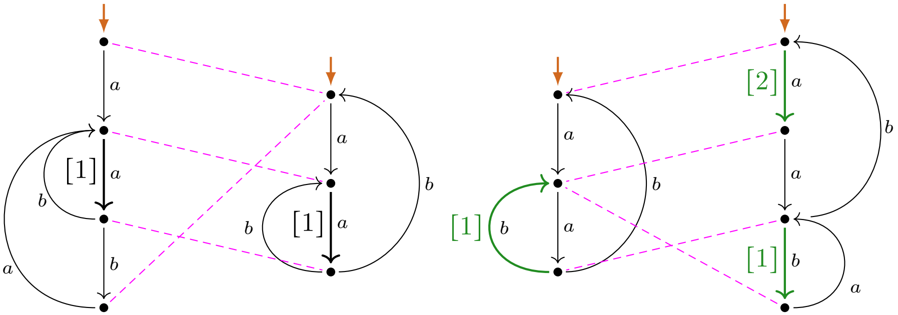




$P(a(a(b+ba))^* \cdot 0)$

$P((aa(ba)^* \cdot b)^* \cdot 0)$

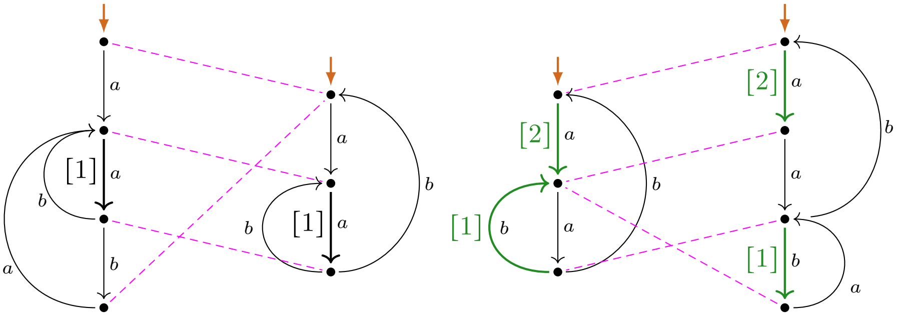
Collapsing LEE-witnesses



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

$P((aa(ba)^* \cdot b)^* \cdot 0)$

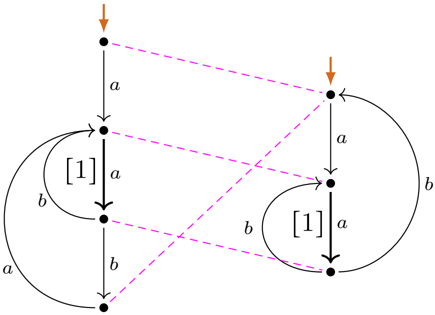
Collapsing LEE-witnesses

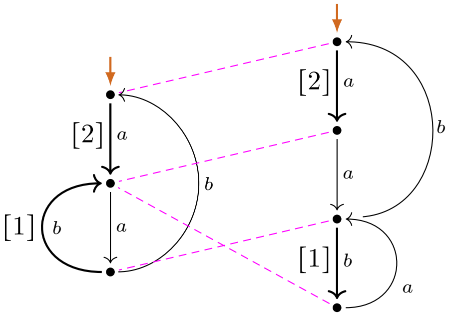


$P(a(a(b+ba))^* \cdot 0)$

$P((aa(ba)^* \cdot b)^* \cdot 0)$

Collapsing LEE-witnesses





$P(a(a(b+ba))^* \cdot 0)$ $P((aa(ba)^* \cdot b)^* \cdot 0)$

LEE under functional bisimulation

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\rightarrow}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

Idea of Proof for (i)

Use loop elimination in G_1 to carry out loop elimination in G_2.

LEE under functional bisimulation / bisimulation collapse

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\to}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

(ii) LEE *is preserved from a process graph to its bisimulation collapse:*

$$\mathsf{LEE}(G) \wedge G \textit{ has bisimulation collapse } C \implies \mathsf{LEE}(C)\,.$$

Idea of Proof for (i)

Use loop elimination in G_1 to carry out loop elimination in G_2.

LEE under functional bisimulation / bisimulation collapse

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\rightarrow}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

(ii) LEE *is preserved from a process graph to its bisimulation collapse:*

$$\mathsf{LEE}(G) \wedge G \textit{ has bisimulation collapse } C \implies \mathsf{LEE}(C)\,.$$

Idea of Proof for (i)

Use loop elimination in G_1 to carry out loop elimination in G_2.

- images of loop subcharts in G_1 under $\mathrel{\underline{\rightarrow}}$ are loop subcharts of G_2.

LEE under functional bisimulation / bisimulation collapse

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\to}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

(ii) LEE *is preserved from a process graph to its bisimulation collapse:*

$$\mathsf{LEE}(G) \wedge G \textit{ has bisimulation collapse } C \implies \mathsf{LEE}(C)\,.$$

Idea of Proof for (i)

Use loop elimination in G_1 to carry out loop elimination in G_2.

- images of loop subcharts in G_1 under $\mathrel{\underline{\to}}$ are loop subcharts of G_2.
- eliminating a loop subchart from G_2 amounts, via $\mathrel{\underline{\to}}$, to eliminating a transition induced subgraph from G_1.

LEE under functional bisimulation / bisimulation collapse

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\rightarrow}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

(ii) LEE *is preserved from a process graph to its bisimulation collapse:*

$$\mathsf{LEE}(G) \wedge G \textit{ has bisimulation collapse } C \implies \mathsf{LEE}(C)\,.$$

Idea of Proof for (i)

Use loop elimination in G_1 to carry out loop elimination in G_2.

- images of loop subcharts in G_1 under $\mathrel{\underline{\rightarrow}}$ are loop subcharts of G_2.
- eliminating a loop subchart from G_2 amounts, via $\mathrel{\underline{\rightarrow}}$, to eliminating a transition induced subgraph from G_1.
- LEE is preserved by dropping transition-induced subgraphs.

LEE under functional bisimulation / bisimulation collapse

Lemma

(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \mathrel{\underline{\rightarrow}} G_2 \implies \mathsf{LEE}(G_2)\,.$$

(ii) LEE *is preserved from a process graph to its bisimulation collapse:*

$$\mathsf{LEE}(G) \wedge G \textit{ has bisimulation collapse } C \implies \mathsf{LEE}(C)\,.$$

Idea of Proof for (i)

Use loop elimination in G_1 to carry out loop elimination in G_2.

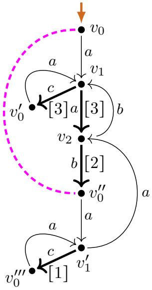
- images of loop subcharts in G_1 under $\mathrel{\underline{\rightarrow}}$ are loop subcharts of G_2.
- eliminating a loop subchart from G_2 amounts, via $\mathrel{\underline{\rightarrow}}$, to eliminating a transition induced subgraph from G_1.
- LEE is preserved by dropping transition-induced subgraphs.

Due to $\mathsf{LEE}(G_1)$, then such loop elimination in G_2 terminates in a graph without an infinite trace. This establishes $\mathsf{LEE}(G_2)$.

LLEE-preserving collapse (example, corollary)

Lemma (C)

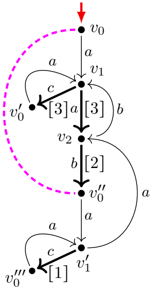
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

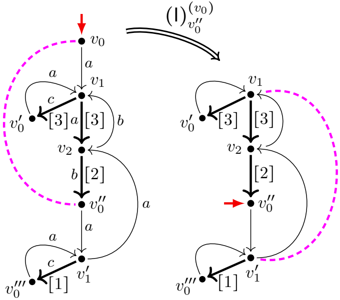
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

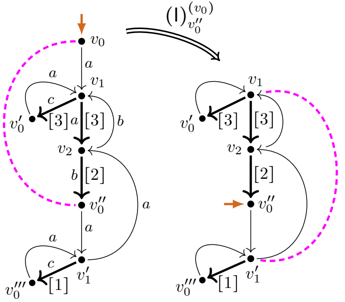
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

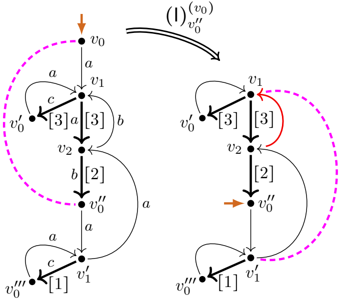
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

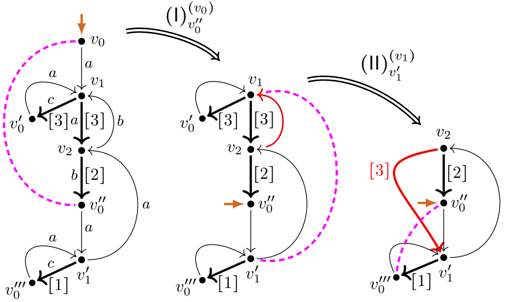
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

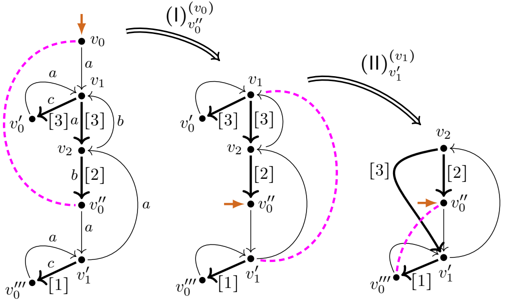
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

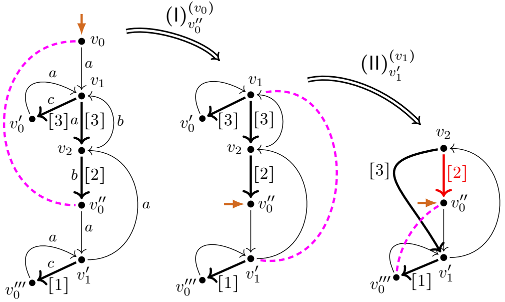
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

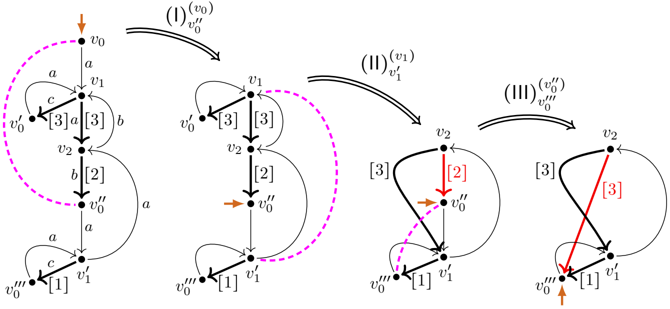
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

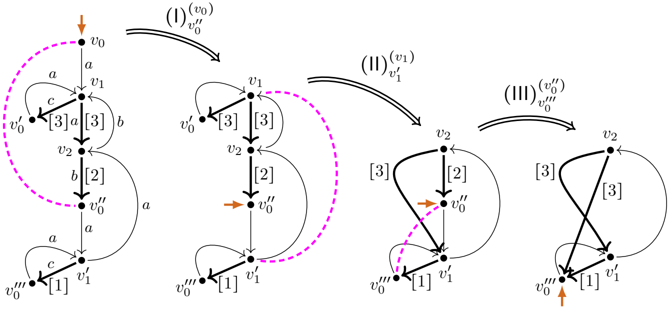
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

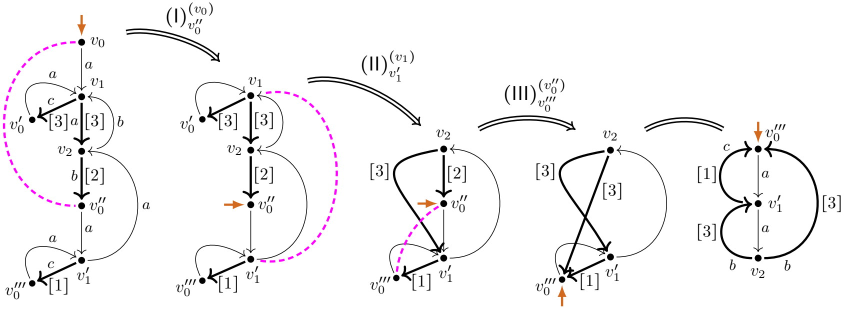
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

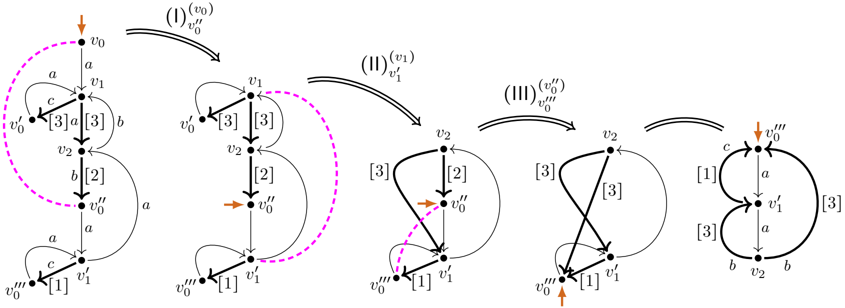
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



Corollary

A process graph is $[\![\cdot]\!]_P$-expressible by an $(*/\cancel{1})$ regular expression
if and only if its bisimulation collapse is a LLEE-graph.

Properties of LEE-charts

Theorem ($\Leftarrow$ G/Fokkink, 2020)

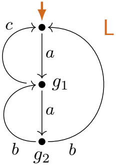
A process graph G
is $[\![\cdot]\!]_P$-expressible by an under-star-1-free regular expression
(i.e. P-expressible modulo bisimilarity by an $(\cancel{1}\backslash *)$ reg. expr.)
if and only if
the bisimulation collapse of G satisfies LEE.

Properties of LEE-charts

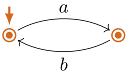
Theorem ($\Leftarrow$ G/Fokkink, 2020)

A process graph G
is $[\![\cdot]\!]_P$-expressible by an under-star-1-free regular expression
(i.e. P-expressible modulo bisimilarity by an $(\cancel{1}\backslash *)$ reg. expr.)
if and only if
the bisimulation collapse of G satisfies LEE.

Hence $[\![\cdot]\!]_P$-expressible | **not** $[\![\cdot]\!]_P$-expressible by 1-free regular expressions:

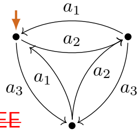


LEE



~~LEE~~

~~LEE~~



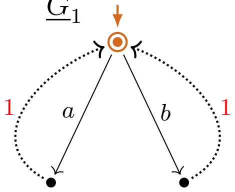
1-Graphs and induced graphs

Definition

$\xrightarrow{1} \cdot \cdots \cdot \xrightarrow{1} \cdot \xrightarrow{a} \quad \widehat{=} \quad \xrightarrow{(a]}$ induced a-transitions, for $a \in A$

$\xrightarrow{1} \cdot \cdots \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad \Downarrow^{(1)}$ induced termination.

$\underline{G}_1$

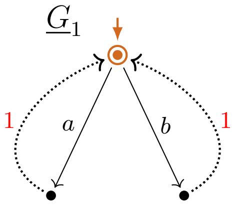


1-Graphs and induced graphs

Definition

$v_1 \xrightarrow{1} \cdot \; \cdots \; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$ induced a-transitions, for $a \in A$

$v \xrightarrow{1} \cdot \; \cdots \; \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v\Downarrow^{(1)}$ induced termination.

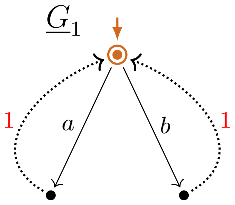


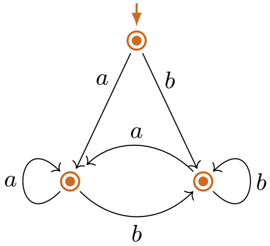
1-Graphs and induced graphs

Definition

$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$ induced a-transitions, for $a \in A$

$v \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v \Downarrow^{(1)}$ induced termination.



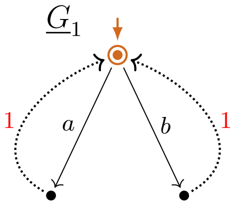


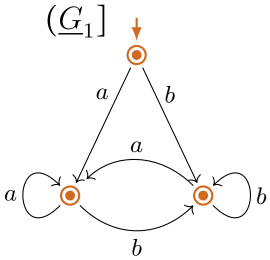
1-Graphs and induced graphs

Definition

$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$ induced a-transitions, for $a \in A$

$v \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v \Downarrow^{(1)}$ induced termination.





1-Graphs and induced graphs

Definition

$$v_1 \xrightarrow{1} \cdot \ \cdots \ \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$$ induced a-transitions, for $a \in A$

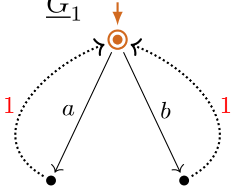
$$v \xrightarrow{1} \cdot \ \cdots \ \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v{\Downarrow}^{(1)}$$ induced termination.

Definition

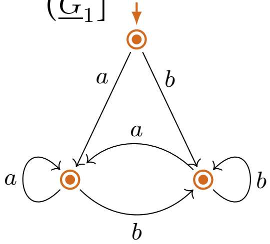
The induced (process) graph of a 1-graph $\underline{G} = \langle V, A, 1, v_s, \rightarrow, \Downarrow \rangle$ is:

$$(\underline{G}] = \langle V, A, v_s, \xrightarrow{(\cdot]}, {\Downarrow}^{(1)} \rangle \,.$$

$\underline{G}_1$



$(\underline{G}_1]$



1-Graphs and induced graphs

Definition

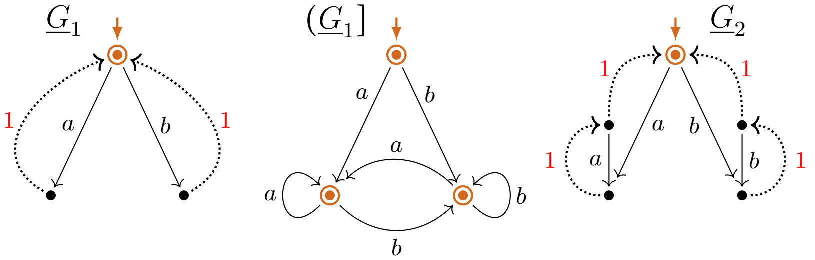
$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$ induced a-transitions, for $a \in A$

$v \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v{\Downarrow}^{(1)}$ induced termination.

Definition

The induced (process) graph of a 1-graph $\underline{G} = \langle V, A, 1, v_s, \to, \Downarrow \rangle$ is:

$$(\underline{G}] = \langle V, A, v_s, \xrightarrow{(\cdot]}, {\Downarrow}^{(1)} \rangle \,.$$



1-Graphs and induced graphs

Definition

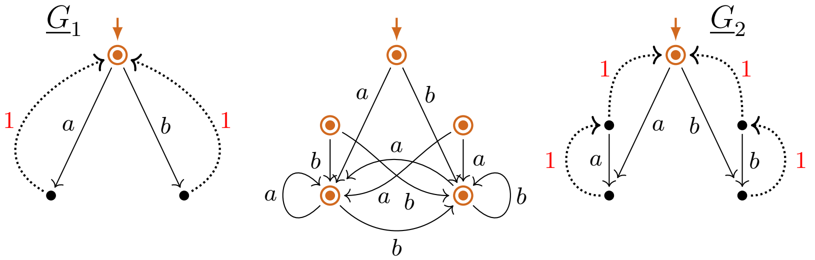
$$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$$ induced a-transitions, for $a \in A$

$$v \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v \Downarrow^{(1)}$$ induced termination.

Definition

The induced (process) graph of a 1-graph $\underline{G} = \langle V, A, 1, v_s, \to, \Downarrow \rangle$ is:

$$(\underline{G}] = \langle V, A, v_s, \xrightarrow{(\cdot]}, \Downarrow^{(1)} \rangle \,.$$



1-Graphs and induced graphs

Definition

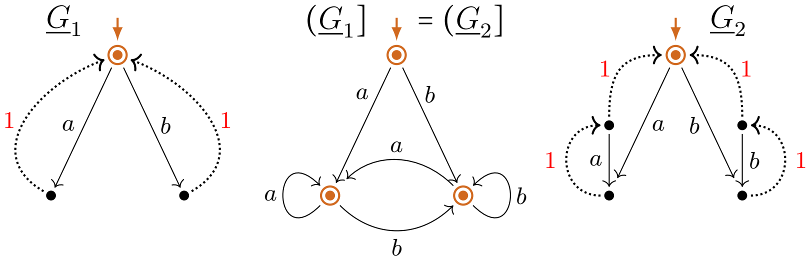
$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \;\;\widehat{=}\;\; v_1 \xrightarrow{(a]} v_2$ induced a-transitions, for $a \in A$

$v \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \Downarrow \;\;\widehat{=}\;\; v {\Downarrow}^{(1)}$ induced termination.

Definition

The induced (process) graph of a 1-graph $\underline{G} = \langle V, A, 1, v_s, \to, \Downarrow \rangle$ is:

$$(\underline{G}] = \langle V, A, v_s, \xrightarrow{(\cdot]}, {\Downarrow}^{(1)} \rangle \,.$$



1-LEE

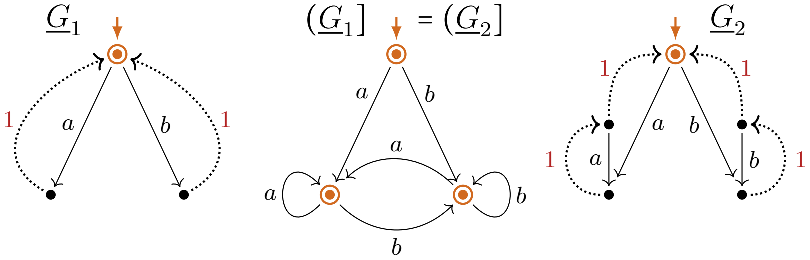
Definition

1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some 1-graph $\underline{G}$.

1-LEE

Definition

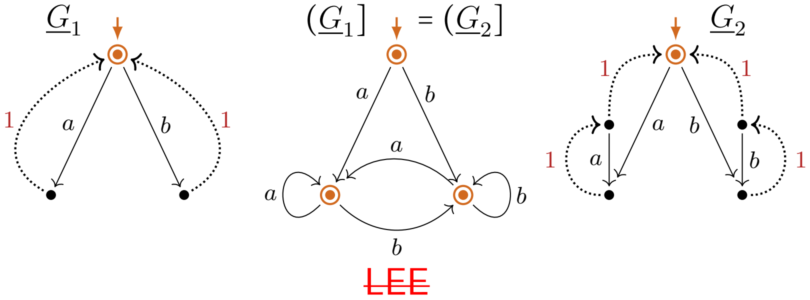
1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some 1-graph $\underline{G}$.



1-LEE

Definition

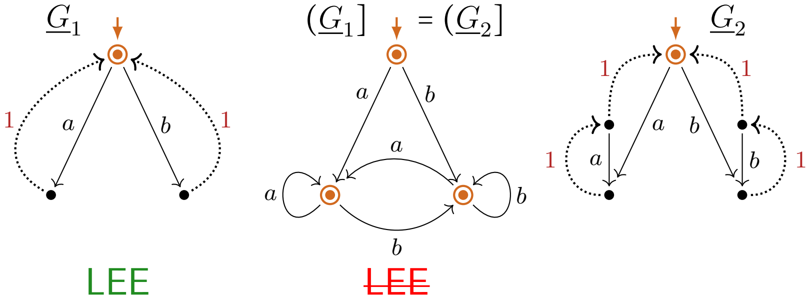
1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some 1-graph $\underline{G}$.



1-LEE

Definition

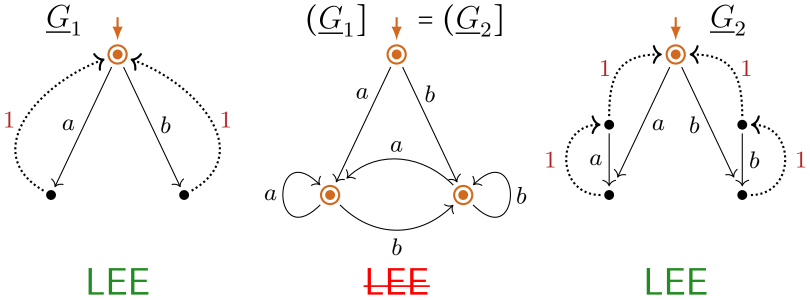
1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some 1-graph $\underline{G}$.



1-LEE

Definition

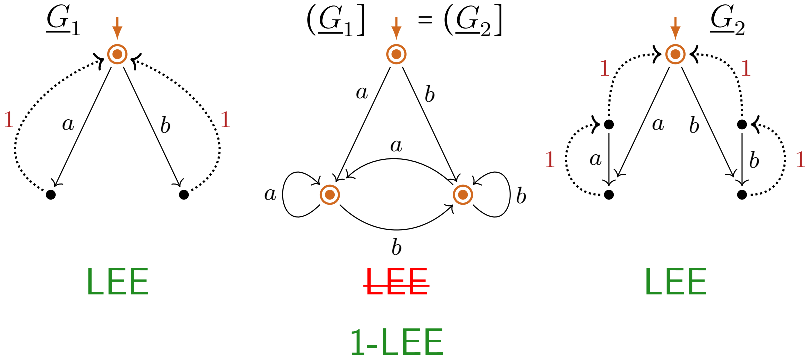
$1\text{-LEE}(G)$ holds for a graph G,
if $G = (\underline{G}]$ for some 1-graph $\underline{G}$.



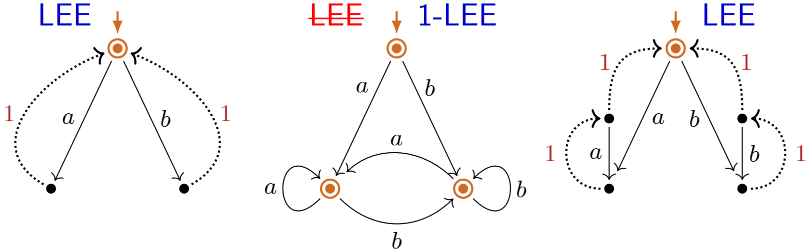
1-LEE

Definition

1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some 1-graph $\underline{G}$.



1-LEE holds for process interpretations



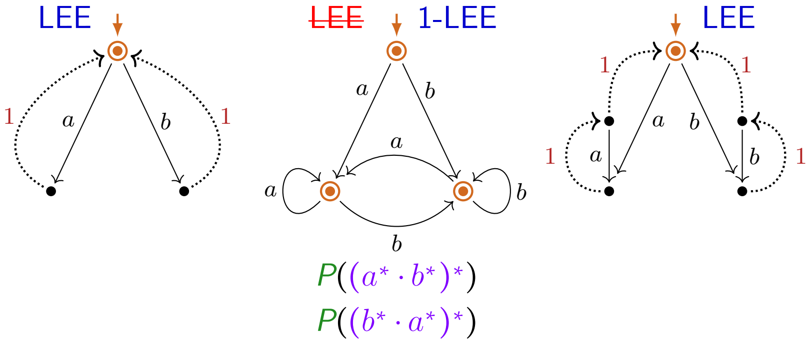
$P((a^* \cdot b^*)^*)$

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

1-LEE holds for process interpretations

Lemma

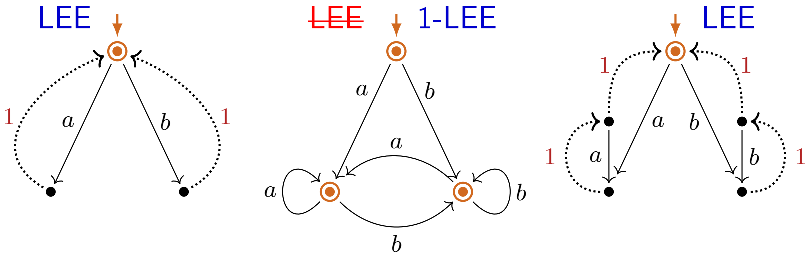
There is a 1-graph interpretation $\underline{P}$ of reg. expression e as 1-graphs $\underline{P}(e)$ such that for all $e \in RExp$: (i): LEE($\underline{P}(e)$), (ii): $(\underline{P}(e)] = P(e)$.



1-LEE holds for process interpretations

Lemma

There is a 1-graph interpretation $\underline{P}$ of reg. expression e as 1-graphs $\underline{P}(e)$ such that for all $e \in \mathit{RExp}$: (i): LEE($\underline{P}(e)$), (ii): $(\underline{P}(e)] = P(e)$.



$P((a^* \cdot b^*)^*)$ $\underline{P}((a^* \cdot b^*)^*)$

$P((b^* \cdot a^*)^*)$ $\underline{P}((b^* \cdot a^*)^*)$

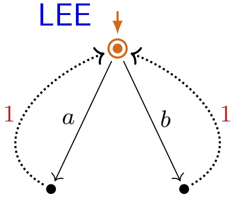
1-LEE holds for process interpretations

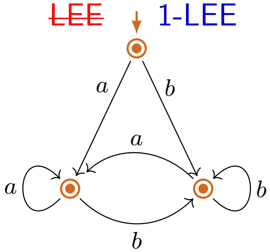
Lemma

There is a 1-graph interpretation $\underline{P}$ of reg. expression e as 1-graphs $\underline{P}(e)$ such that for all $e \in RExp$: (i): LEE($\underline{P}(e)$), (ii): $(\underline{P}(e)] = P(e)$.

Theorem

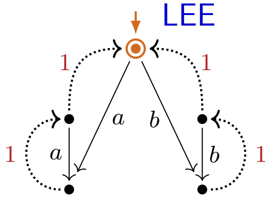
1-LEE($P(e)$) holds for all regular expressions e.





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

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



$\underline{P}((a^* \cdot b^*)^*)$

$\underline{P}((b^* \cdot a^*)^*)$

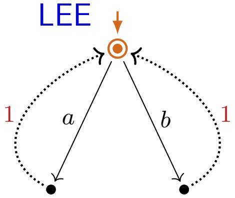
1-LEE holds for process interpretations

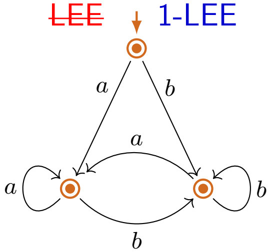
Lemma

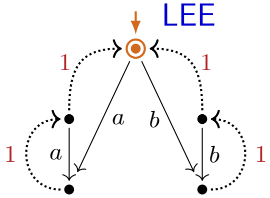
There is a 1-graph interpretation $\underline{P}$ of reg. expression e as 1-graphs $\underline{P}(e)$ such that for all $e \in REExp$: (i): LEE($\underline{P}(e)$), (ii): $(\underline{P}(e)] = P(e)$.

Theorem

1-LEE($P(e)$) holds for all regular expressions e.







1-LEE($P((a^* \cdot b^*)^*)$)

1-LEE($P((b^* \cdot a^*)^*)$)

$\underline{P}((a^* \cdot b^*)^*)$

$\underline{P}((b^* \cdot a^*)^*)$

Image of P is **not** closed under bisimulation collapse

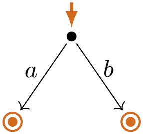
$P(uf)$

$P(uf)$

$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

Image of P is **not** closed under bisimulation collapse

$P(uf)$



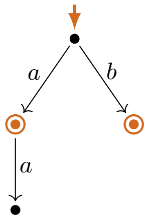
$P(uf)$

$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

$$1 \cdot uf_a \Downarrow \xleftarrow{a} uf \xrightarrow{b} \Downarrow 1 \cdot uf_b$$

Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$

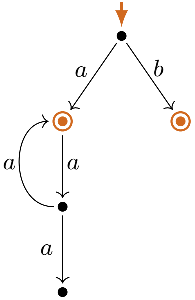
$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

$$uf \xrightarrow{a} 1 \cdot uf_a \Downarrow \qquad uf \xrightarrow{b} {\Downarrow} 1 \cdot uf_b$$

$$1 \cdot uf_a \xrightarrow{a} (1 \cdot (a + a \cdot 0)) \cdot uf_a$$

Image of P is **not** closed under bisimulation collapse

$P(uf)$

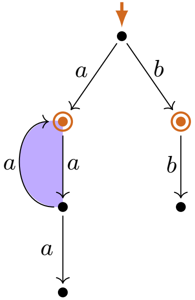


$P(uf)$

$$
\begin{tikzcd}[column sep=small]
& uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b} \arrow[dl, "a"'] \arrow[dr, "b"] & \\
1 \cdot uf_a \Downarrow \arrow[d, "a"] & & \Downarrow 1 \cdot uf_b \\
(1 \cdot (a + a \cdot 0)) \cdot uf_a \arrow[u, bend left, "a"] \arrow[d, "a"] & & \\
(1 \cdot 0) \cdot uf_a & &
\end{tikzcd}
$$

Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$

$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^{*}}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^{*}}^{uf_b}$$

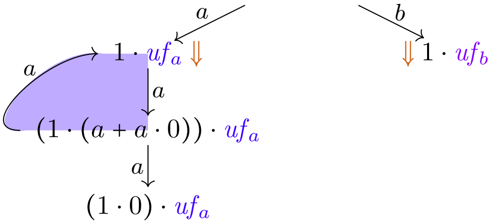
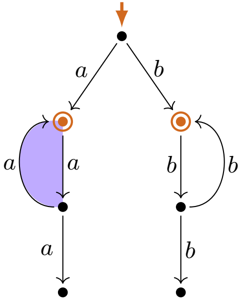


Image of *P* is **not** closed under bisimulation collapse

$P(uf)$



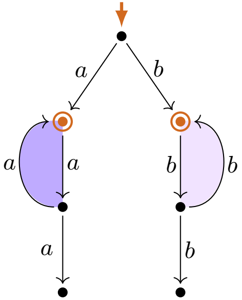
$P(uf)$

$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

$$\begin{tikzcd}
 & uf \arrow[dl, "a"'] \arrow[dr, "b"] & \\
1 \cdot uf_a \Downarrow \arrow[d, "a"] & & \Downarrow 1 \cdot uf_b \arrow[d, "b"] \\
(1 \cdot (a + a \cdot 0)) \cdot uf_a \arrow[u, bend left, "a"] \arrow[d, "a"] & & (1 \cdot (b + b \cdot 0)) \cdot uf_b \\
(1 \cdot 0) \cdot uf_a & &
\end{tikzcd}$$

Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$

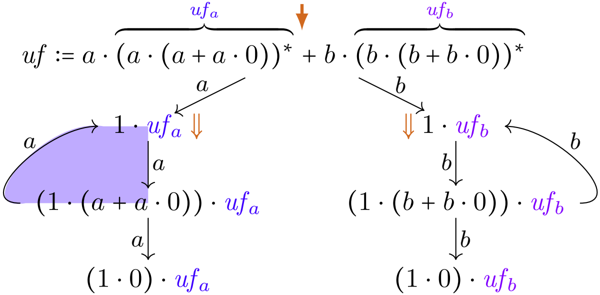
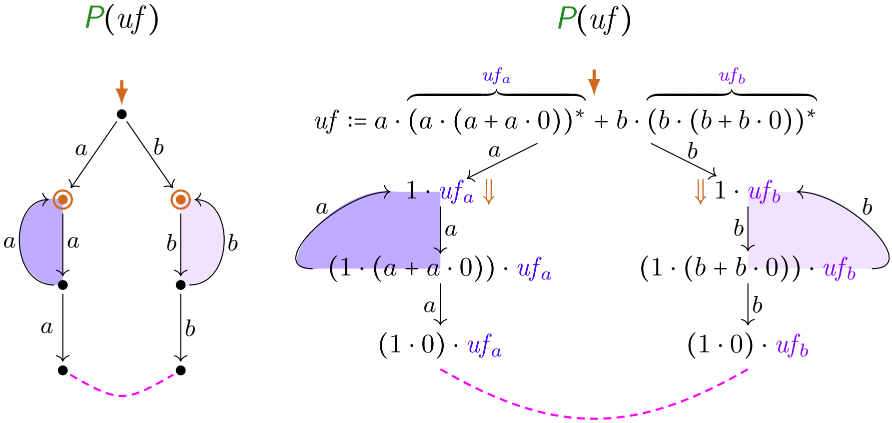


Image of P is **not** closed under bisimulation collapse



Compact process interpretation $P^{\bullet}$

Definition (Transition system specification $\mathcal{T}$)

$$\frac{}{1{\Downarrow}} \qquad \frac{e_i{\Downarrow}}{(e_1 + e_2){\Downarrow}}\;(i \in \{1,2\}) \qquad \frac{e_1{\Downarrow} \quad e_2{\Downarrow}}{(e_1 \cdot e_2){\Downarrow}} \qquad \frac{}{(e^*){\Downarrow}}$$

$$\frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} e_i'}{e_1 + e_2 \xrightarrow{a} e_i'}\;(i \in \{1,2\})$$

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \qquad \frac{e_1{\Downarrow} \quad e_2 \xrightarrow{a} e_2'}{e_1 \cdot e_2 \xrightarrow{a} e_2'} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Compact process interpretation $P^{\bullet}$

Definition (Transition system specification $\mathcal{T}$)

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2}$$

$$\frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Compact process interpretation $P^{\bullet}$

Definition (Transition system specification $\mathcal{T}^{\bullet}$, changed rules w.r.t. $\mathcal{T}$)

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \text{ (if } e_1' \text{ is normed)}$$

$$\frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*} \text{ (if } e' \text{ is normed)}$$

Compact process interpretation $P^{\bullet}$

Definition (Transition system specification $\mathcal{T}^{\bullet}$, changed rules w.r.t. $\mathcal{T}$)

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \text{ (if } e_1' \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1'} \text{ (if } e_1' \text{ is not normed)}$$

$$\frac{e \xrightarrow{a} e'}{e^{*} \xrightarrow{a} e' \cdot e^{*}} \text{ (if } e' \text{ is normed)} \qquad \frac{e \xrightarrow{a} e'}{e^{*} \xrightarrow{a} e'} \text{ (if } e' \text{ is not normed)}$$

Compact process interpretation $P^{\bullet}$

Definition (Transition system specification $\mathcal{T}^{\bullet}$, changed rules w.r.t. $\mathcal{T}$)

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \text{ (if } e_1' \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1'} \text{ (if } e_1' \text{ is not normed)}$$

$$\frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*} \text{ (if } e' \text{ is normed)} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e'} \text{ (if } e' \text{ is not normed)}$$

Definition

The compact process (graph) interpretation $P^{\bullet}(e)$ of a reg. expr's e:

$P^{\bullet}(e) :=$ labeled transition graph generated by e by derivations in $\mathcal{T}^{\bullet}$.

Compact process interpretation $P^{\bullet}$

Definition (Transition system specification $\mathcal{T}^{\bullet}$, changed rules w.r.t. $\mathcal{T}$)

$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \text{ (if } e_1' \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1'} \text{ (if } e_1' \text{ is not normed)}$$

$$\frac{e \xrightarrow{a} e'}{e^{*} \xrightarrow{a} e' \cdot e^{*}} \text{ (if } e' \text{ is normed)} \qquad \frac{e \xrightarrow{a} e'}{e^{*} \xrightarrow{a} e'} \text{ (if } e' \text{ is not normed)}$$

Definition

The compact process (graph) interpretation $P^{\bullet}(e)$ of a reg. expr's e:

$P^{\bullet}(e)$:= labeled transition graph generated by e by derivations in $\mathcal{T}^{\bullet}$.

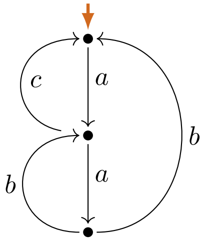
Lemma ($P^{\bullet}$ increases sharing; $P^{\bullet}$, P have same bisimulation semantics)

(i) $P(e) \mathrel{\underline{\to}} P^{\bullet}(e)$ *for all regular expressions* e.

(ii) *(*G *is* $[\![\cdot]\!]_{P^{\bullet}}$*-expressible* $\iff$ G *is* $[\![\cdot]\!]_{P}$*-expressible) for all graphs* G.

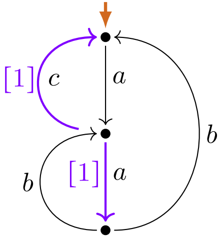
Refined extraction expression (example)

G_4



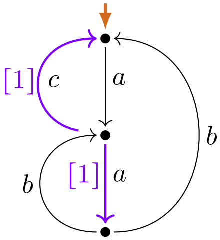
Refined extraction expression (example)

$\widehat{G_4}$



Refined extraction expression (example)

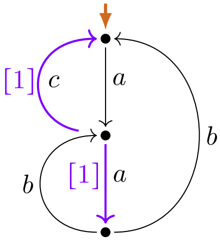
$\widehat{G_4}$



$$(1 \cdot (\qquad\qquad\qquad)^*) \cdot 0$$

Refined extraction expression (example)

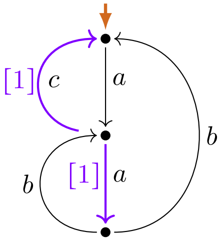
$\widehat{G_4}$



$$(1 \cdot (\qquad \overset{a}{\downarrow} \qquad)^*) \cdot 0$$

Refined extraction expression (example)

$\widehat{G_4}$



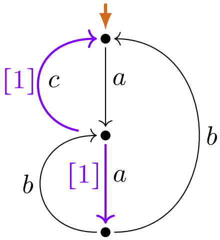
$$((1 \cdot a) \cdot (\qquad\qquad)^{*}) \cdot 0$$

$$\Big\downarrow a$$

$$(1 \cdot (\qquad\qquad)^{*}) \cdot 0$$

Refined extraction expression (example)

$\widehat{G_4}$



$$((1 \cdot a) \cdot (\qquad)^*) \cdot 0$$

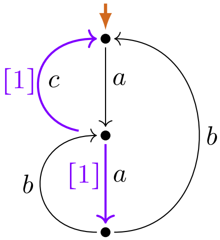
a

$$(1 \cdot (\qquad)^*) \cdot 0$$

b $\quad$ b

Refined extraction expression (example)

$\widehat{G_4}$



$((1 \cdot a) \cdot (\qquad)^{*}) \cdot 0$

a

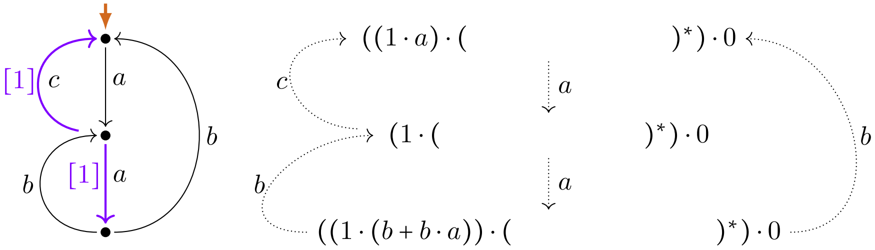
$(1 \cdot (\qquad)^{*}) \cdot 0$

b $\qquad$ b

$((1 \cdot (b + b \cdot a)) \cdot (\qquad)^{*}) \cdot 0$

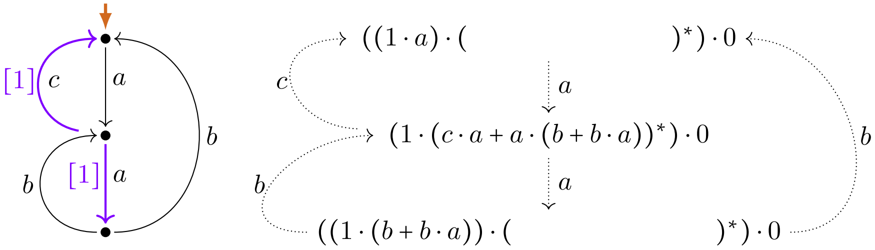
Refined extraction expression (example)

$\widehat{G_4}$



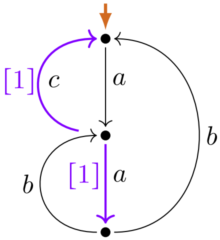
Refined extraction expression (example)

$\widehat{G_4}$



Refined extraction expression (example)

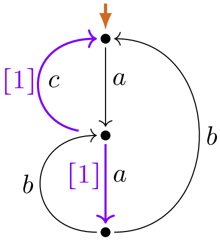
$\widehat{G_4}$

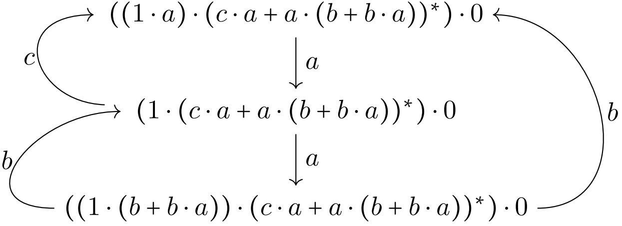


$$
\begin{tikzcd}
((1\cdot a)\cdot(c\cdot a + a\cdot(b+b\cdot a))^*)\cdot 0 \arrow[d, dotted, "a"] \\
(1\cdot(c\cdot a + a\cdot(b+b\cdot a))^*)\cdot 0 \arrow[d, dotted, "a"] \arrow[u, dotted, bend left=60, "c"] \\
((1\cdot(b+b\cdot a))\cdot(c\cdot a + a\cdot(b+b\cdot a))^*)\cdot 0 \arrow[u, dotted, bend left=60, "b"] \arrow[uu, dotted, bend right=80, "b"']
\end{tikzcd}
$$

Refined extraction expression (example)

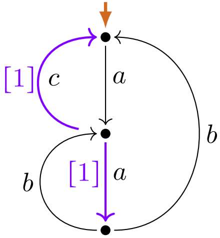
$\widehat{G_4}$





Refined extraction expression (example)

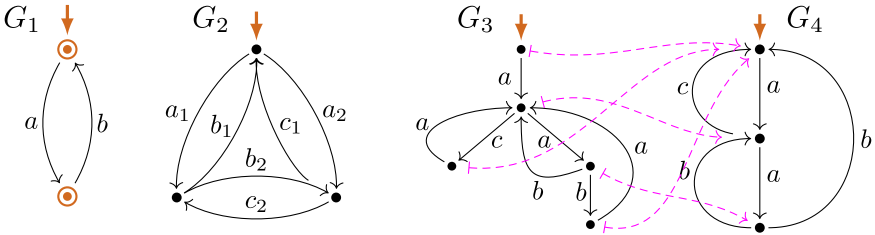
$\widehat{G_4}$



$$P^{\bullet}(uf) = P(uf) \simeq G_4$$

$$\begin{tikzcd}
\overbrace{((1\cdot a)\cdot(c\cdot a + a\cdot(b + b\cdot a))^{*})\cdot 0}^{uf} \arrow[d, "a"] \\
(1\cdot(c\cdot a + a\cdot(b + b\cdot a))^{*})\cdot 0 \arrow[d, "a"] \arrow[u, bend left, "c"] \\
((1\cdot(b + b\cdot a))\cdot(c\cdot a + a\cdot(b + b\cdot a))^{*})\cdot 0 \arrow[u, bend left, "b"] \arrow[uu, bend right, "b"']
\end{tikzcd}$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples revisited)



not P-expressible
not $[\![\cdot]\!]_P$-expressible

P-/$P^{\bullet}$-expressible
$[\![\cdot]\!]_P$-expressible

$P^{\bullet}$-expressible
$[\![\cdot]\!]_P$-expressible

Characterizations of the image of $P^{\bullet}$

LEE $\triangleq$ image of $P^{\bullet}|_{RExp^{(*/\cancel{1})}}$

Theorem

For every process graph G *TFAE:*

(i) LEE(G).

LEE $\triangleq$ image of $P^{\bullet}|_{RExp^{(*/\cancel{1})}}$

Theorem

For every process graph G TFAE:

(i) LEE(G).

(ii) *G is $P^{\bullet}$-expressible by an $(*/\cancel{1})$ regular expression (i.e. $G \simeq P^{\bullet}(e)$ for some $e \in RExp^{(*/\cancel{1})}$).*

LEE $\triangleq$ image of $P^{\bullet}|_{RExp^{(*/\not{1})}}$

Theorem

For every process graph G TFAE:

(i) $\mathsf{LEE}(G)$.

(ii) G is $P^{\bullet}$-expressible by an $(*/\not{1})$ regular expression
(i.e. $G \simeq P^{\bullet}(e)$ for some $e \in RExp^{(*/\not{1})}$).

(iii) G is isomorphic to a graph in the image of $P^{\bullet}$ on $(*/\not{1})$ reg. expr's
(i.e. $G \simeq G'$ for some $G' \in im(P^{\bullet}|_{RExp^{(*/\not{1})}})$).

1-LEE $\hat{=}$ image of $P^{\bullet}$

Theorem

For every process graph G TFAE:

(i) 1-LEE(G)

(i.e. $G = (\underline{G}]$ for some 1-transition-process-graph $\underline{G}$ with LEE($\underline{G}$)*).*

1-LEE $\triangleq$ image of $P^{\bullet}$

Theorem

For every process graph G TFAE:

(i) 1-LEE(G)
(i.e. $G = (\underline{G}]$ for some 1-transition-process-graph $\underline{G}$ with LEE($\underline{G}$)*).*

(ii) G *is* $P^{\bullet}$*-expressible by a regular expression*
(i.e. $G \simeq P^{\bullet}(e)$ for some $e \in \mathit{RExp}$).

1-LEE $\triangleq$ image of $P^\bullet$

Theorem

For every process graph G TFAE:

(i) 1-LEE(G)
(i.e. $G = (\underline{G}]$ for some 1-transition-process-graph $\underline{G}$ with LEE($\underline{G}$)*).*

(ii) G is $P^\bullet$-expressible by a regular expression
(i.e. $G \simeq P^\bullet(e)$ for some $e \in \mathit{RExp}$).

(iii) G is isomorphic to a graph in the image of $P^\bullet$
(i.e. $G \simeq G'$ for some $G' \in \mathit{im}(P^\bullet)$).

Summary

- process interpretation P / semantics $[\![\cdot]\!]_P$ of regular expressions
  - expressibility and completeness questions
- loop existence and elimination (LEE)
  - loop elimination rewrite system can be completed
  - interpretation/extraction correspondences with $(*/\cancel{1})$ reg. expr.s
  - LEE-witnesses: labelings of graphs with LEE
  - stepwise LEE-preserving bisimulation collapse
- 1-LEE = sharing via 1-transitions facilitates LEE
  - interpretation/extraction correspondences with all regular expressions
  - not preserved under bisim. collapse (approximation possible)
- Characterizations of the image of $P^{\bullet}$ (refinement of P):
  - LEE $\triangleq$ image of $P^{\bullet}|_{RExp^{(*/\cancel{1})}}$ $\supsetneqq$ image of $P|_{RExp^{(*/\cancel{1})}}$
  - 1-LEE $\triangleq$ image of $P^{\bullet}$ $\supsetneqq$ image of P
- outlook on work-to-do

Summary and outlook

- 1-free/under-star-1-free $(*/\cancel{1})$ reg. expr'ss defined (also) with unary star
- image of $(*/\cancel{1})$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse

Summary and outlook

- 1-free/under-star-1-free $(*/\cancel{1})$ reg. expr'ss defined (also) with unary star
- image of $(*/\cancel{1})$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse
- compact process interpretation $P^{\bullet}$
- refined expression extraction from process graphs with LEE
- image of $(*/\cancel{1})$ reg. expr's under $P^{\bullet}$ **is closed** under collapse

Summary and outlook

- 1-free/under-star-1-free $(*/\cancel{1})$ reg. expr'ss defined (also) with unary star
- image of $(*/\cancel{1})$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse
- compact process interpretation $P^{\bullet}$
- refined expression extraction from process graphs with LEE
- image of $(*/\cancel{1})$ reg. expr's under $P^{\bullet}$ **is closed** under collapse
- A finite process graph G is $[\![\cdot]\!]_P$-expressible by a $(*/\cancel{1})$ regular expression
  $\iff$ the bisimulation collapse of G satisfies LEE (G/Fokkink 2020).

Summary and outlook

- 1-free/under-star-1-free $(*/\mathbb{1})$ reg. expr'ss defined (also) with unary star
- image of $(*/\mathbb{1})$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse
- compact process interpretation $P^\bullet$
- refined expression extraction from process graphs with LEE
- image of $(*/\mathbb{1})$ reg. expr's under $P^\bullet$ **is closed** under collapse
- A finite process graph G is $[\![\cdot]\!]_P$-expressible by a $(*/\mathbb{1})$ regular expression
  $\Longleftrightarrow$ the bisim. collapse of G is $P^\bullet$-expressible by a $(*/\mathbb{1})$ reg. expr..

Summary and outlook

- 1-free/under-star-1-free $(*/\underline{1})$ reg. expr'ss defined (also) with unary star
- image of $(*/\underline{1})$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse
- compact process interpretation $P^{\bullet}$
- refined expression extraction from process graphs with LEE
- image of $(*/\underline{1})$ reg. expr's under $P^{\bullet}$ **is closed** under collapse
- A finite process graph G is $[\![\cdot]\!]_P$-expressible by a $(*/\underline{1})$ regular expression
  $\iff$ the bisim. collapse of G is $P^{\bullet}$-expressible by a $(*/\underline{1})$ reg. expr..

Outlook on an extension:

- image of $(*/\underline{1})$ reg. expr's under $P^{\bullet}$ $=$ finite process graphs with LEE.

Summary and outlook

- 1-free/under-star-1-free $(*/1)$ reg. expr'ss defined (also) with unary star
- image of $(*/1)$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse
- compact process interpretation $P^{\bullet}$
- refined expression extraction from process graphs with LEE
- image of $(*/1)$ reg. expr's under $P^{\bullet}$ **is closed** under collapse
- A finite process graph G is $[\![\cdot]\!]_P$-expressible by a $(*/1)$ regular expression
  $\Longleftrightarrow$ the bisim. collapse of G is $P^{\bullet}$-expressible by a $(*/1)$ reg. expr..

Outlook on an extension:

- image of $(*/1)$ reg. expr's under $P^{\bullet}$ $=$ finite process graphs with LEE.

  A finite process graph G is $P^{\bullet}$-expressible by a $(*/1)$ regular expression
  $\Longleftrightarrow$ G satisfies LEE.

Summary and outlook

- 1-free/under-star-1-free $(*/\underline{1})$ reg. expr'ss defined (also) with unary star
- image of $(*/\underline{1})$ regular expressions under the process interpretation P
  is **not** closed under bisimulation collapse
- compact process interpretation $P^\bullet$
- refined expression extraction from process graphs with LEE
- image of $(*/\underline{1})$ reg. expr's under $P^\bullet$ **is closed** under collapse
- A finite process graph G is $[\![\cdot]\!]_P$-expressible by a $(*/\underline{1})$ regular expression
  $\iff$ the bisimulation collapse of G satisfies LEE (G/Fokkink 2020).

Outlook on an extension:

- image of $(*/\underline{1})$ reg. expr's under $P^\bullet$ $=$ finite process graphs with LEE.

  A finite process graph G is $P^\bullet$-expressible by a $(*/\underline{1})$ regular expression
  $\iff$ G satisfies LEE.

Resources

- Slides/abstract on `clegra.github.io`
  - slides: `.../lf/IFIP-1_6-2024.pdf`
  - abstract: `.../lf/abstract-IFIP-1_6-2024.pdf`

- CG, Wan Fokkink: A Complete Proof System for 1-Free Regular Expressions Modulo Bisimilarity,
  - LICS 2020, arXiv:2004.12740, video on youtube.
- CG: Modeling Terms by Graphs with Structure Constraints,
  - TERMGRAPH 2018, EPTCS 288, arXiv:1902.02010.
- CG: The Image of the Process Interpretation of Regular Expressions is Not Closed under Bisimulation Collapse,
  - arXiv:2303.08553.
- CG: Milner's Proof System for Regular Expressions Modulo Bisimilarity is Complete,
  - LICS 2022, arXiv:2209.12188, poster.

Language semantics $[\![\cdot]\!]_L$ of reg. expr's *(Copi–Elgot–Wright, 1958)*

$\mathbf{0} \xmapsto{L}$ empty language $\varnothing$

$\mathbf{1} \xmapsto{L} \{\epsilon\}$ (ϵ the empty word)

$a \xmapsto{L} \{a\}$

Language semantics $[\![\cdot]\!]_L$ of reg. expr's *(Copi–Elgot–Wright, 1958)*

$\mathbf{0} \xmapsto{L}$ empty language $\varnothing$

$\mathbf{1} \xmapsto{L} \{\epsilon\}$ (ϵ the empty word)

$a \xmapsto{L} \{a\}$

$e_1 + e_2 \xmapsto{L}$ union of $L(e_1)$ and $L(e_2)$

$e_1 \cdot e_2 \xmapsto{L}$ element-wise concatenation of $L(e_1)$ and $L(e_2)$

$e^* \xmapsto{L}$ set of words formed by concatenating words in $L(e)$, and adding the empty word ϵ

Language semantics $[\![\cdot]\!]_L$ of reg. expr's *(Copi–Elgot–Wright, 1958)*

$\mathbf{0} \stackrel{L}{\longmapsto}$ empty language $\varnothing$

$\mathbf{1} \stackrel{L}{\longmapsto} \{\epsilon\}$ (ϵ the empty word)

$a \stackrel{L}{\longmapsto} \{a\}$

$e_1 + e_2 \stackrel{L}{\longmapsto}$ union of $L(e_1)$ and $L(e_2)$

$e_1 \cdot e_2 \stackrel{L}{\longmapsto}$ element-wise concatenation of $L(e_1)$ and $L(e_2)$

$e^* \stackrel{L}{\longmapsto}$ set of words formed by concatenating words in $L(e)$, and adding the empty word ϵ

$[\![e]\!]_L \; := \; L(e)$ (language defined by e)