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 $(*/\cancel{1})$)
- 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
- 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:

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

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

Regular Expressions

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

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

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

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

- 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:

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

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

- 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 (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 binary Kleene star:

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

Regular Expressions (1-free)

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

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

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

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

- 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)

Definition (for process interpretation)

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

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

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

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)*

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

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

$a \overset{P}{\longmapsto}$ atomic action a, then terminate

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

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

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

$a \stackrel{P}{\longmapsto}$ atomic action a, then terminate

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

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

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

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

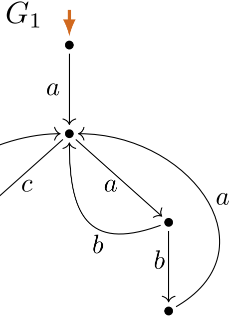
$e_1 + e_2 \xmapsto{P}$ (*choice*) execute $P(e_1)$ or $P(e_2)$

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

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

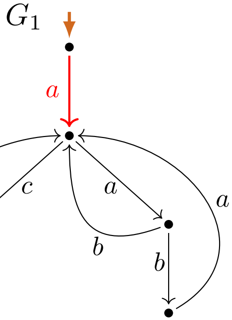
$[\![e]\!]_P := [P(e)]_{\underline{\leftrightarrow}}$ (bisimilarity equivalence class of process $P(e)$)

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



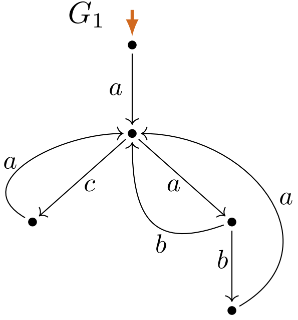
$$P\Big(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^{f} \Big)$$

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



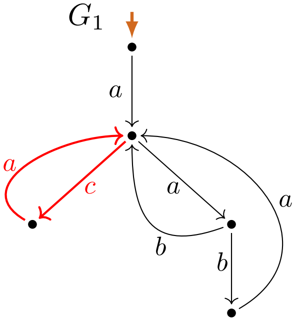
$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^*)\cdot 0}^{f}\Big)$$

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



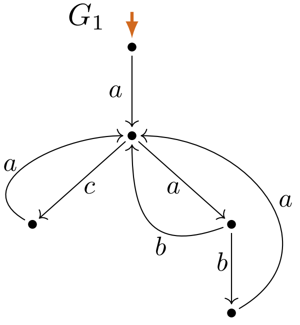
$$P\big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^*)\cdot 0}^{f}\big)$$

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



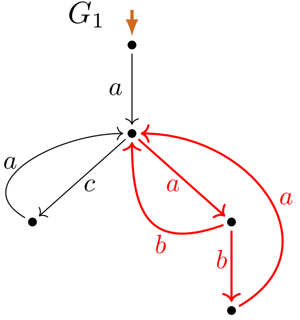
$$P\big(\,\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^{*})\cdot 0}^{f}\,\big)$$

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



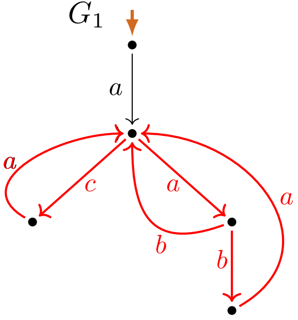
$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^*)\cdot 0}^{f}\Big)$$

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



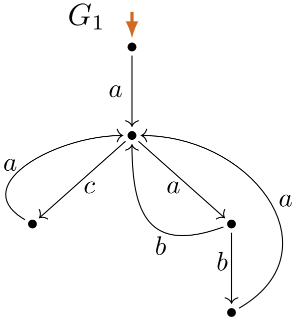
$$P\big(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^{f}\big)$$

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



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

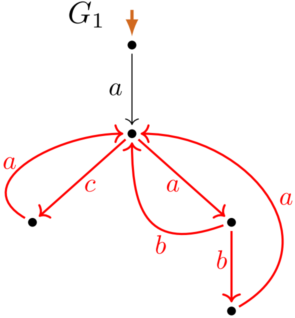
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



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

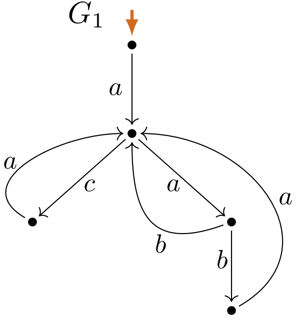
$$P\Big(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\circledast} 0 \Big)$$

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)

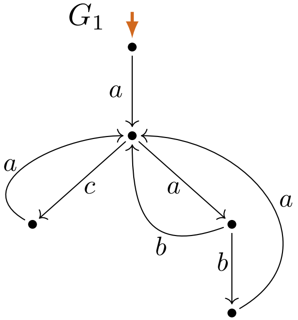


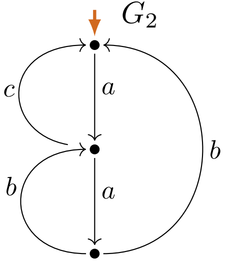
$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b + b\cdot a))^{*})\cdot 0}^{f}\Big)$$

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

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

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



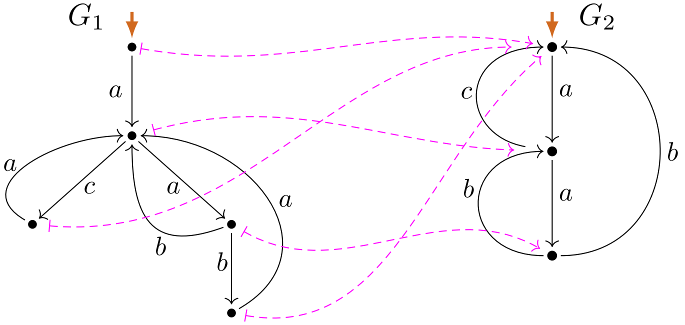


$$P\Big(\overbrace{(a\cdot(c\cdot a + a\cdot(b+b\cdot a))^{*})\cdot 0}^{f}\Big)$$

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

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

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

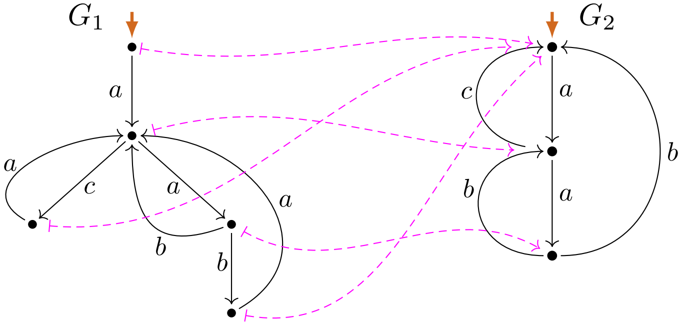


$$P\Big(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^{f}\Big)$$

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

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

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 $\mathcal{T}$)

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

Process interpretation P (formally)

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

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

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

Process interpretation P (formally)

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

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

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

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

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 $\mathcal{T}$)

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

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

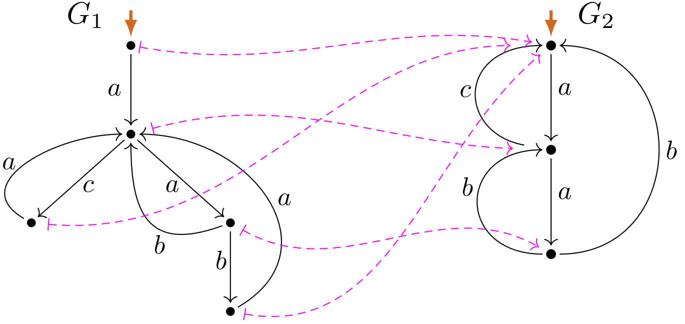
$$\frac{e_1 \xrightarrow{a} e_1'}{e_1 \cdot e_2 \xrightarrow{a} e_1' \cdot e_2} \qquad \frac{e_1{\Downarrow} \qquad e_2 \xrightarrow{a} e_2'}{e_1 \cdot e_2 \xrightarrow{a} e_2'} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

Definition

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

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

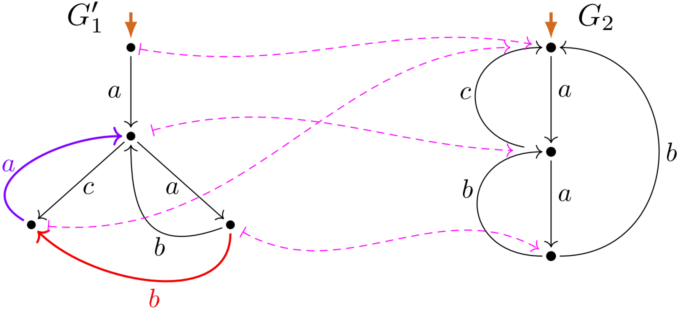
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



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

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

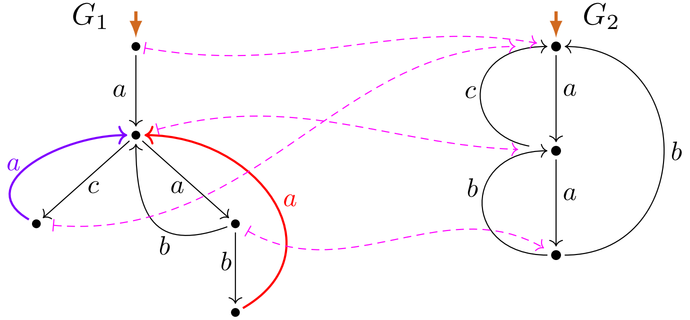
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, formally)



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

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

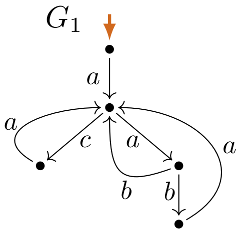
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, formally)

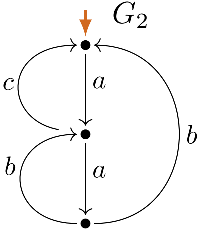


$$P\big(\overbrace{a \cdot ((c \cdot a + a \cdot (b + b \cdot (a + a))^*) \cdot 0)}^{f}\big)$$

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

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

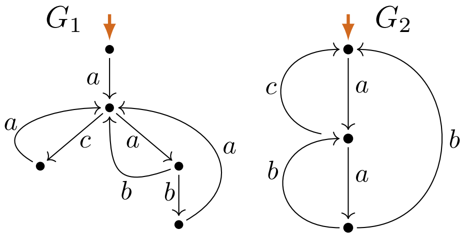




P-expressible

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

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



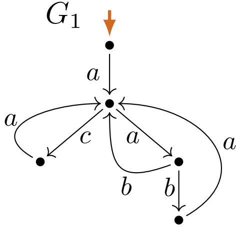
P-expressible

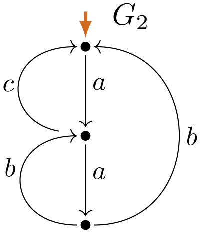
?

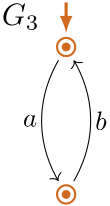
$[\![\cdot]\!]_P$-expressible

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

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





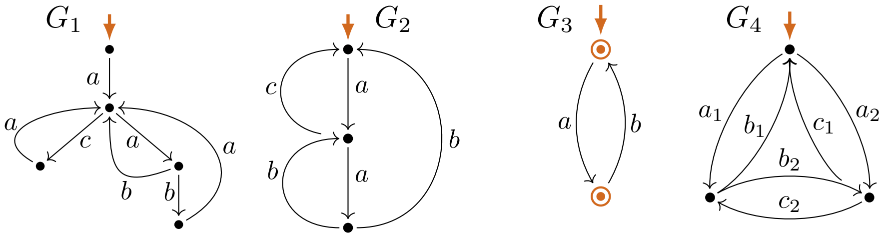


G_1: P-expressible, $[\![\cdot]\!]_P$-expressible

G_2: **?**, $[\![\cdot]\!]_P$-expressible

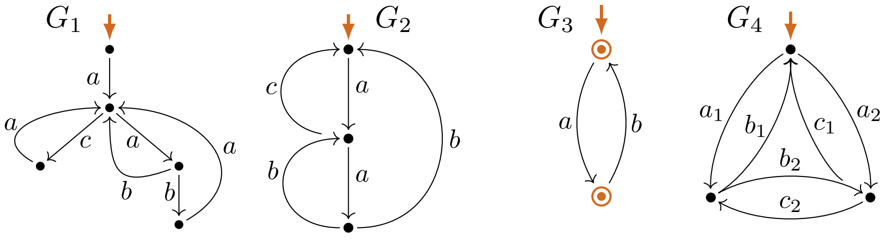
G_3: **not** P-expressible, **not** $[\![\cdot]\!]_P$-expressible

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



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

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

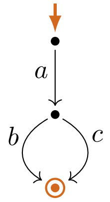


P-expressible	**?**	**not** P-expressible
$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	**not** $[\![\cdot]\!]_P$-expressible

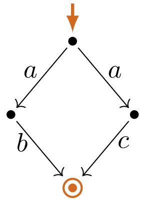
Q2: How can P-expressibility and $[\![\cdot]\!]_P$-expressibility be characterized?

Process semantics equality $=_{[\![\cdot]\!]_P}$

- Fewer identities hold for $=_{[\![\cdot]\!]_P}$ than for $=_{[\![\cdot]\!]_L}$: .



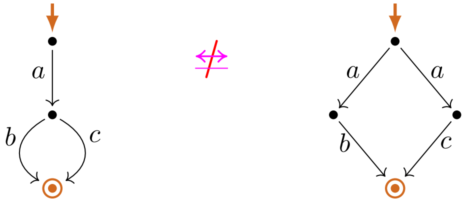
$P(a \cdot (b + c))$



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

Process semantics equality $=_{[\![\cdot]\!]_P}$

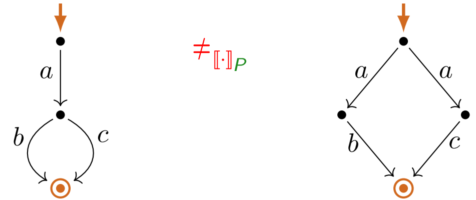
- Fewer identities hold for $=_{[\![\cdot]\!]_P}$ than for $=_{[\![\cdot]\!]_L}$: .



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

Process semantics equality $=_{[\![\cdot]\!]_P}$

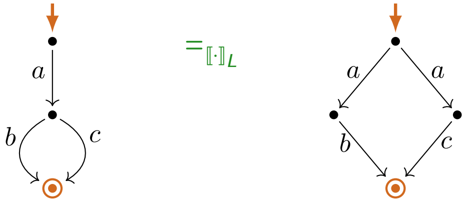
- Fewer identities hold for $=_{[\![\cdot]\!]_P}$ than for $=_{[\![\cdot]\!]_L}$: .



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

Process semantics equality $=_{[\![\cdot]\!]_P}$

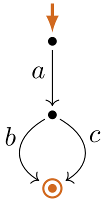
- Fewer identities hold for $=_{[\![\cdot]\!]_P}$ than for $=_{[\![\cdot]\!]_L}$: .



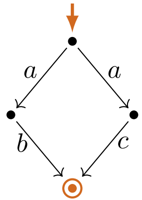
$$a \cdot (b + c) \quad =_{[\![\cdot]\!]_L} \quad a \cdot b + a \cdot c$$

Process semantics equality $=_{[\![\cdot]\!]_P}$

- 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 proof system Mil

Axioms :

(A1)	$e + (f + g) = (e + f) + g$	(A7)	$e = 1 \cdot e$
(A2)	$e + 0 = e$	(A8)	$e = e \cdot 1$
(A3)	$e + f = f + e$	(A9)	$0 = 0 \cdot e$
(A4)	$e + e = e$	(A10)	$e^* = 1 + e \cdot e^*$
(A5)	$e \cdot (f \cdot g) = (e \cdot f) \cdot g$	(A11)	$e^* = (1 + e)^*$
(A6)	$(e + f) \cdot g = e \cdot g + f \cdot g$		

But: $e \cdot (f + g) \neq e \cdot f + e \cdot g$ But: $e \cdot 0 \neq 0$

Inference rules : rules of equational logic *plus*

$$\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \mathsf{RSP}^* \quad \text{(if } f \text{ does not terminate immediately)}$$

Milner's Question (Q1)

Is Mil complete with respect to $=_{[\![\cdot]\!]_P}$? (Does $e =_{[\![\cdot]\!]_P} f \implies e =_{\mathsf{Mil}} f$ hold?)

Milner's questions

(Q1) Complete axiomatization:

Is the proof system Mil complete for $=_{[\![\cdot]\!]_P}$?

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

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

Milner's questions

(Q1) Complete axiomatization:
Is the proof system Mil *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

(Q1) Complete axiomatization:

Is the proof system Mil 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

(Q1) Complete axiomatization:

Is the proof system Mil *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

(Q1) Complete axiomatization:

Is the proof system Mil *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

Question (Q2) specialized

$(\mathrm{Q2})_{\mathbf{0}}$ P-Expressibility and P-$(*/\mathit{1})$-Expressibility:

What structural property characterizes:

- *process graphs that are P-expressible ?*
  (... that are in the image of P?)
- *process graphs that are P-expressible by $(*/\mathit{1})$ regular expressions?*
  (... that are in the image of $(*/\mathit{1})$ expressions under P?)

Loop Existence and Elimination (LEE)

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)

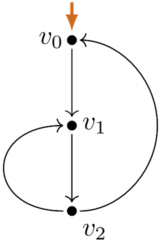
Definition

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)

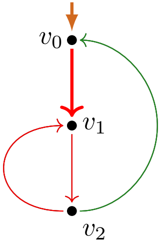
Definition

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)

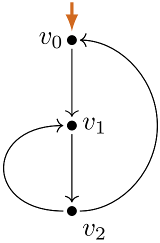
Definition

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)

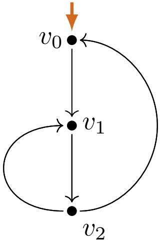
Definition

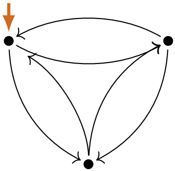
A process graph is a loop graph if:

(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)

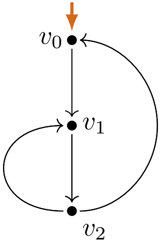
Definition

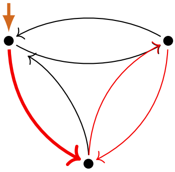
A process graph is a loop graph if:

(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)

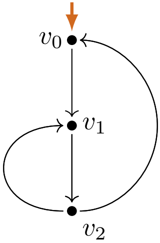
Definition

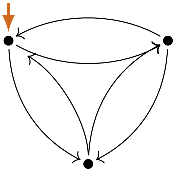
A process graph is a loop graph if:

(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)

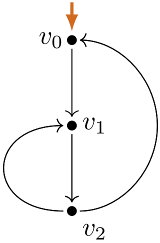
Definition

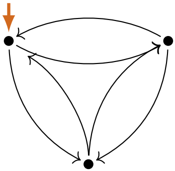
A process graph is a loop graph if:

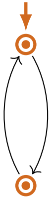
(L1) There is an infinite path from the start vertex.

(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)

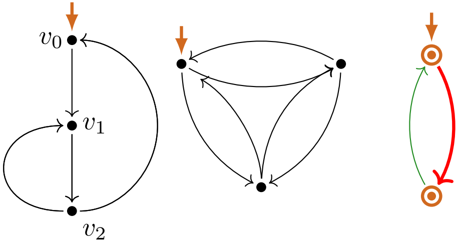
Definition

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)

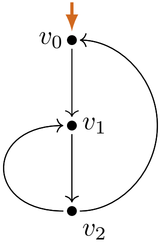
Definition

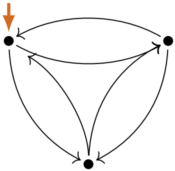
A process graph is a loop graph if:

(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)

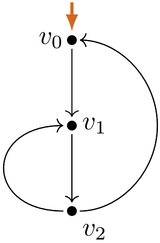
Definition

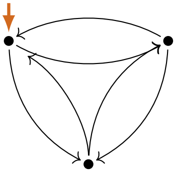
A process graph is a loop graph if:

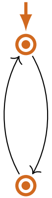
(L1) There is an infinite path from the start vertex.

(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)

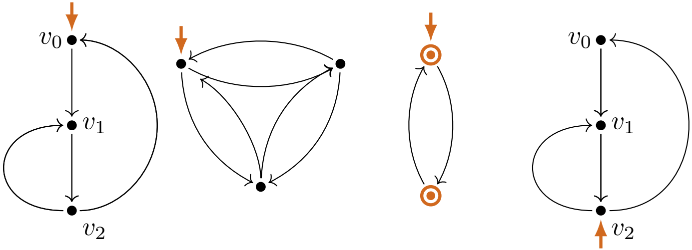
Definition

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)

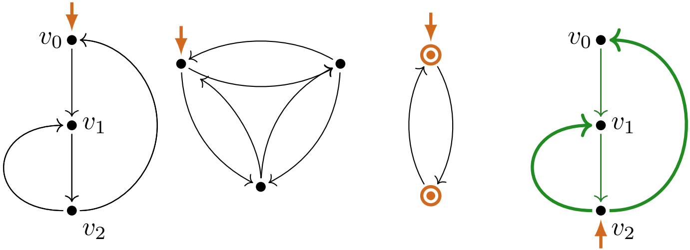
Definition

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)

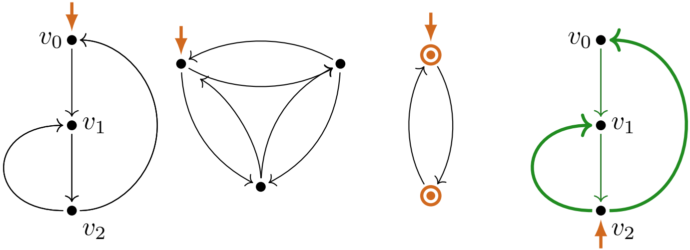
Definition

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)

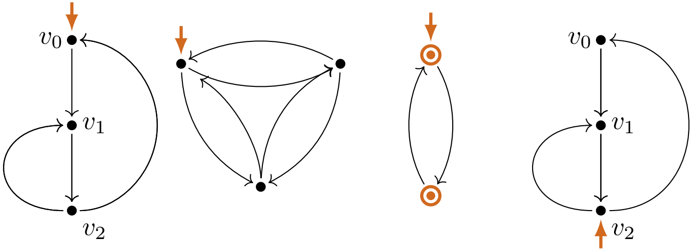
Definition

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)

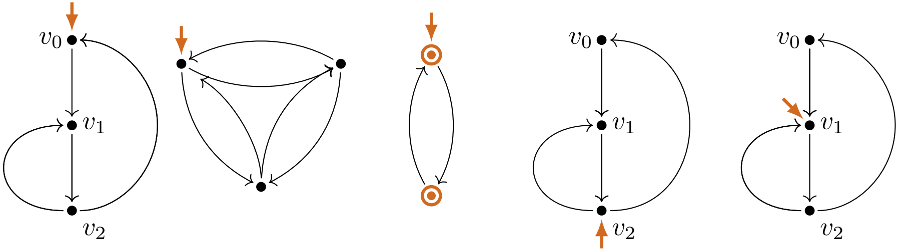
Definition

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)

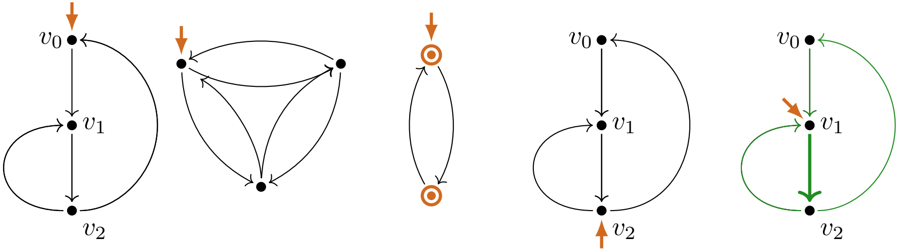
Definition

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)

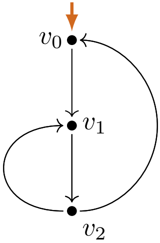
Definition

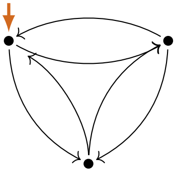
A process graph is a loop graph if:

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

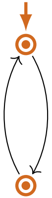
(L2) Every infinite path from the start vertex returns to it.

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

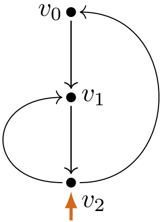




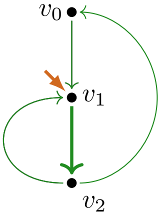
(L1),~~(L2)~~



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



loop chart



loop chart

Loop graphs (interpretations of innermost iterations without 1)

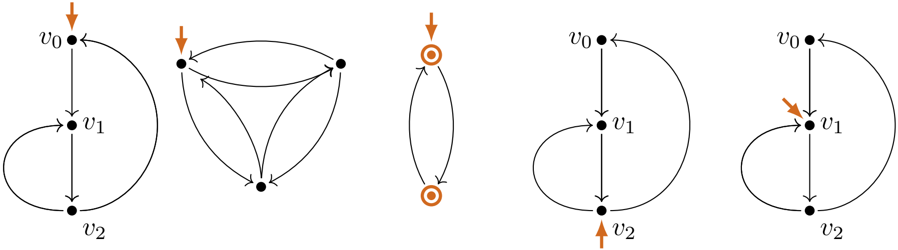
Definition

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)

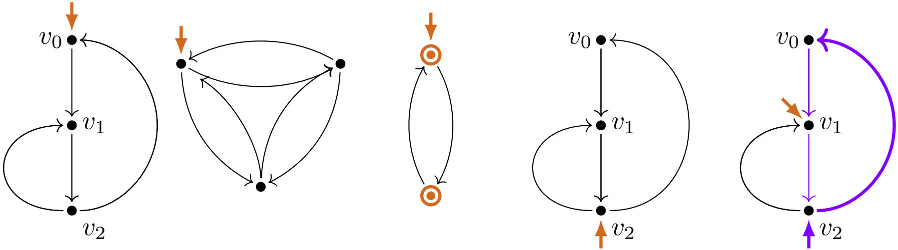
Definition

A process graph is a loop graph if:

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

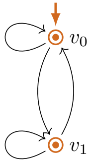
(L2) Every infinite path from the start vertex returns to it.

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

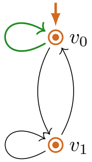


(L1),~~(L2)~~ (L1),(L2),~~(L3)~~ loop chart loop subchart

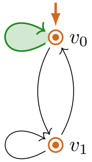
Loop elimination



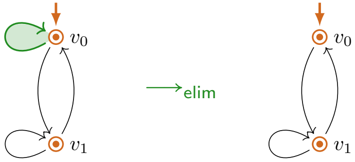
Loop elimination



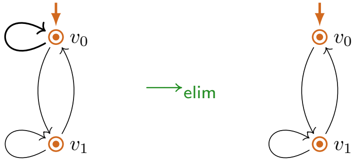
Loop elimination



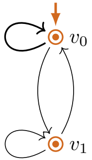
Loop elimination



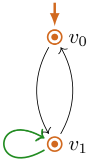
Loop elimination



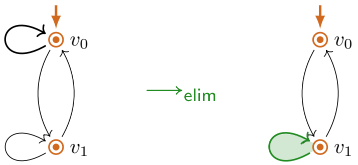
Loop elimination



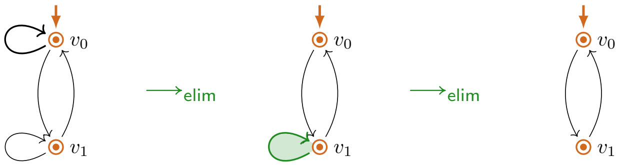
$\longrightarrow_{\text{elim}}$



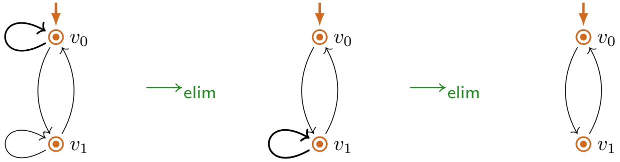
Loop elimination



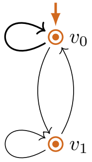
Loop elimination



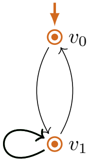
Loop elimination



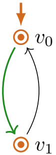
Loop elimination



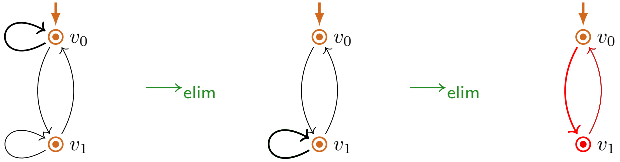
$\longrightarrow_{\text{elim}}$



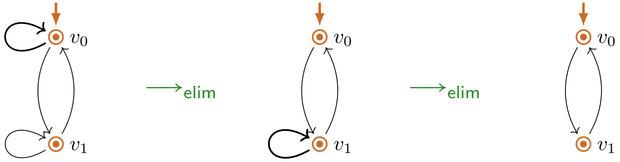
$\longrightarrow_{\text{elim}}$



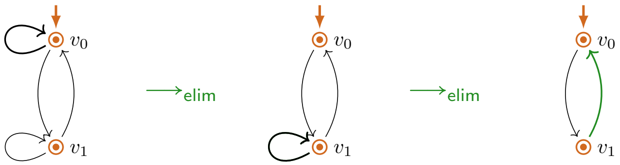
Loop elimination



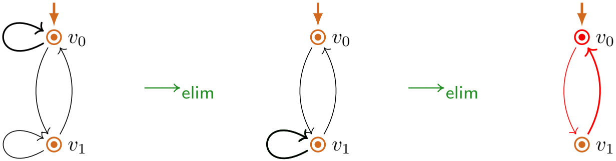
Loop elimination



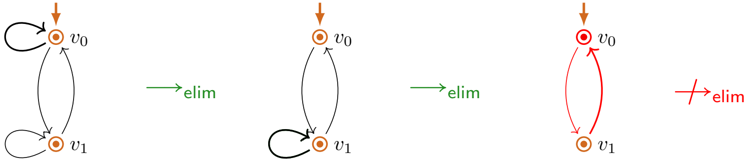
Loop elimination



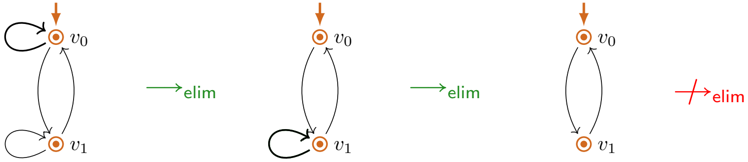
Loop elimination



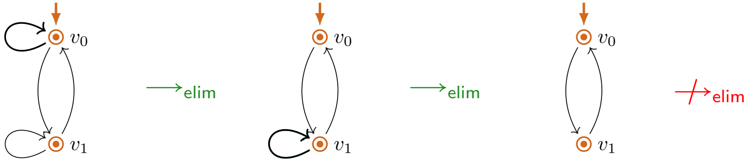
Loop elimination

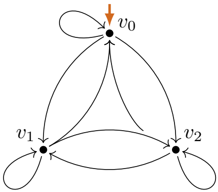


Loop elimination

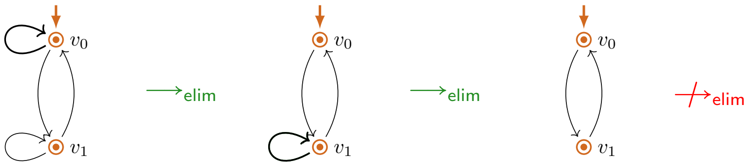


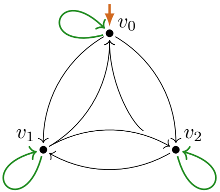
Loop elimination



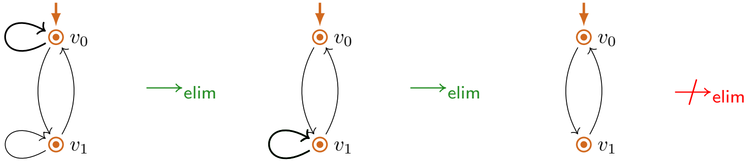


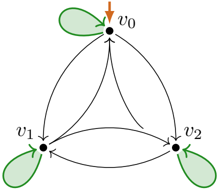
Loop elimination



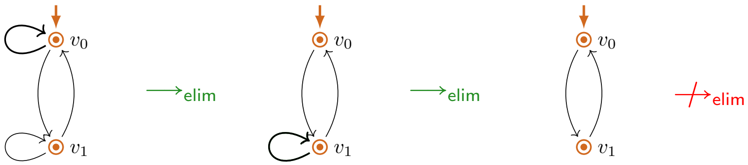


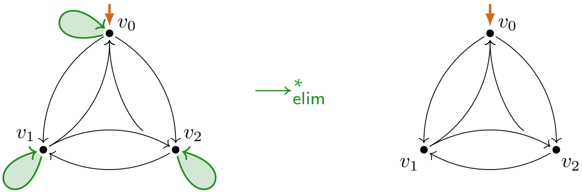
Loop elimination



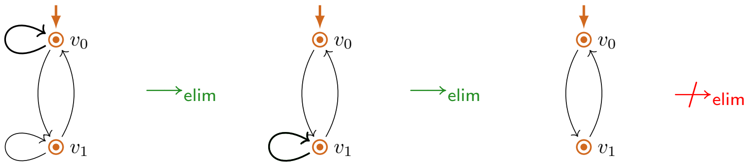


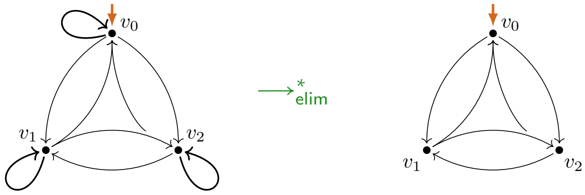
Loop elimination



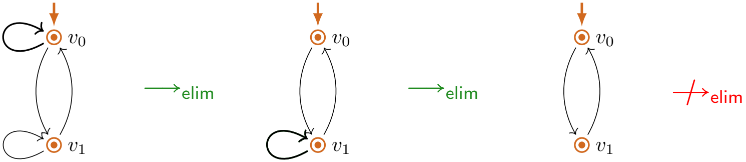


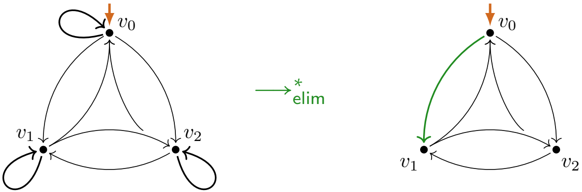
Loop elimination



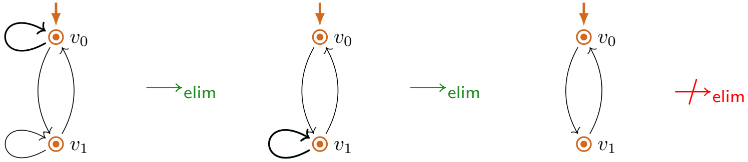


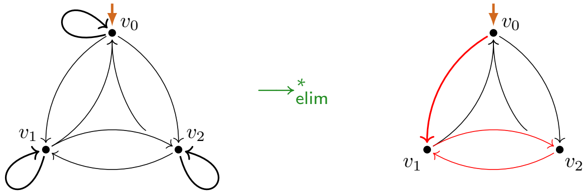
Loop elimination



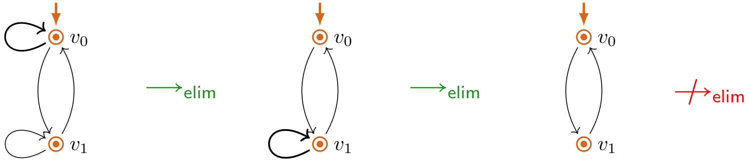


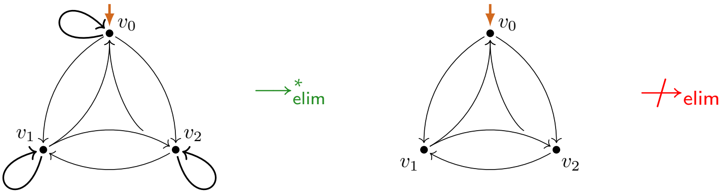
Loop elimination





Loop elimination





Loop elimination

$\longrightarrow_{\text{elim}}$: 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.*

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.*

Loop elimination, and properties

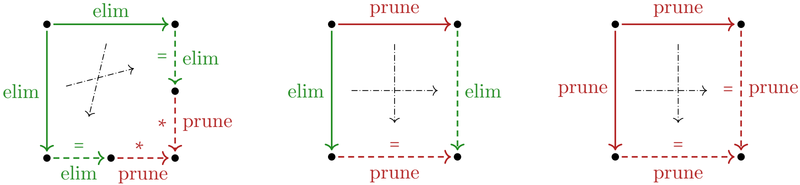
$\longrightarrow_{\text{elim}}$: eliminate a transition-induced loop by:
- removing the loop-entry transition(s)
- garbage collection

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

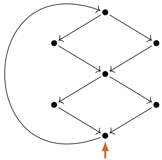
Lemma

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

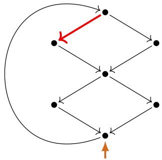
(ii) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is decreasing* [Van Oostrom, de Bruijn]



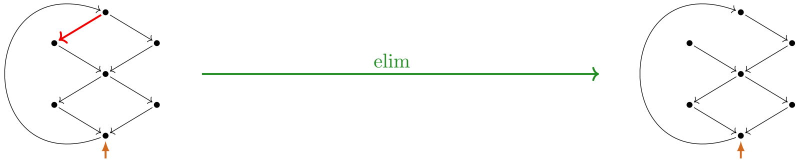
‘Critical pair’: bi-loop elimination



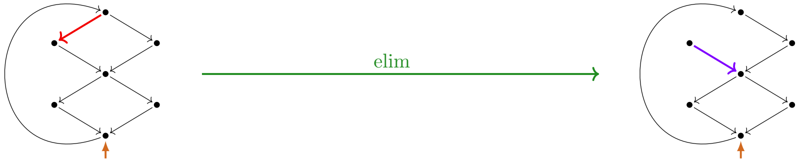
‘Critical pair’: bi-loop elimination



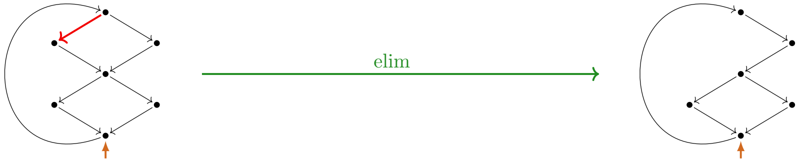
'Critical pair': bi-loop elimination



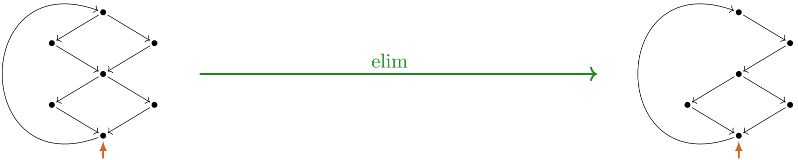
‘Critical pair’: bi-loop elimination



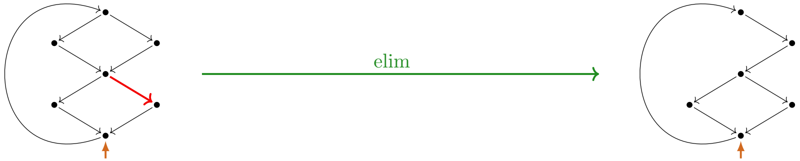
‘Critical pair’: bi-loop elimination



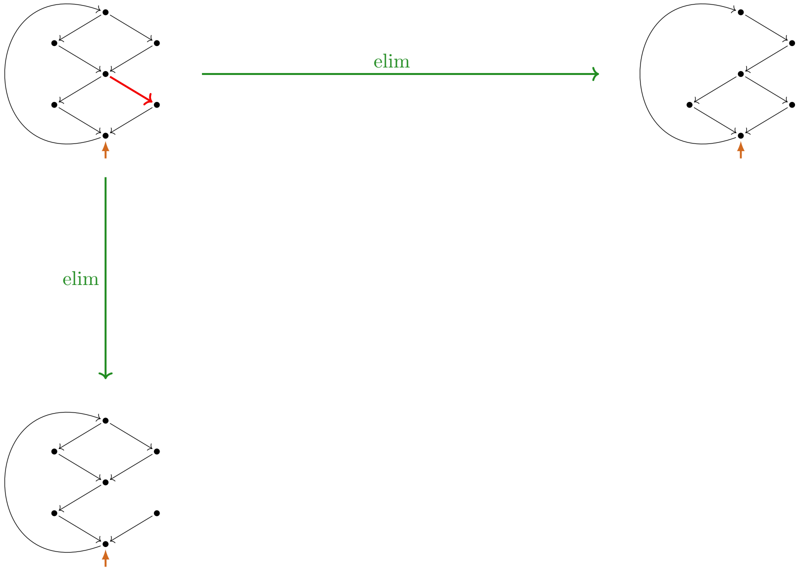
‘Critical pair’: bi-loop elimination



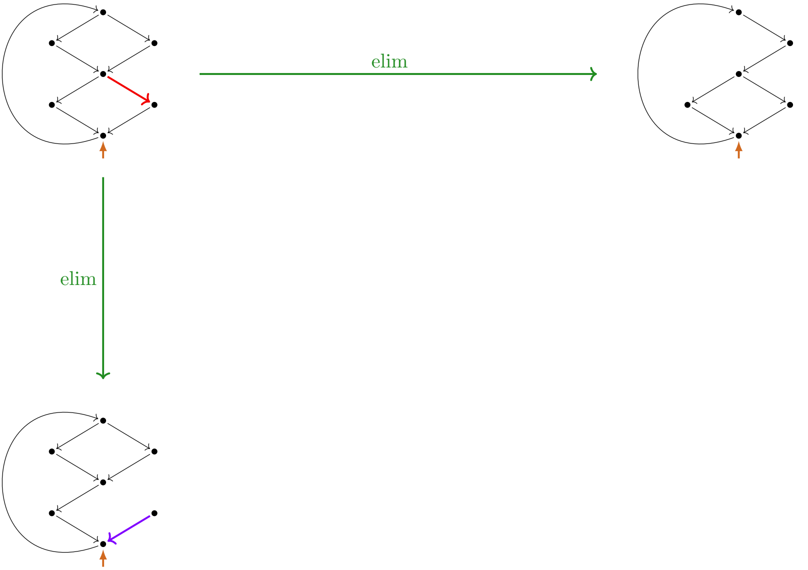
‘Critical pair’: bi-loop elimination



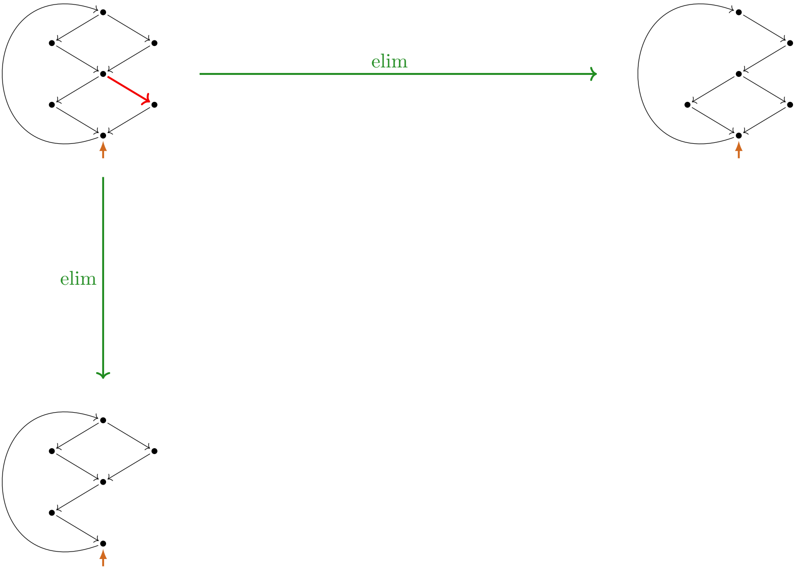
'Critical pair': bi-loop elimination



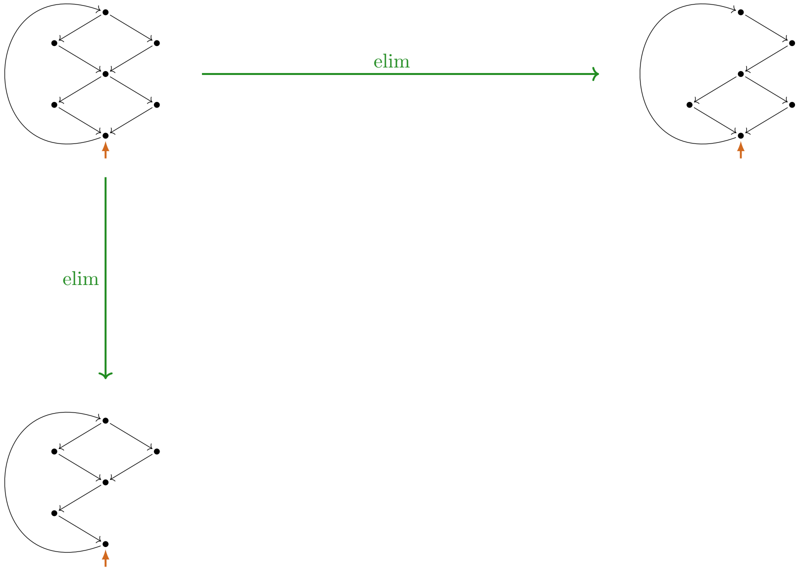
'Critical pair': bi-loop elimination



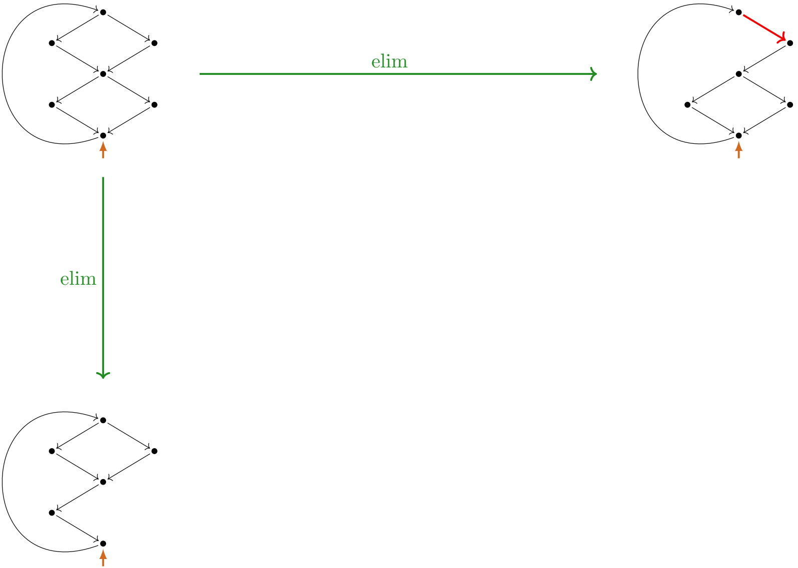
‘Critical pair’: bi-loop elimination



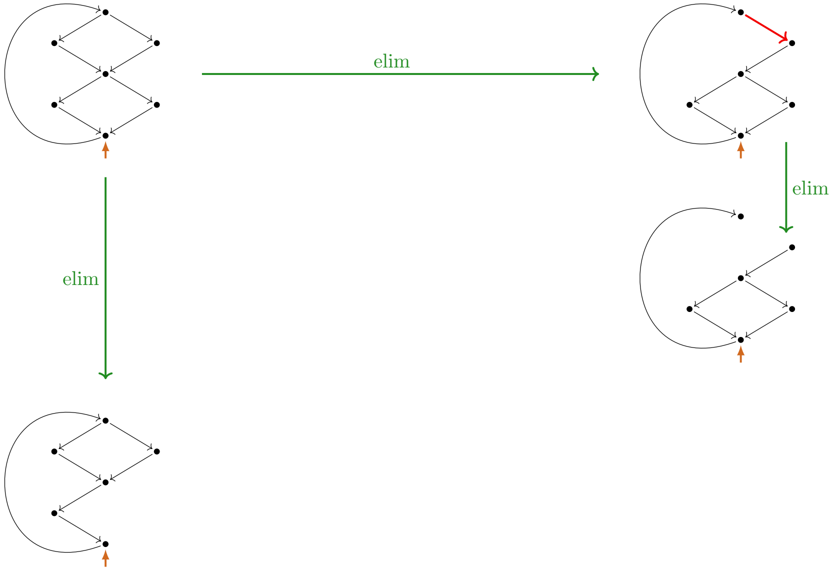
‘Critical pair’: bi-loop elimination



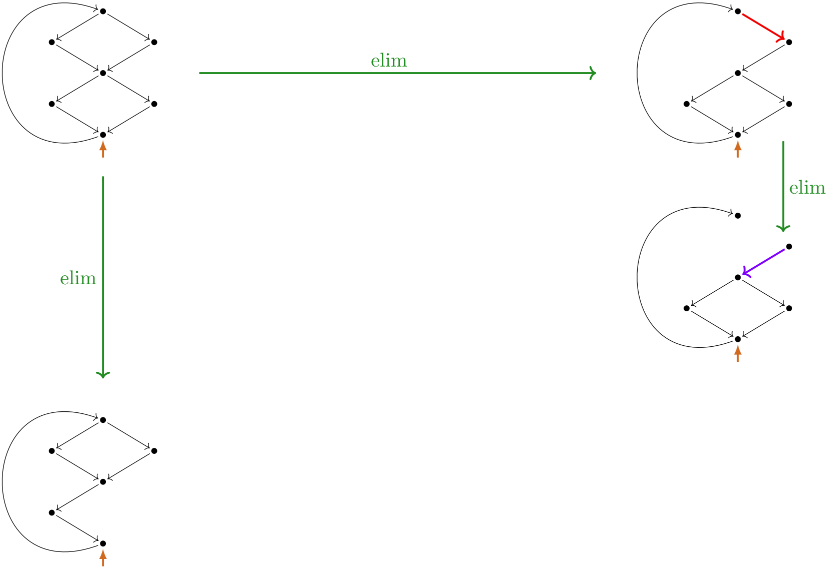
'Critical pair': bi-loop elimination



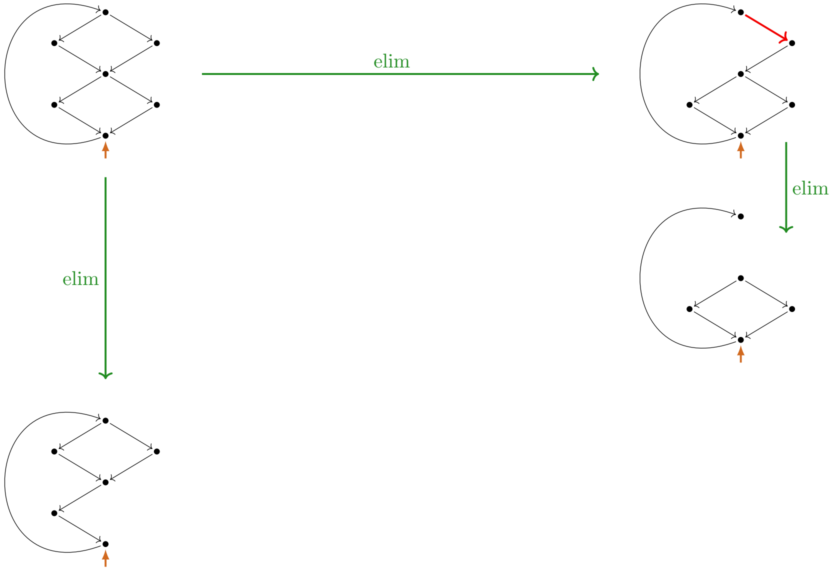
'Critical pair': bi-loop elimination



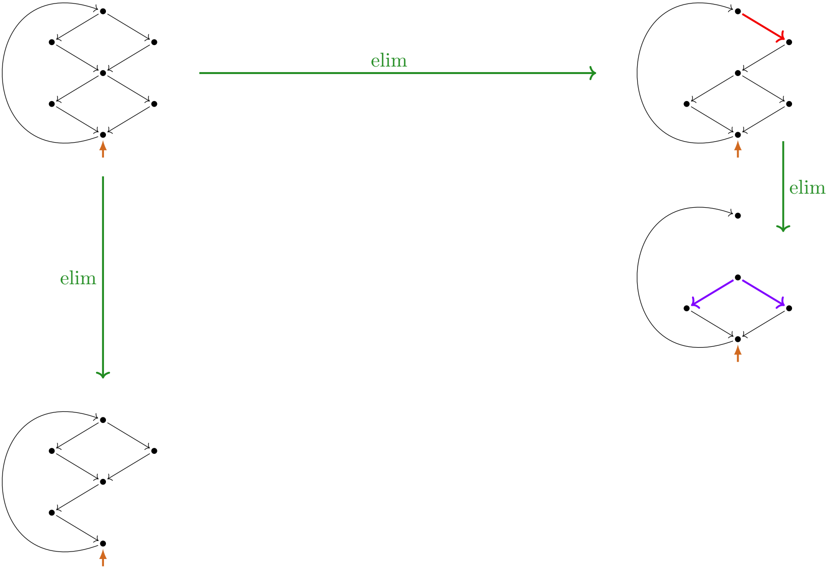
'Critical pair': bi-loop elimination



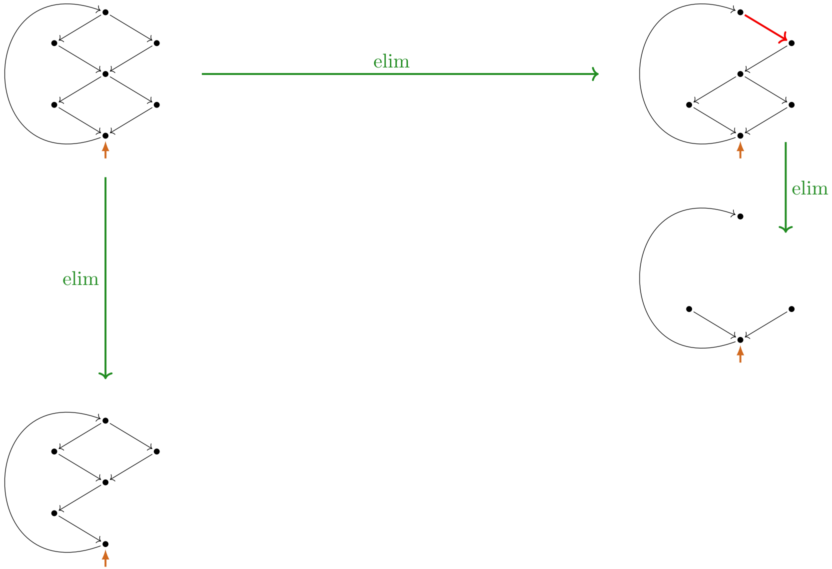
‘Critical pair’: bi-loop elimination



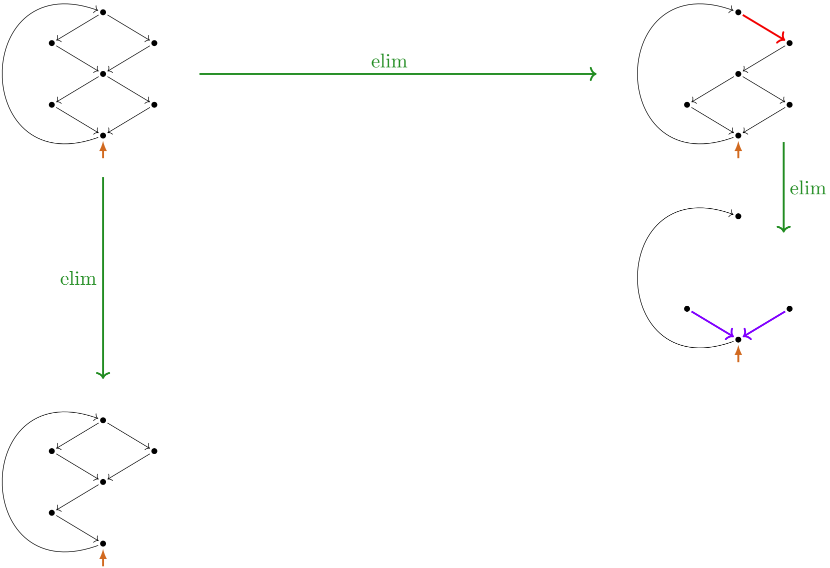
'Critical pair': bi-loop elimination



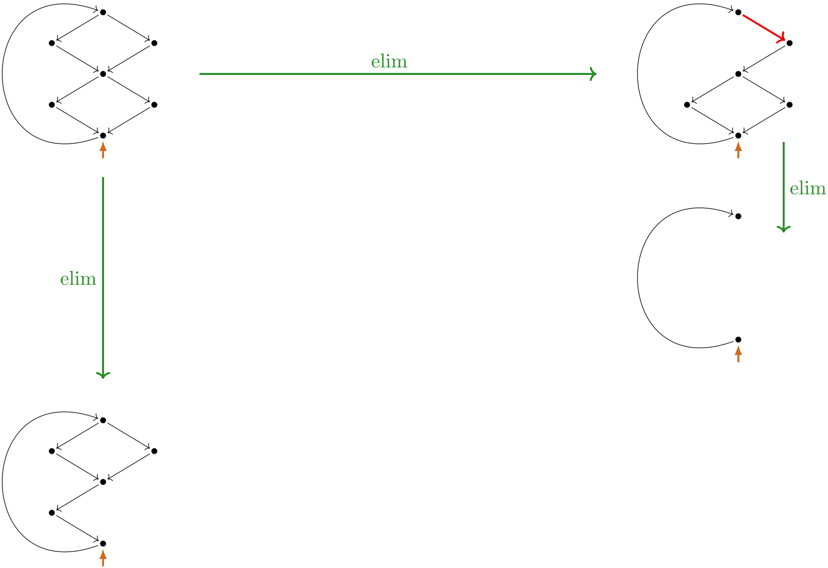
‘Critical pair’: bi-loop elimination



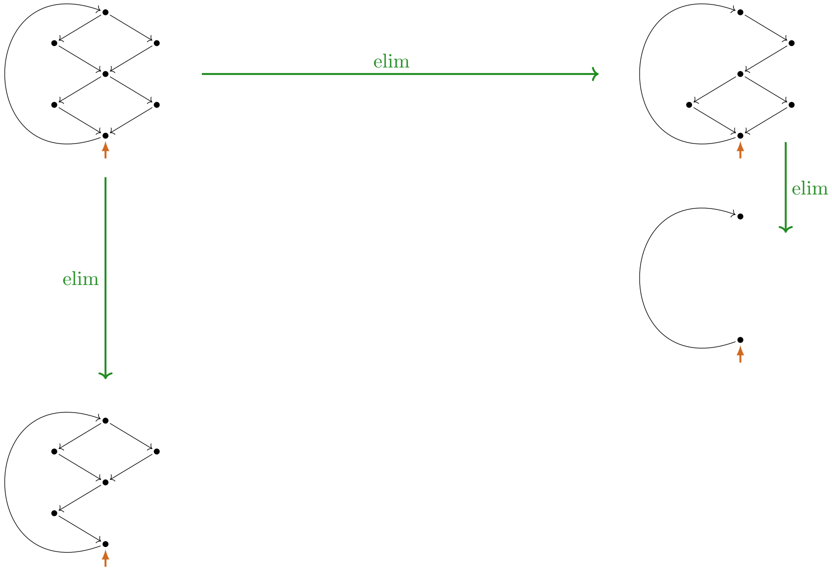
‘Critical pair’: bi-loop elimination



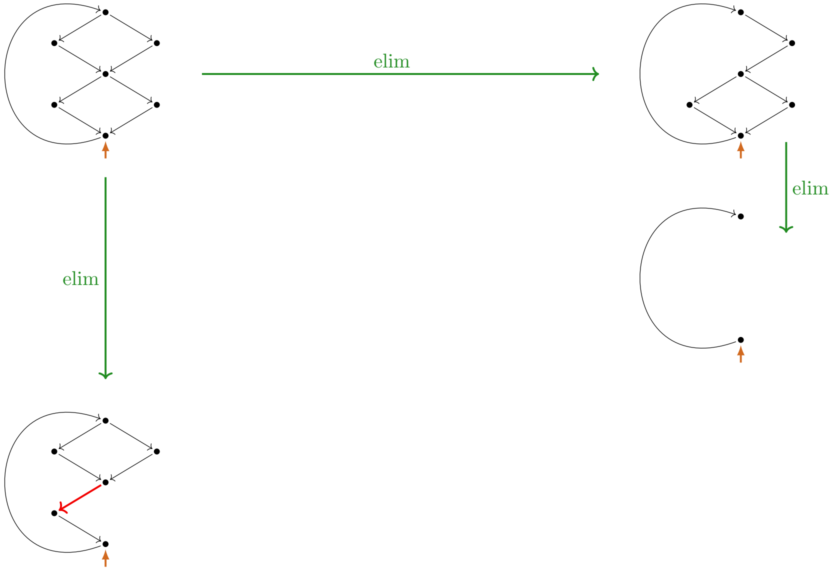
'Critical pair': bi-loop elimination



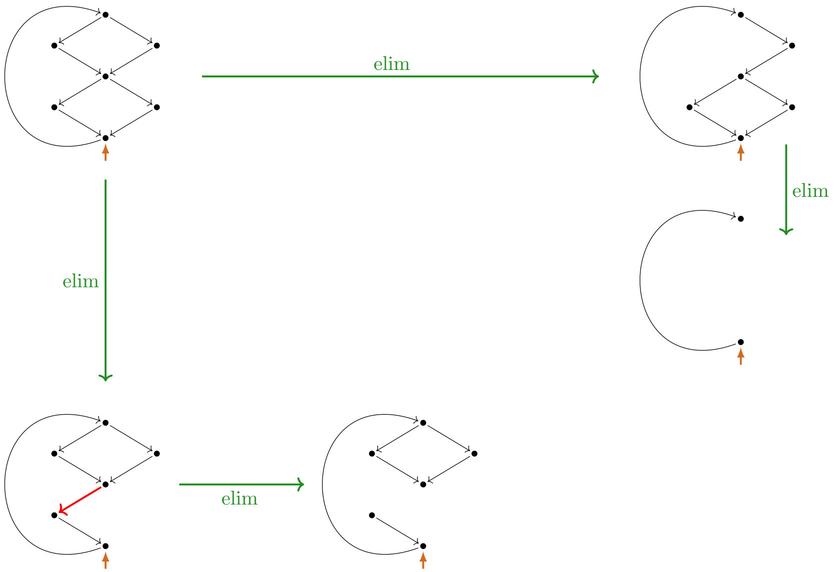
‘Critical pair’: bi-loop elimination



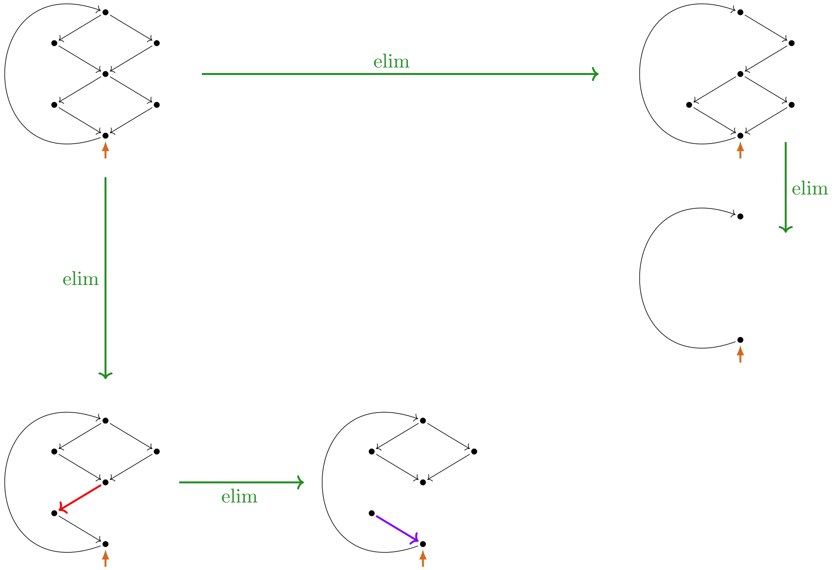
'Critical pair': bi-loop elimination



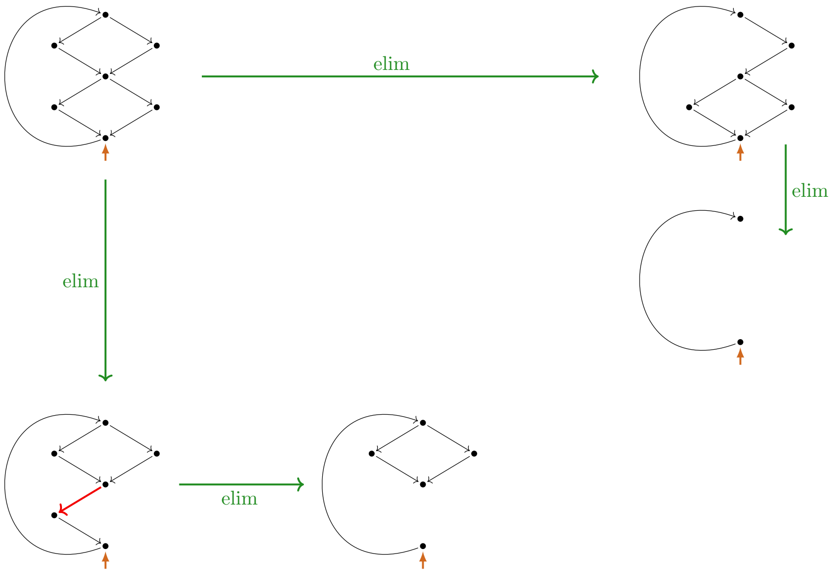
'Critical pair': bi-loop elimination



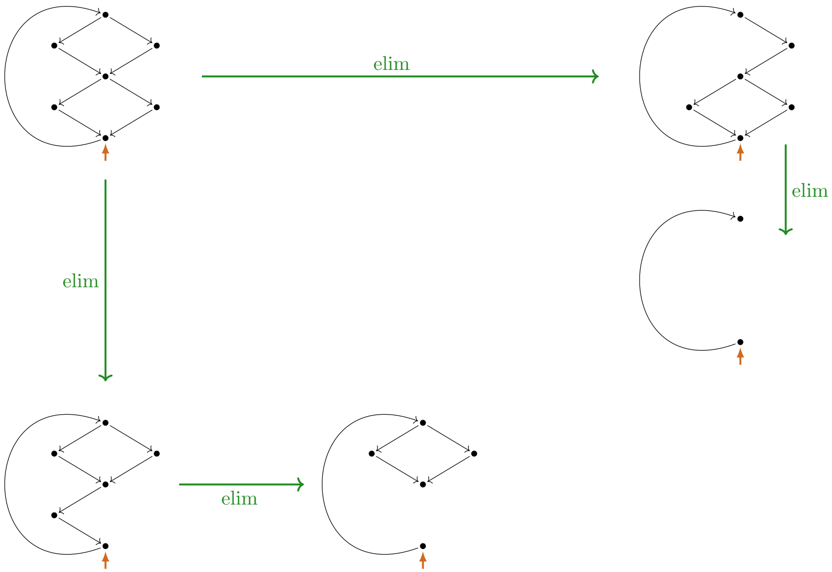
‘Critical pair’: bi-loop elimination



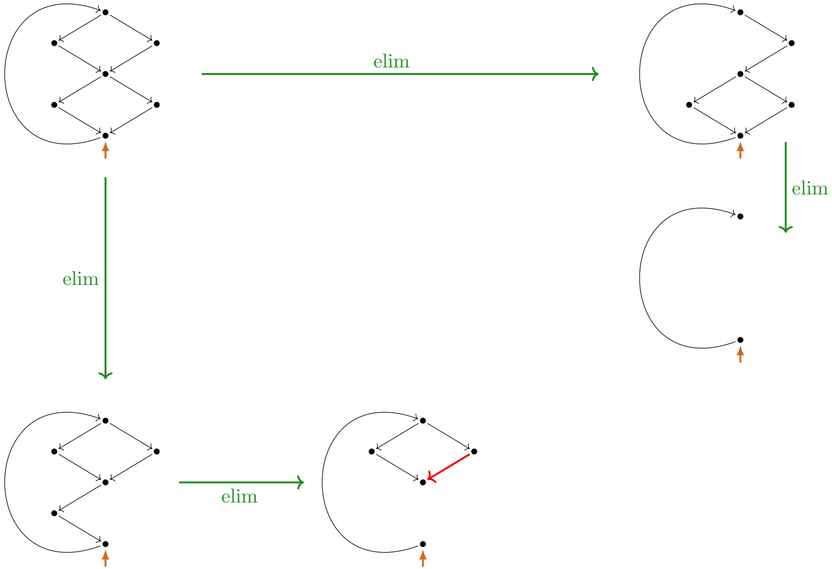
‘Critical pair’: bi-loop elimination



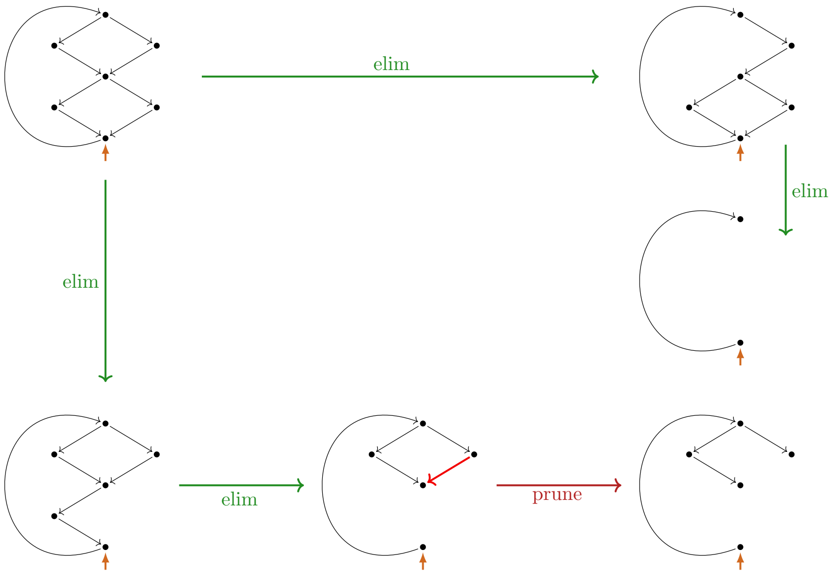
'Critical pair': bi-loop elimination



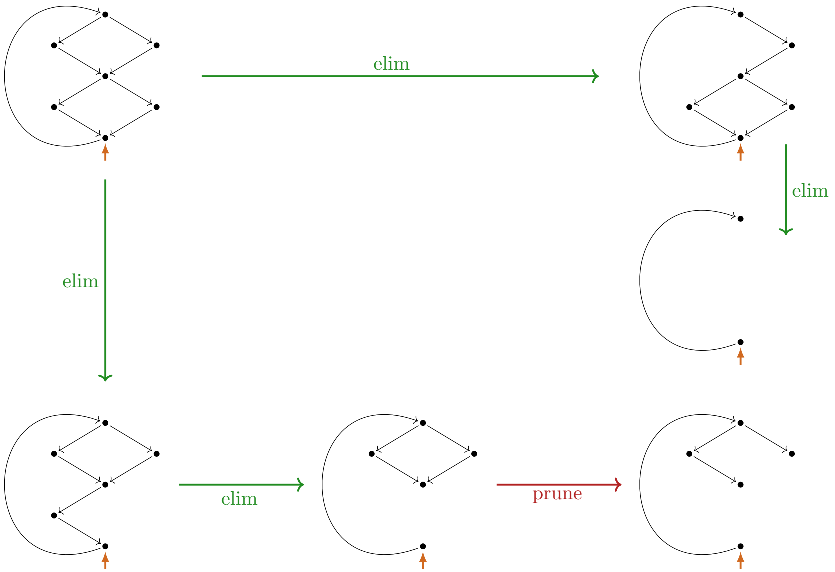
‘Critical pair’: bi-loop elimination



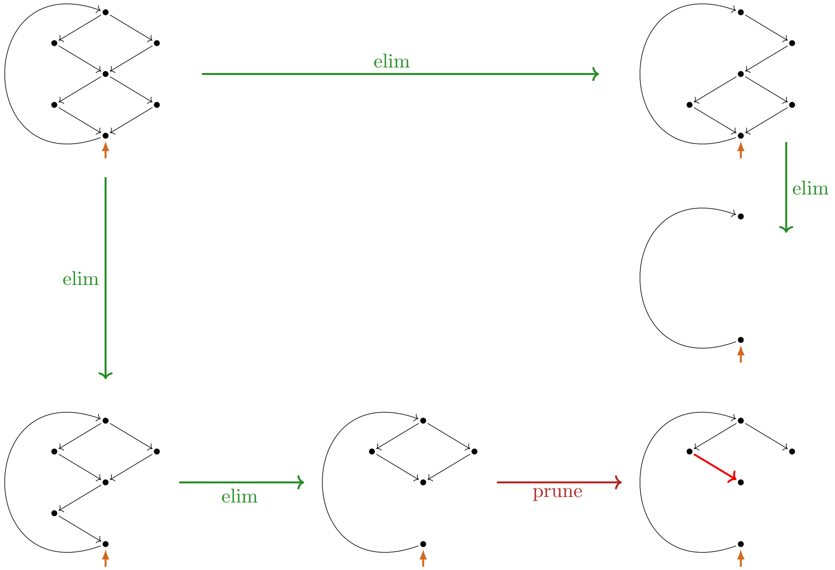
'Critical pair': bi-loop elimination



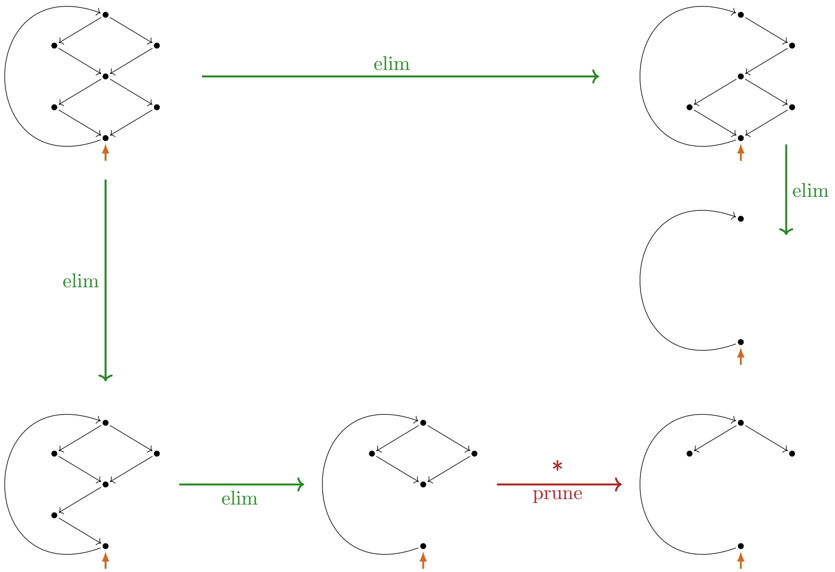
‘Critical pair’: bi-loop elimination



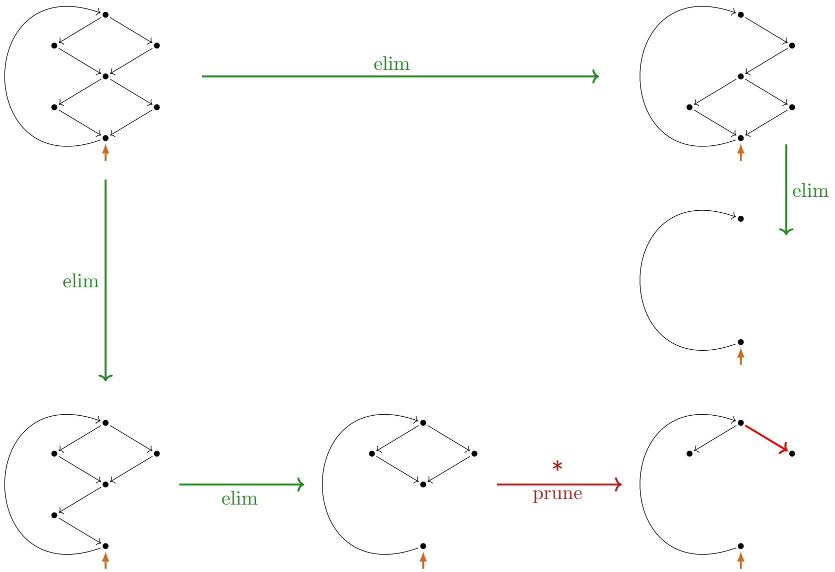
‘Critical pair’: bi-loop elimination



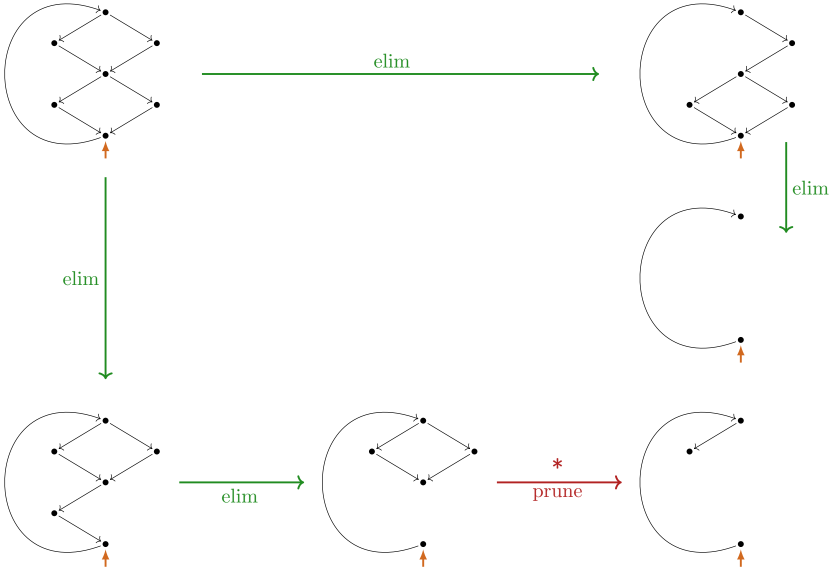
‘Critical pair’: bi-loop elimination



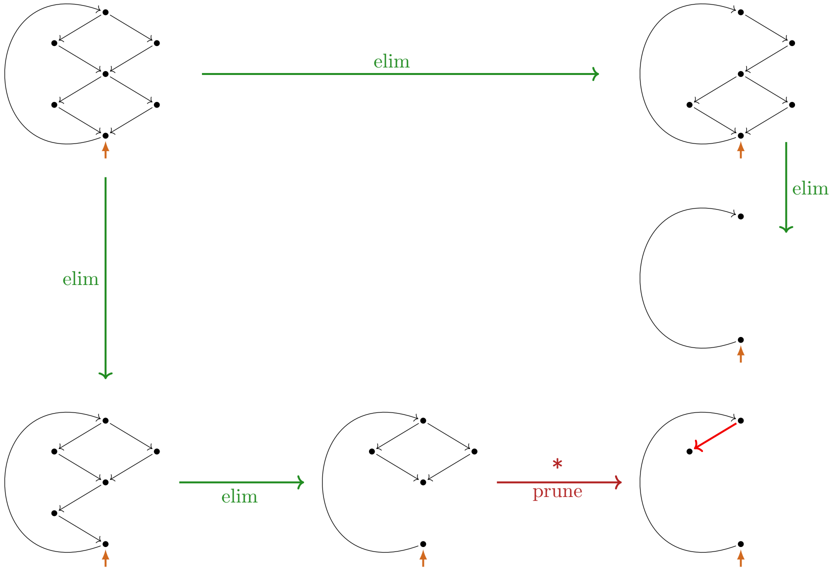
‘Critical pair’: bi-loop elimination



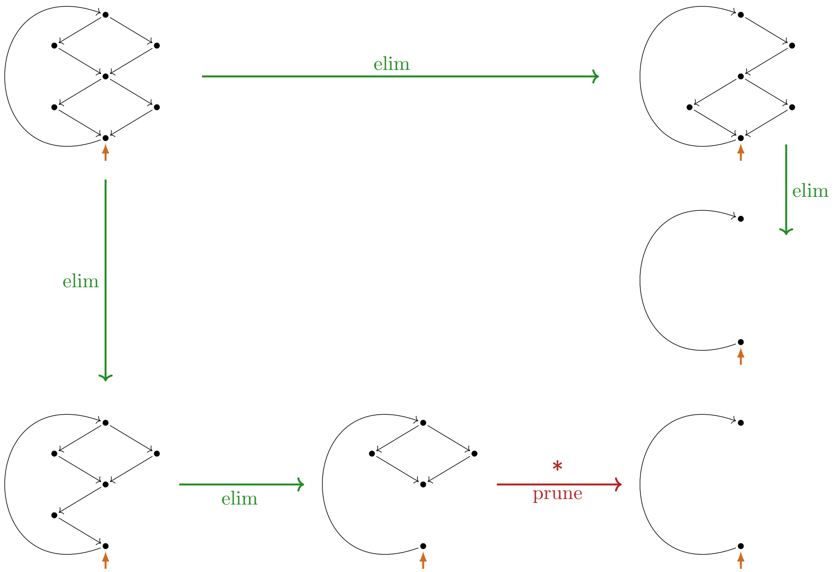
'Critical pair': bi-loop elimination



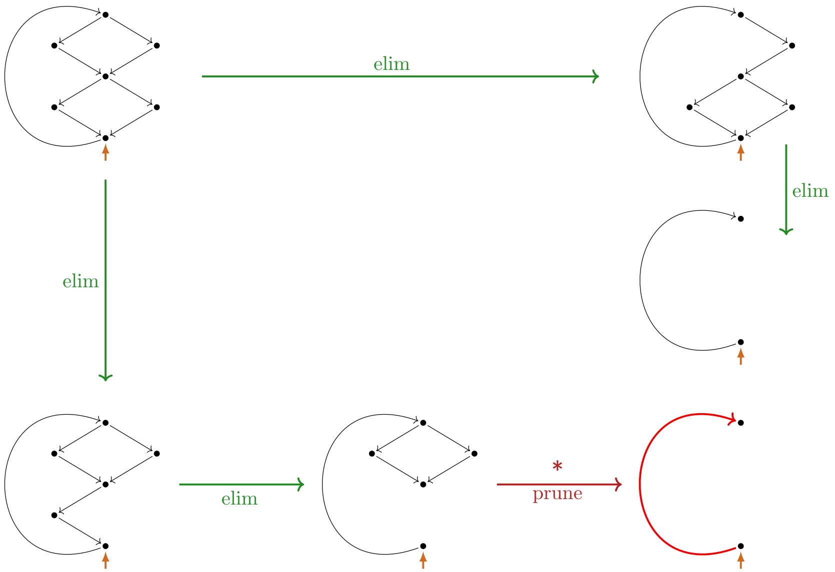
‘Critical pair’: bi-loop elimination



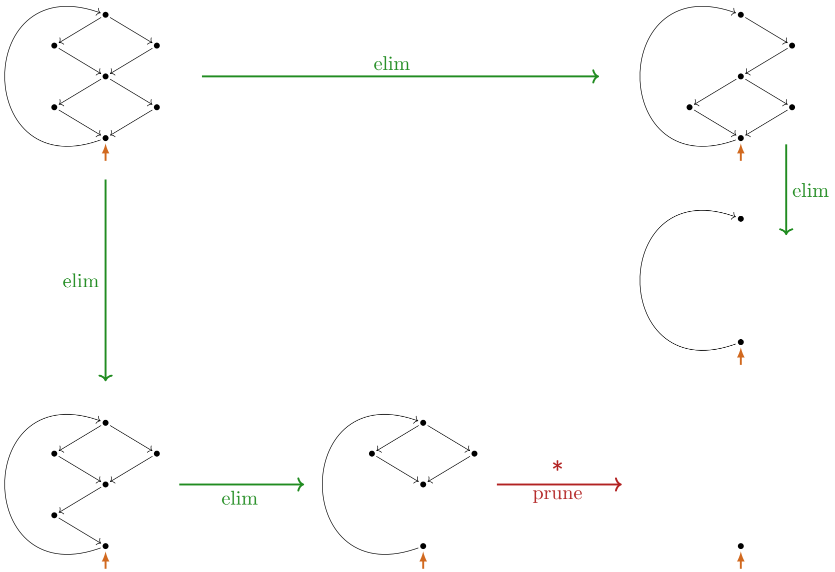
‘Critical pair’: bi-loop elimination



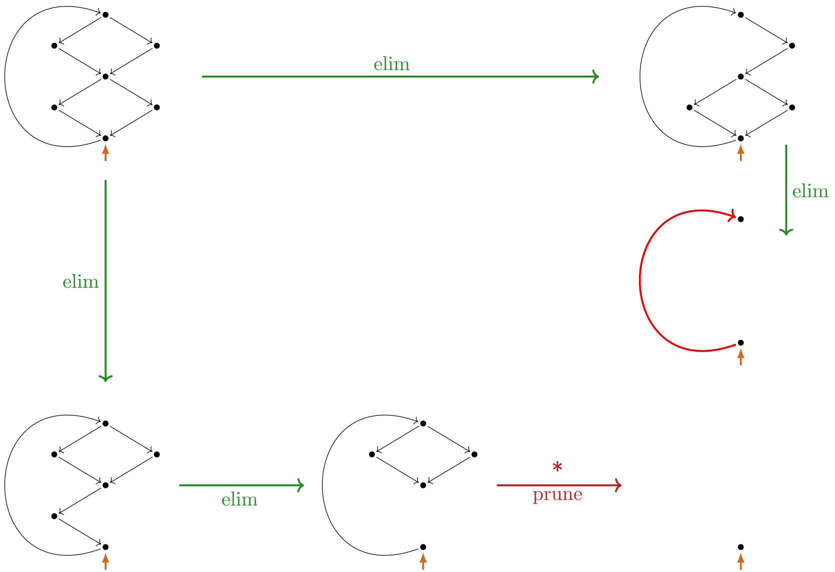
‘Critical pair’: bi-loop elimination



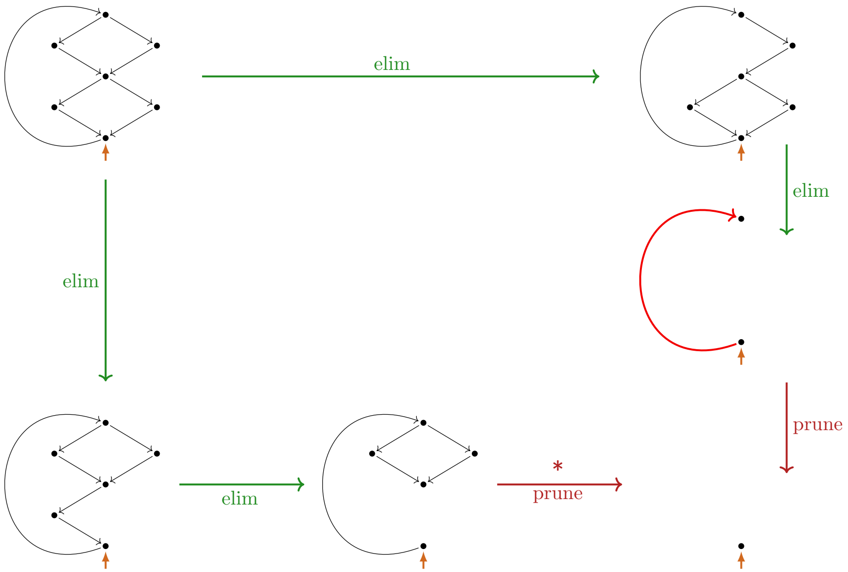
'Critical pair': bi-loop elimination



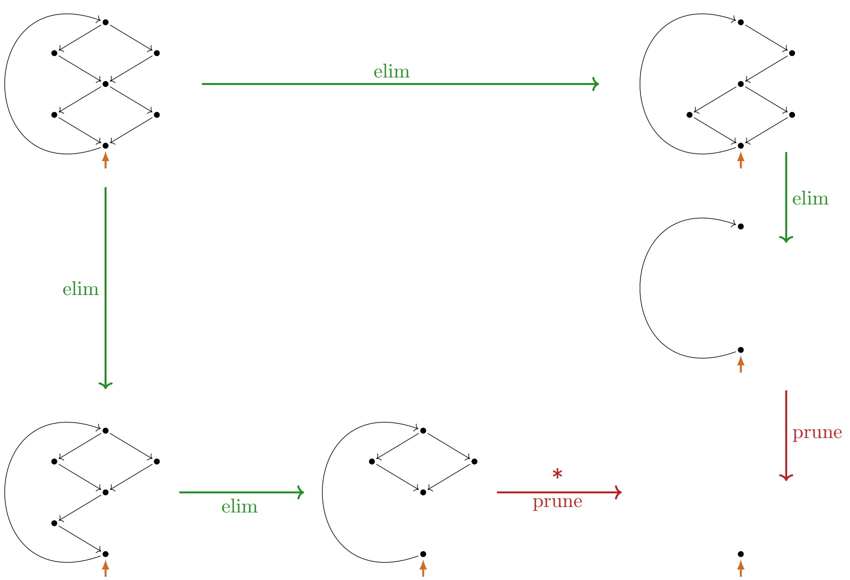
'Critical pair': bi-loop elimination



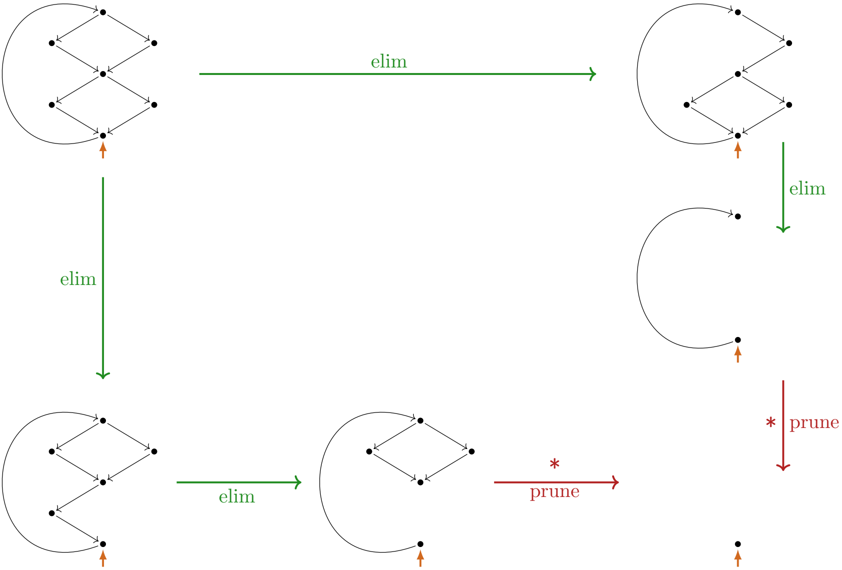
‘Critical pair’: bi-loop elimination



‘Critical pair’: bi-loop elimination



‘Critical pair’: bi-loop elimination



Loop elimination, and properties

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

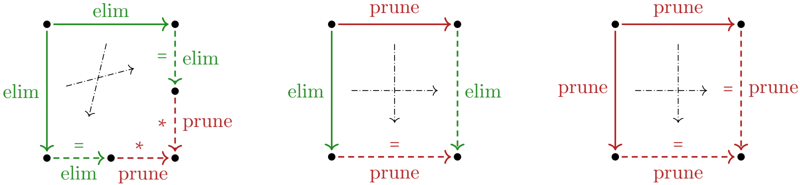
- removing the loop-entry transition(s)
- garbage collection

$\longrightarrow_{\text{prune}}$: 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

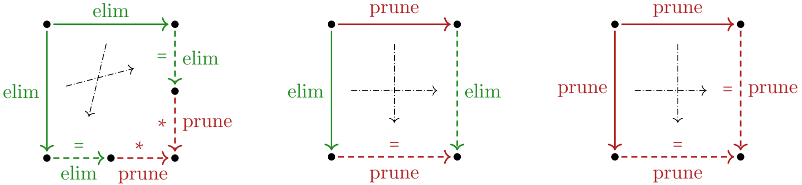
$\longrightarrow_{\text{prune}}$: remove a transition to a deadlocking state

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:

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

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

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

(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:

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

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

(ii) *There is an $\longrightarrow_{\text{elim}}$ 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:

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

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

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

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

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

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

Structure property LEE

Definition

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

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

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 $\longrightarrow_{\text{elim,prune}}$ normal form without an infinite trace.

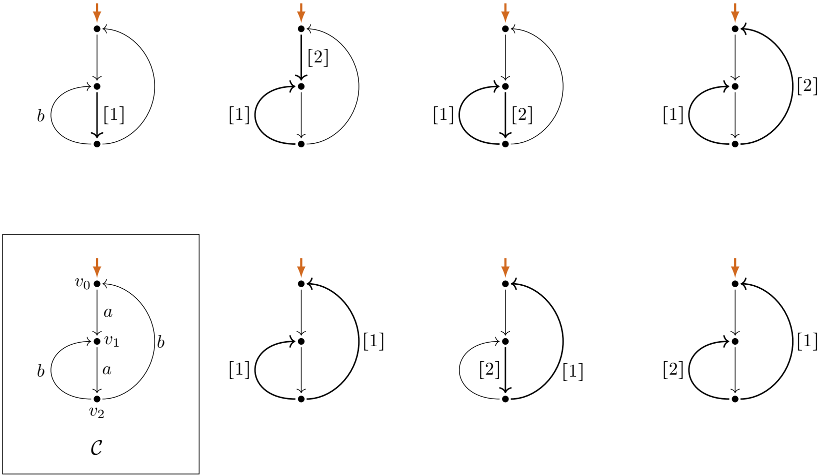
(iv) Every $\longrightarrow_{\text{elim}}$ normal form is without an infinite trace.

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

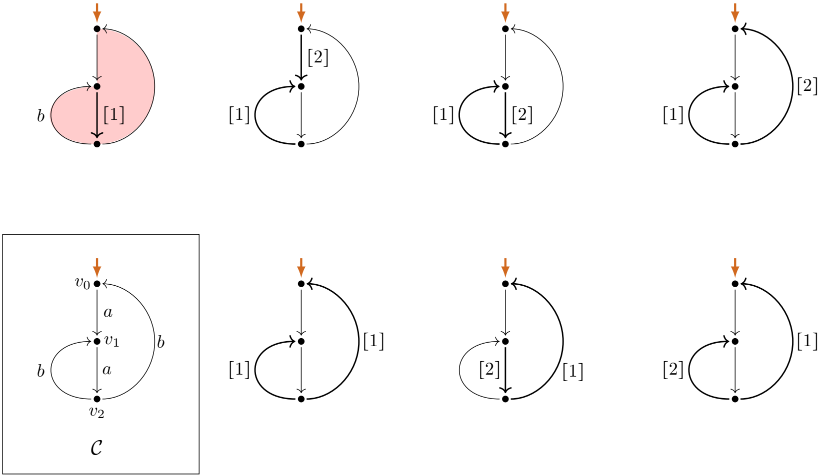
Theorem (efficient decidability)

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

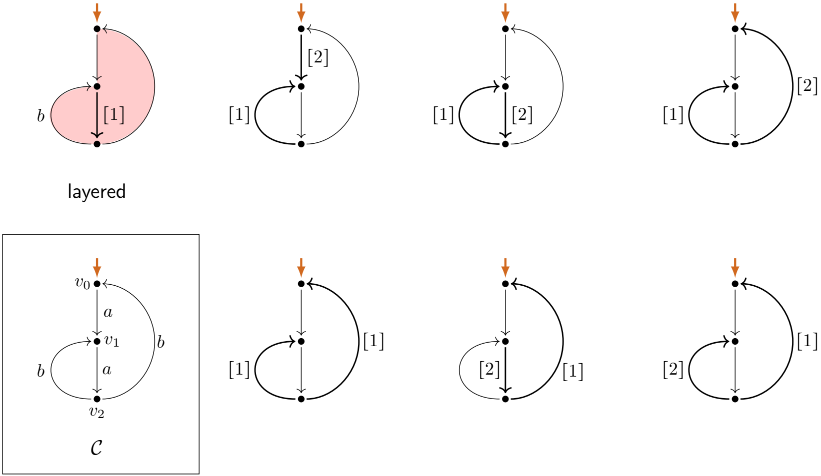
7 LEE-witnesses



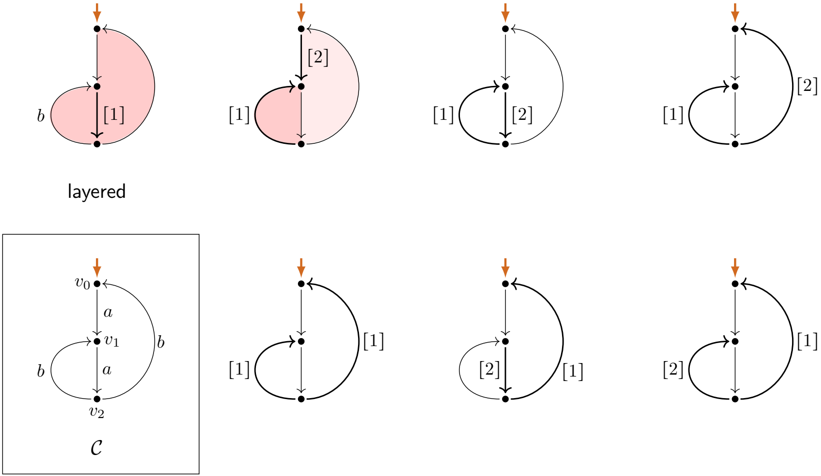
7 LEE-witnesses



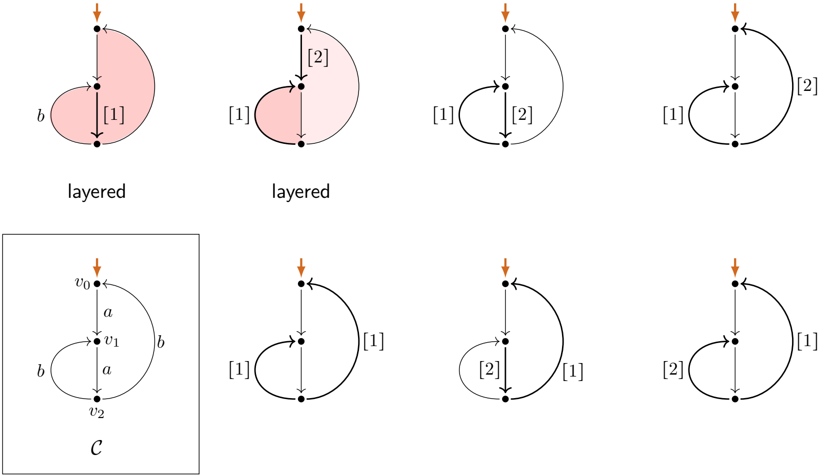
7 LEE-witnesses



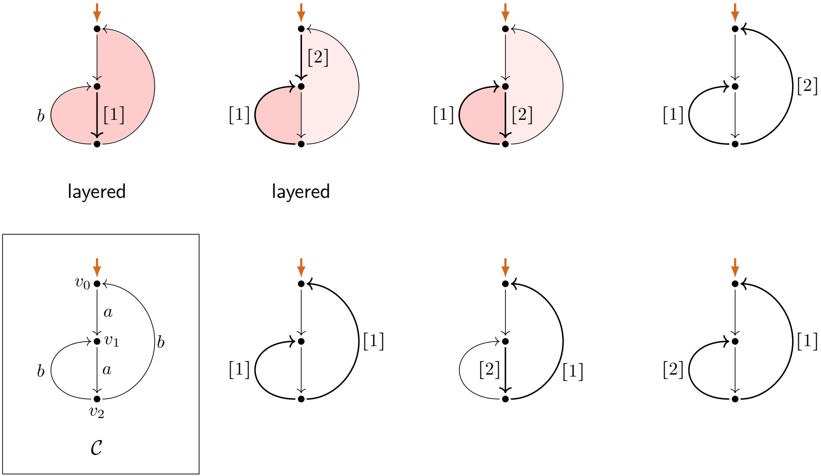
7 LEE-witnesses



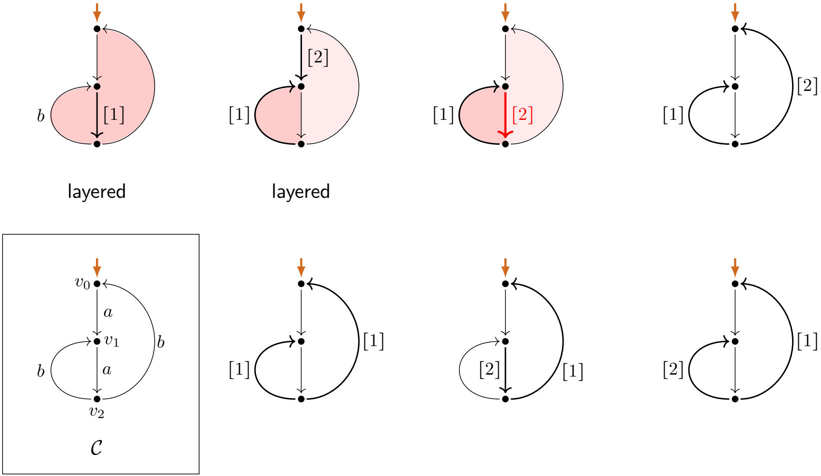
7 LEE-witnesses



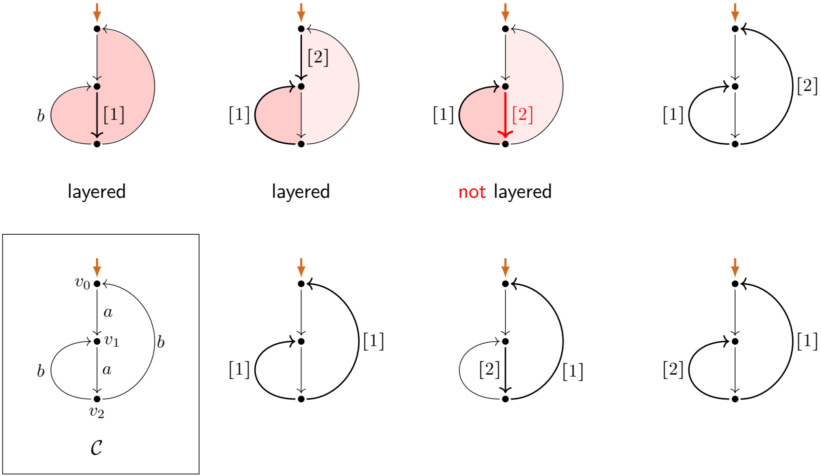
7 LEE-witnesses



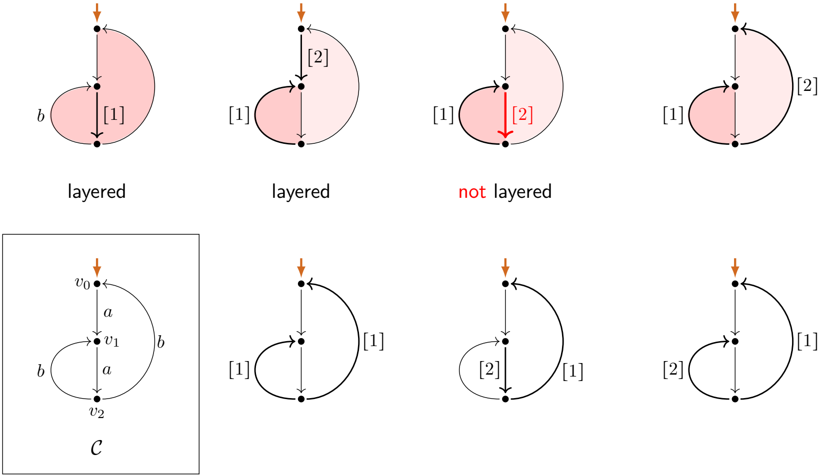
7 LEE-witnesses



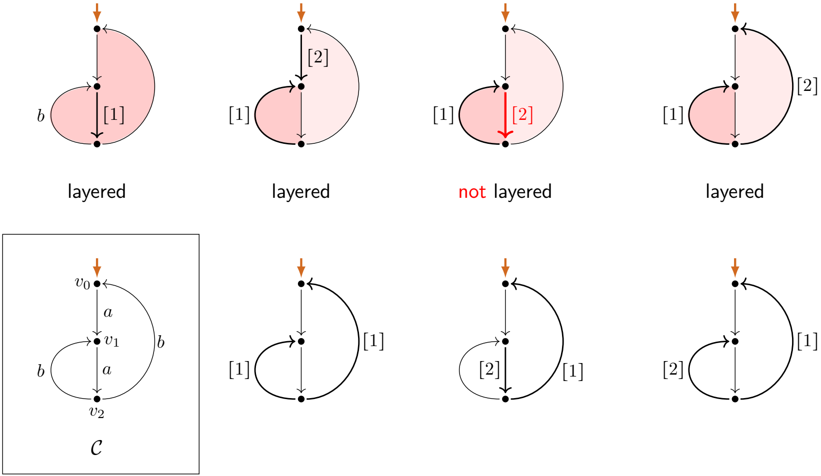
7 LEE-witnesses



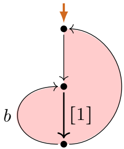
7 LEE-witnesses



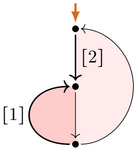
7 LEE-witnesses



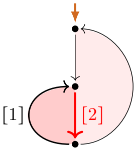
7 LEE-witnesses



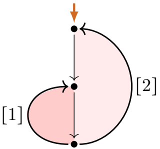
layered



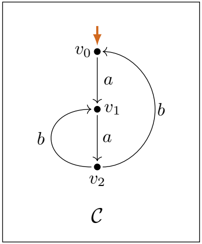
layered

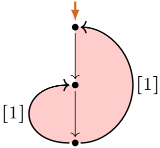


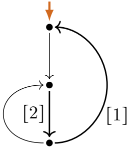
not layered

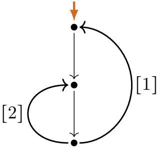


layered

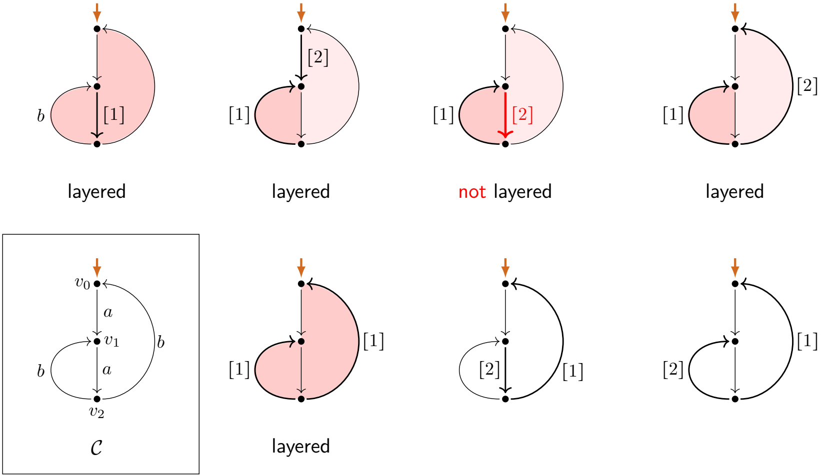




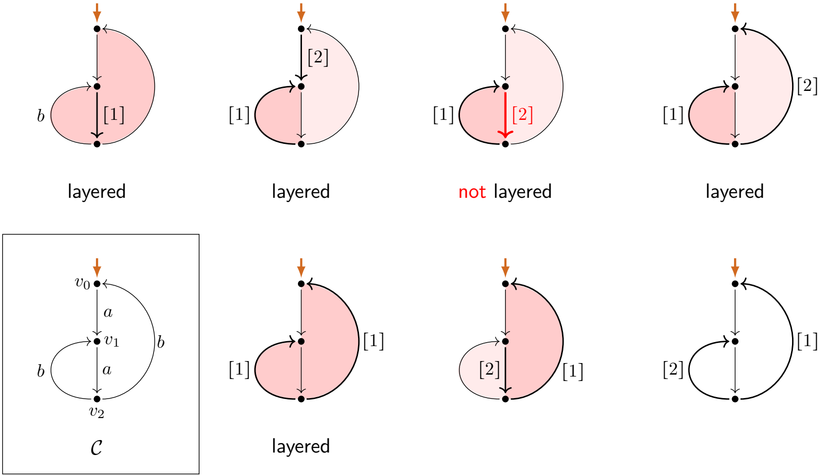




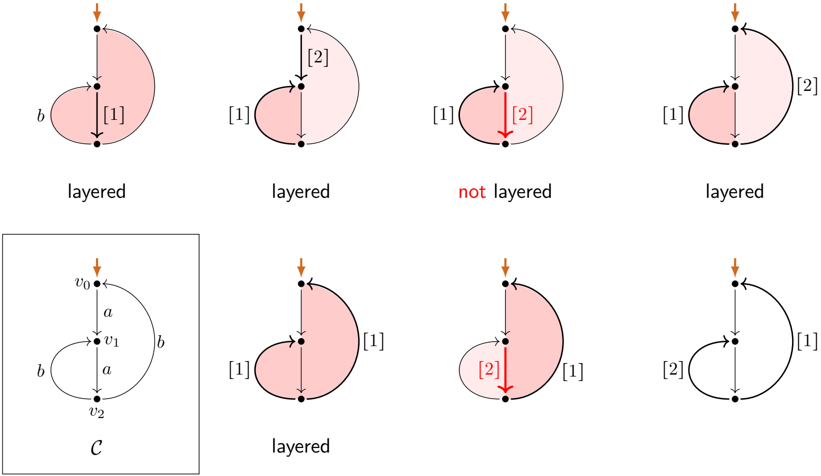
7 LEE-witnesses



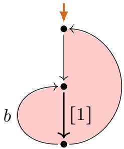
7 LEE-witnesses



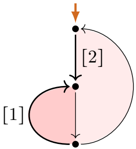
7 LEE-witnesses



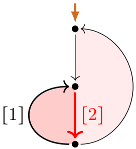
7 LEE-witnesses



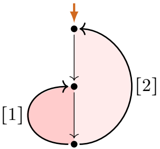
layered



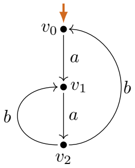
layered



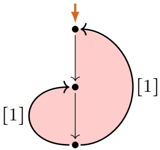
not layered



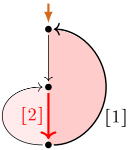
layered



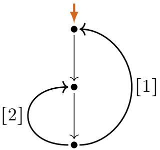
$\mathcal{C}$



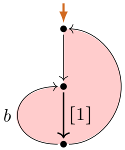
layered



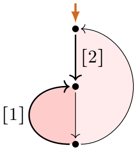
not layered



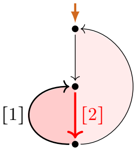
7 LEE-witnesses



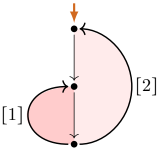
layered



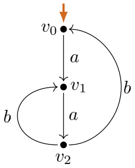
layered



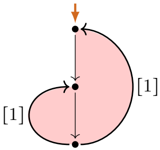
not layered



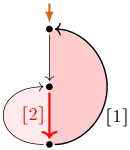
layered



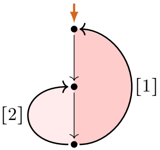
$\mathcal{C}$



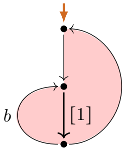
layered



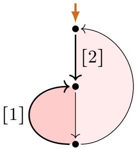
not layered



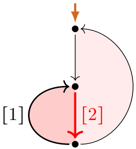
7 LEE-witnesses



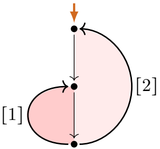
layered



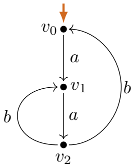
layered



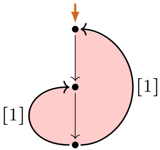
not layered



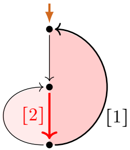
layered



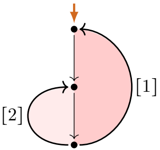
$\mathcal{C}$



layered

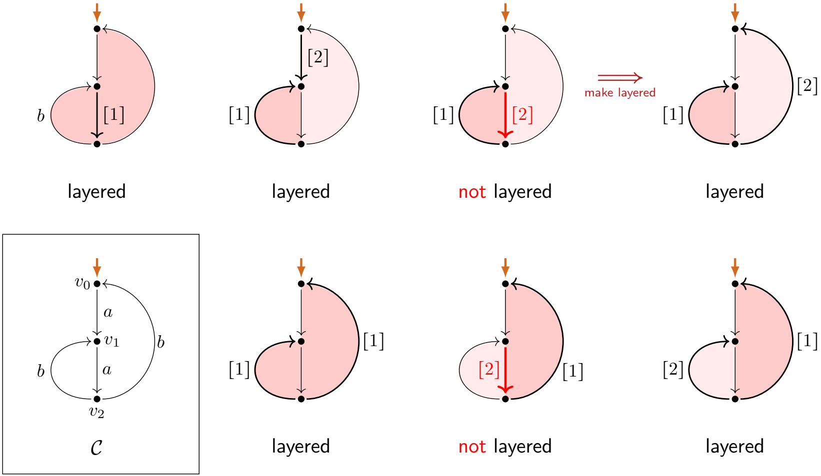


not layered

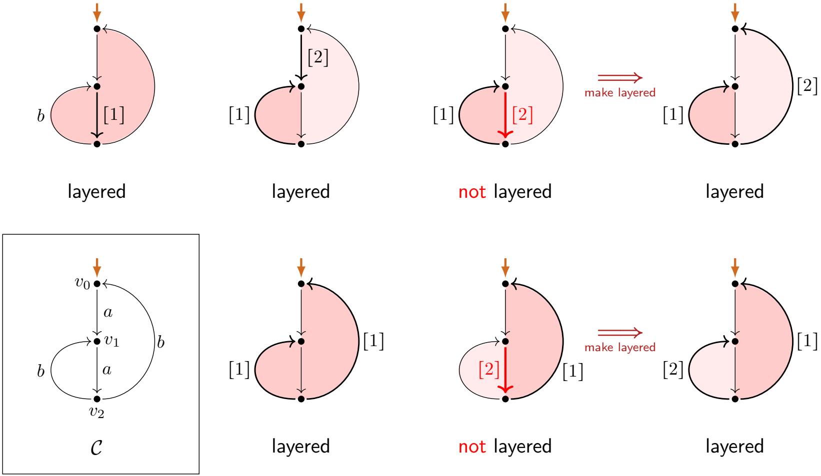


layered

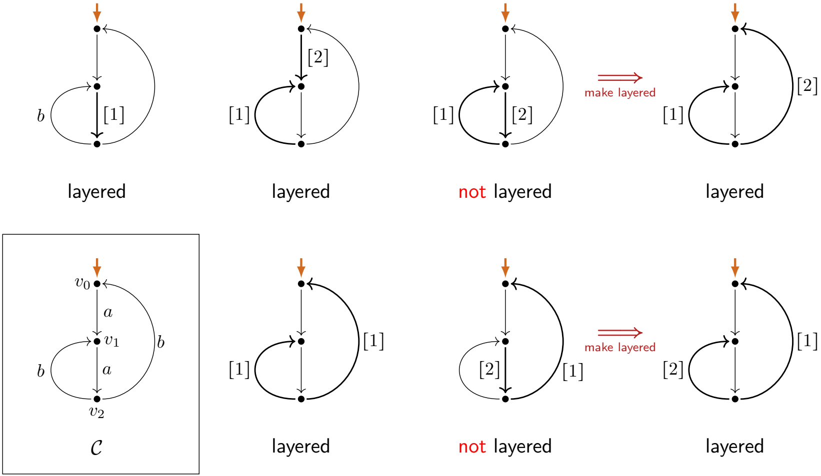
7 LEE-witnesses



7 LEE-witnesses



7 LEE-witnesses



Interpretation/extraction correspondences with LEE

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

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

Process **i**nterpretations $P(e)$ of $(*/\mathbf{1})$ 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 $G \,\underline{\leftrightarrow}\, P(e)$.

Interpretation/extraction correspondences with LEE

(⇐ G/Fokkink 2020, G 2021)

(Int)$_{P}^{(*/\cancel{1})}$: *P-$(*/\cancel{1})$-expressible graphs have the 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*

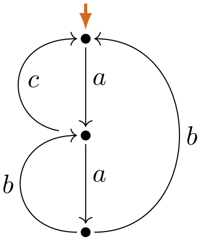
From every finite process graph G with LEE a regular expression e can be **e**xtracted such that $G \leftrightarrow P(e)$.

(Coll): LEE *is preserved under collapse*

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

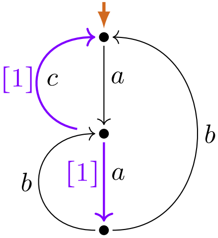
Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

G_2



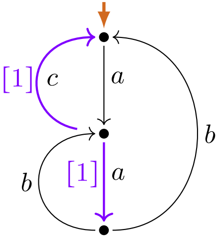
Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_2}$



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

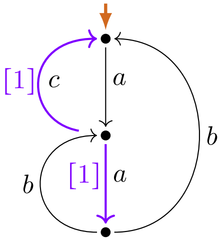
$\widehat{G_2}$



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

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

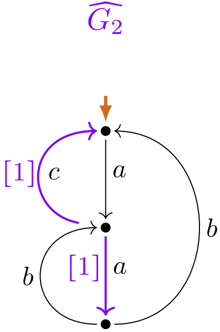
$\widehat{G_2}$



a

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

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



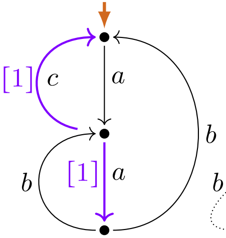
$$(a \cdot 1) \cdot ((\qquad)^{*} \cdot 0)$$

$$\downarrow a$$

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

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

$\widehat{G_2}$



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

a

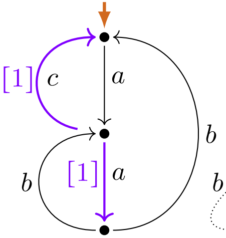
$(\qquad)^* \cdot 0$

b

b

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

$\widehat{G_2}$



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

$\downarrow a$

$(\qquad)^* \cdot 0$

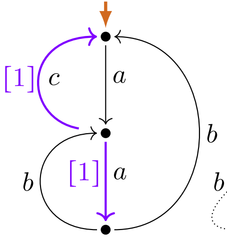
b

b

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

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

$\widehat{G_2}$



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

$\downarrow a$

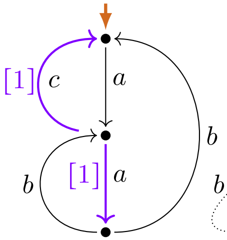
$(\qquad)^* \cdot 0$

$\downarrow a$

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

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

$\widehat{G_2}$



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

$$\Big\downarrow a$$

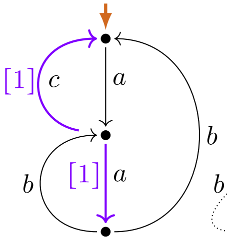
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0$$

$$\Big\downarrow a$$

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

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

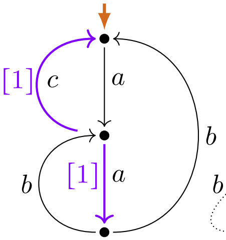
$\widehat{G_2}$



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

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

$\widehat{G_2}$



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

$$\Big\downarrow a$$

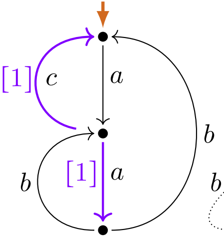
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0$$

$$\Big\downarrow a$$

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

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

$\widehat{G_2}$



$P(e) \rightharpoonup G_2$

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

$$\downarrow a$$

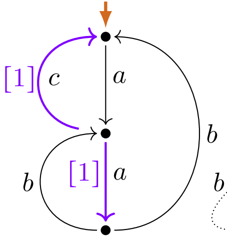
$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_2}$



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

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

$$\Big\downarrow a$$

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

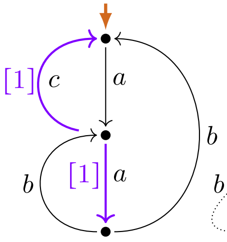
$$\downarrow a$$

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

c b b

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

$\widehat{G_2}$



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

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

$$\downarrow a$$

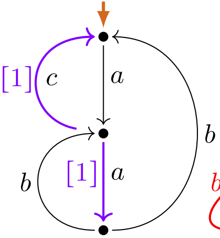
$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_2}$



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

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

$$\downarrow a$$

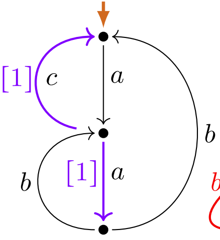
$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_2}$



$$P(e) \mathrel{\underline{\rightarrow}} G_2 \not\mathrel{\underline{\leftrightarrow}} P(e)$$

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

$$\downarrow a$$

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

$$\downarrow a$$

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

$$\not\xrightarrow{c} \qquad \not\xrightarrow{b} \qquad \not\xrightarrow{b}$$

Interpretation of extracted expression

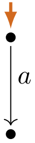
G_2'

$P(e) = G_2'$

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

Interpretation of extracted expression

G_2'



$$P(e) = G_2'$$

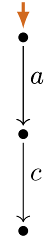
$$\overbrace{(a \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0 \Big)}^{e}$$

$$\big\downarrow a$$

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

Interpretation of extracted expression

G'_2 $\qquad\qquad$ $P(e) = G'_2$



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

$$\Big\downarrow a$$

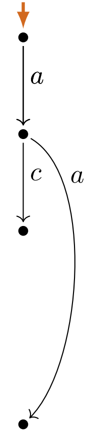
$$(1 \cdot 1) \cdot \Big(\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\big)^* \cdot 0\Big)$$

$$\Big\downarrow c$$

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

Interpretation of extracted expression

G'_2



$P(e) = G'_2$

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

$$\downarrow a$$

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

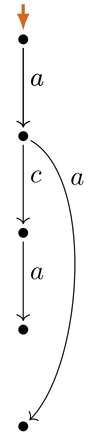
$$\downarrow c \qquad\qquad\qquad \xrightarrow{a}$$

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

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

Interpretation of extracted expression

G_2'



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

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\color{red}{\downarrow a}$$

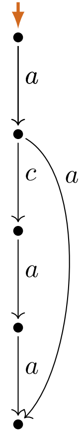
$$\big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \big) \cdot 0 \big)$$

$$a$$

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

Interpretation of extracted expression

G_2'



$P(e) = G_2'$

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

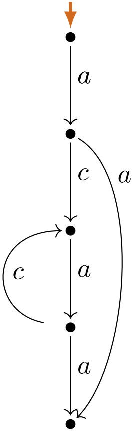
$$\Big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \Big) \cdot 0 \Big)$$

$$\downarrow a$$

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

Interpretation of extracted expression

G_2'



$P(e) = G_2'$

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

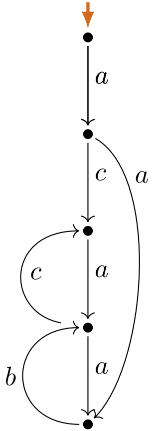
$$\big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \big) \cdot 0 \big)$$

$$\downarrow a$$

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

Interpretation of extracted expression

G_2'



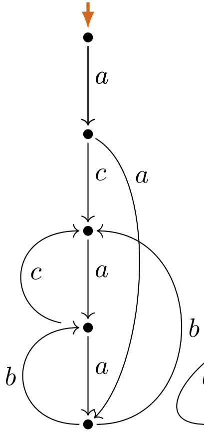
$$P(e) = G_2'$$

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

c b a

Interpretation of extracted expression

G'_2

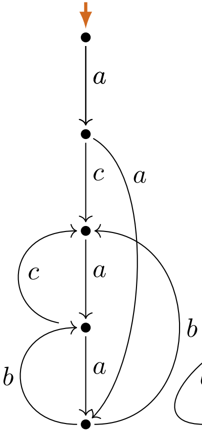


$$P(e) = G'_2$$

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

Interpretation of extracted expression

G_2'



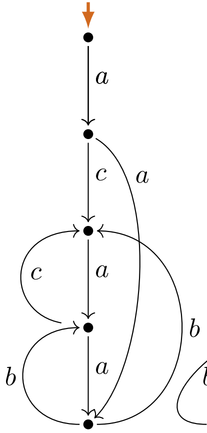
$$P(e) = G_2' \mathrel{\underline{\rightarrow}} G_2$$

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

Interpretation of extracted expression

G_2'

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



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

LEE under bisimulation?

LEE under bisimulation

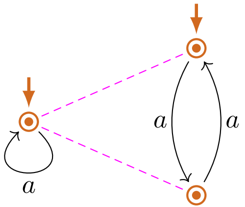
Observation

- LEE is **not** invariant under bisimulation.

LEE under bisimulation

Observation

- LEE is not invariant under bisimulation.

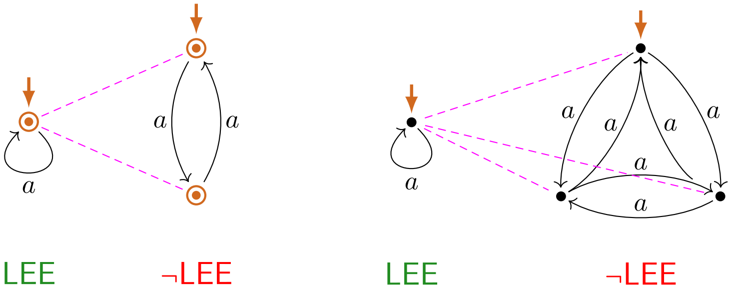


LEE ¬LEE

LEE under bisimulation

Observation

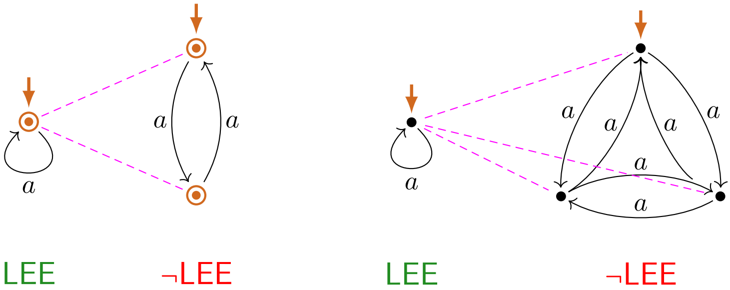
- LEE is not invariant under bisimulation.



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:*

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

LEE under functional bisimulation

Lemma

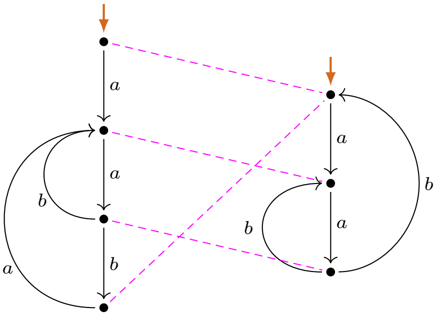
(i) LEE *is preserved by functional bisimulations:*

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

Proof (Idea).

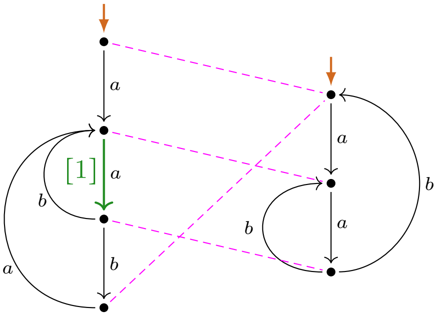
Use loop elimination in G_1 to carry out loop elimination in G_2.

Collapsing LEE-witnesses



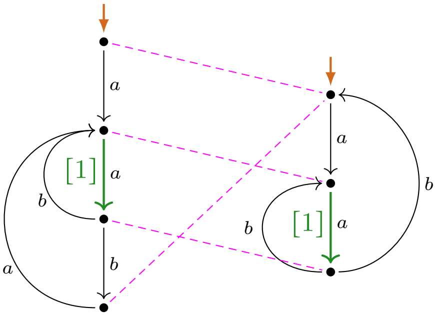
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



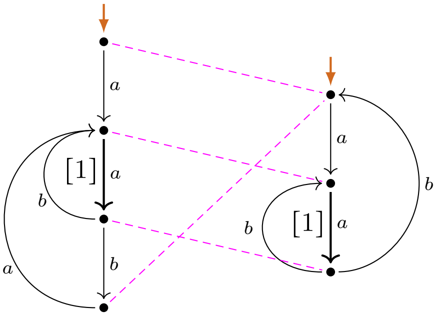
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



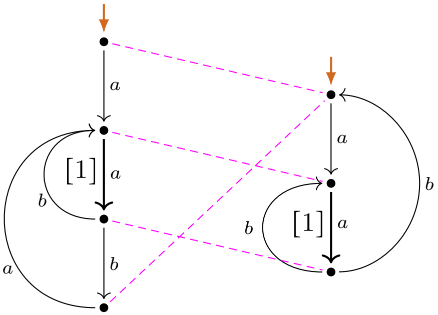
$P(a(a(b+ba))^* \cdot 0)$

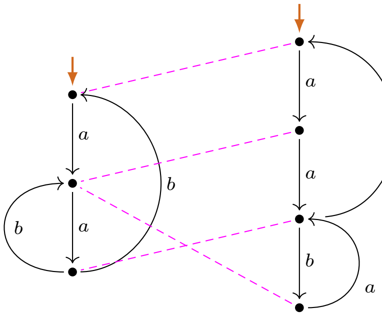
Collapsing LEE-witnesses



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

Collapsing LEE-witnesses

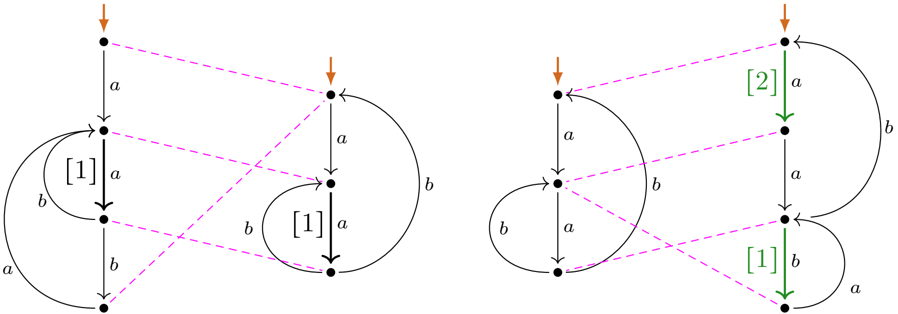




$P(a(a(b+ba))^* \cdot 0)$

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

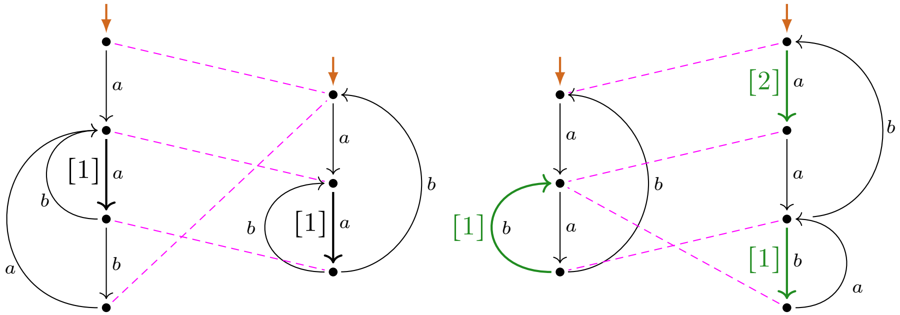
Collapsing LEE-witnesses



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

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

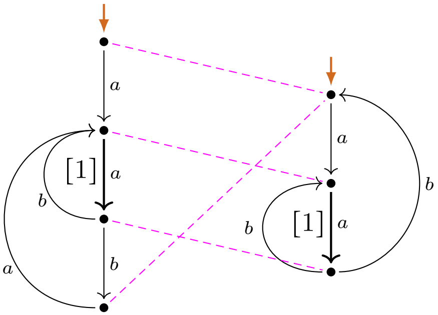
Collapsing LEE-witnesses

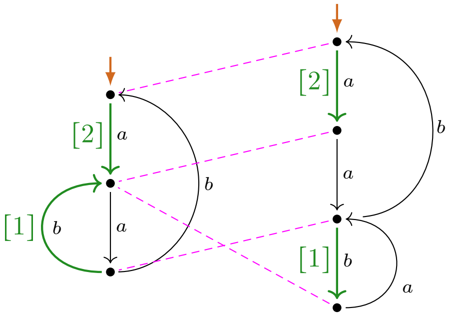


$P(a(a(b+ba))^* \cdot 0)$

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

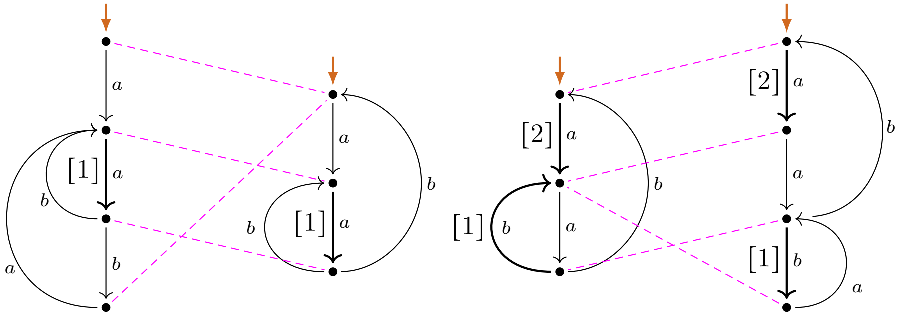
Collapsing LEE-witnesses





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

Collapsing LEE-witnesses



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

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

LEE under functional bisimulation / 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:*

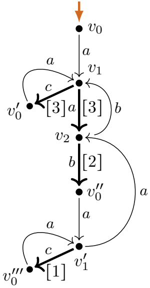
$$\mathsf{LEE}(G) \wedge G \textit{ has bisimulation collapse } C \implies \mathsf{LEE}(C)\,.$$

Idea of Proof for (i)

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)



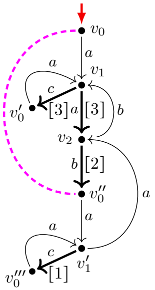
(C1.1)

Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)



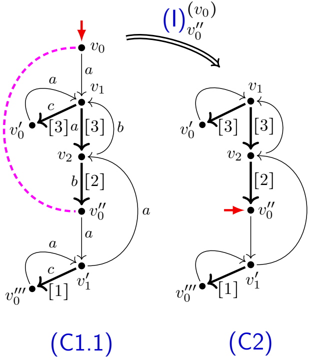
(C1.1)

Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

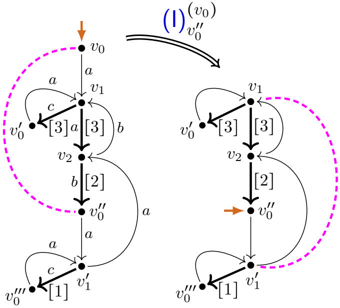


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)



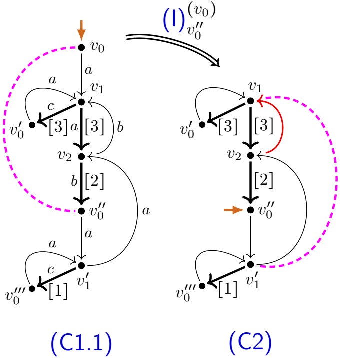
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

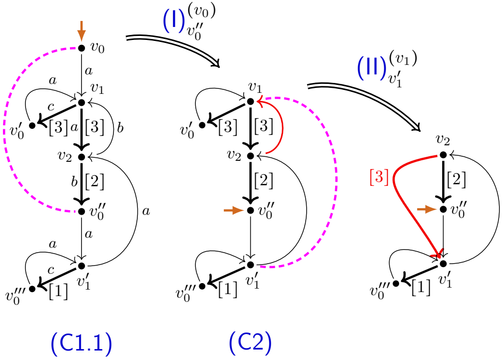


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

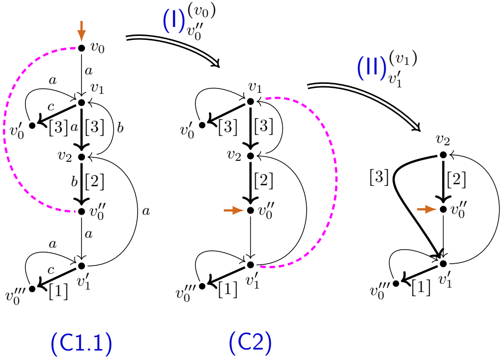


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

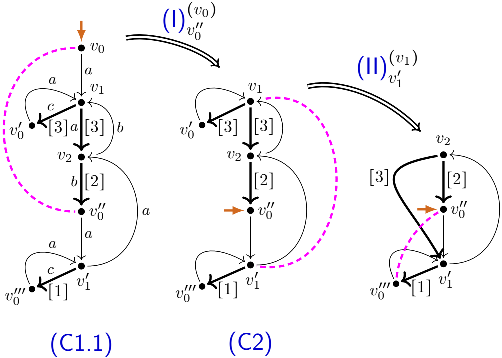


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

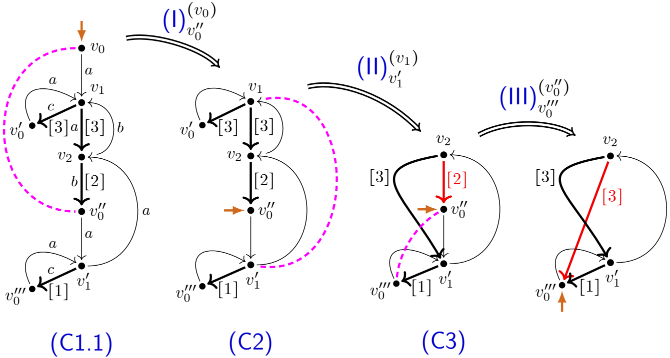


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

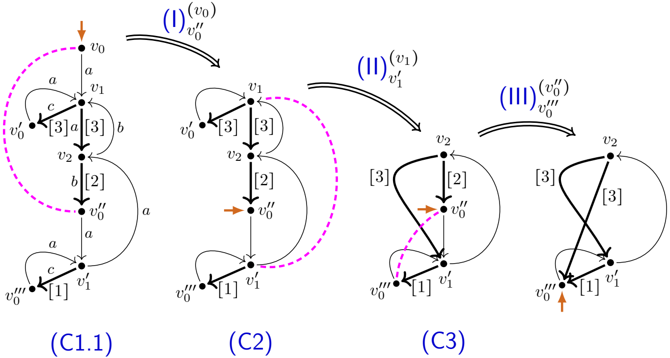


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)

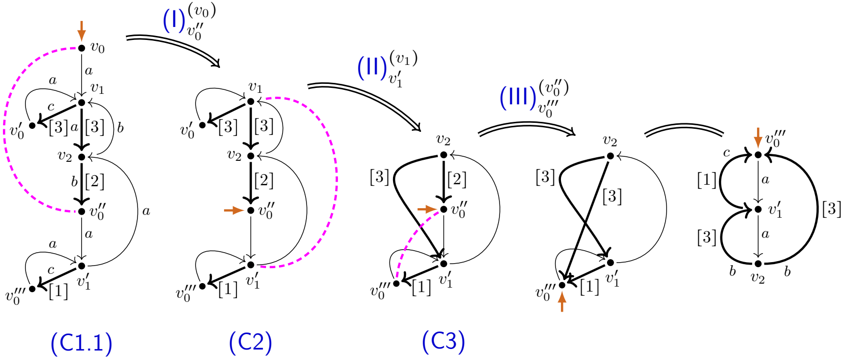


Lemma

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

LLEE-preserving collapse of LLEE-charts (G/Fokkink, LICS'20)

(no 1-transitions!)



Lemma

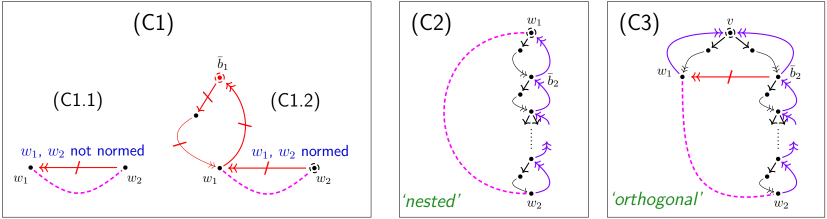
The bisimulation collapse of a LLEE-chart is again a LLEE-chart.

Reduced bisimilarity redundancies in LLEE-graphs (no 1-trans.!)

(G/Fokkink, LICS'20)

w_1, w_2 in different scc's

w_1, w_2 in the same scc



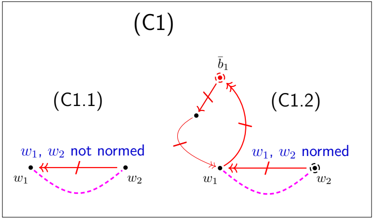
Lemma

Every not collapsed LLEE-graph contains bisimilar vertices $w_1 \neq w_2$ of kind (C1), (C2), or (C3) (a reduced bisimilarity redundancy $\langle w_1, w_2 \rangle$):

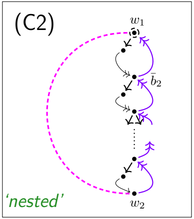
Reduced bisimilarity redundancies in LLEE-graphs (no 1-trans.!)

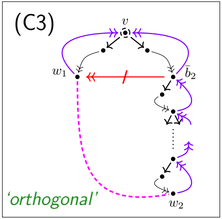
(G/Fokkink, LICS'20)

w_1, w_2 in different scc's



w_1, w_2 in the same scc





Lemma

Every not collapsed LLEE-graph contains bisimilar vertices $w_1 \neq w_2$ of kind (C1), (C2), or (C3) (a reduced bisimilarity redundancy $\langle w_1, w_2 \rangle$):

Lemma

Every reduced bisimilarity redundancy in a LLEE-graph can be eliminated LLEE-preservingly.

Properties of LEE-charts

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

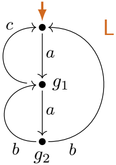
A process graph G
is $[\![\cdot]\!]_P$-expressible by an under-star-1-free regular expression
(i.e. P-expressible modulo bisimilarity by an $(\cancel{1}\backslash *)$ reg. expr.)
if and only if
the bisimulation collapse of G satisfies LEE.

Properties of LEE-charts

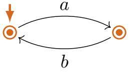
Theorem (⇐ G/Fokkink, 2020)

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.

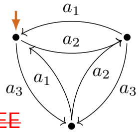
Hence $[\![\cdot]\!]_P$-expressible | **not** $[\![\cdot]\!]_P$-expressible by 1-free regular expressions:



LEE



~~LEE~~



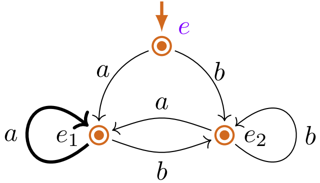
~~LEE~~

1-LEE

$\hat{=}$ sharing via 1-transitions facilitates LEE

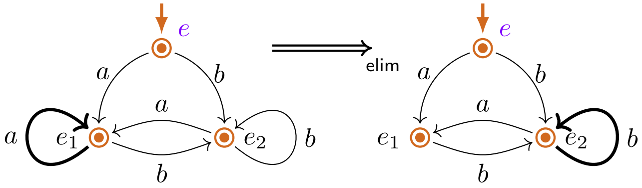
Failure of LEE in general (example)

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



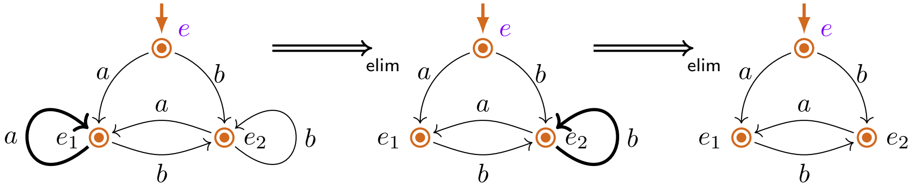
Failure of LEE in general (example)

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



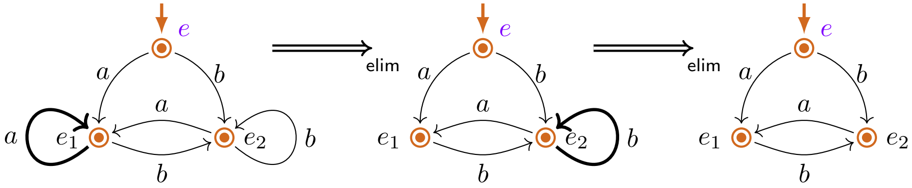
Failure of LEE in general (example)

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



Failure of LEE in general (example)

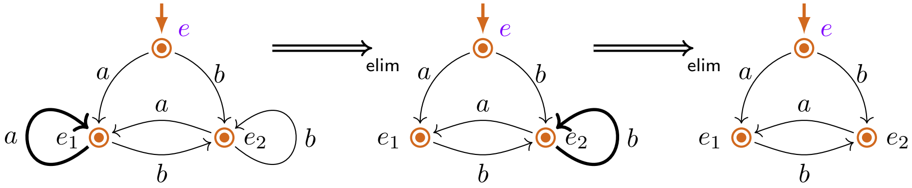
$P((a^* \cdot b^*)^*)$



no loop subchart,
but infinite paths

Failure of LEE in general (example)

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



~~LEE~~

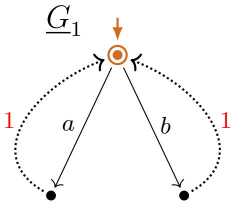
no loop subchart,
but infinite paths

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.

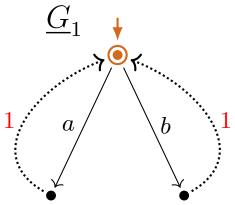


1-Graphs and induced graphs

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.



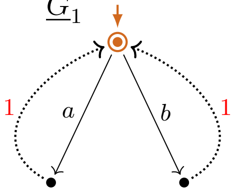
1-Graphs and induced graphs

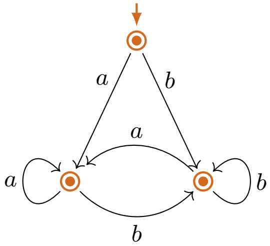
Definition

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

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

$\underline{G}_1$



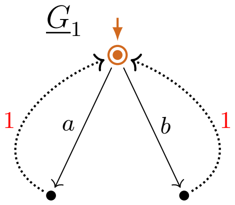


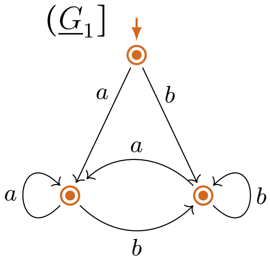
1-Graphs and induced graphs

Definition

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

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





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 $a \in A$

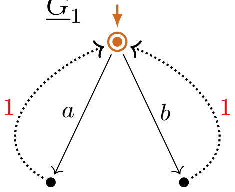
$v \xrightarrow{1} \cdot \cdots \cdot \xrightarrow{1} \cdot \Downarrow \quad \widehat{=} \quad v \Downarrow^{(1)}$ induced termination.

Definition

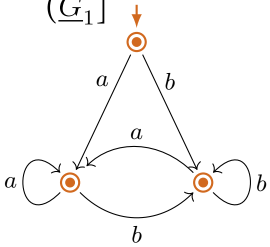
The induced (process) graph of a 1-graph $\underline{G} = \langle V, A, 1, v_s, \rightarrow, \Downarrow \rangle$ is:

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

$\underline{G}_1$



$(\underline{G}_1]$



1-Graphs and induced graphs

Definition

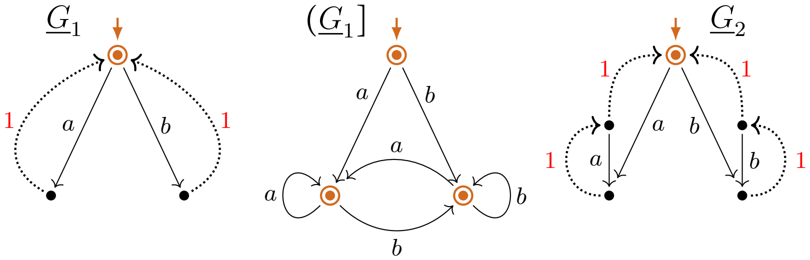
$$v_1 \xrightarrow{1} \cdot \; \cdots \; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \quad \widehat{=} \quad v_1 \xrightarrow{(a]} v_2$$ induced a-transitions, for $a \in A$

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

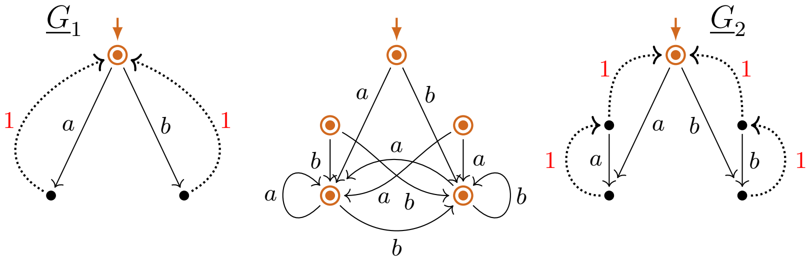
$$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \;\;\widehat{=}\;\; v_1 \xrightarrow{(a]} v_2$$ induced a-transitions, for $a \in A$

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

Definition

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

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



1-Graphs and induced graphs

Definition

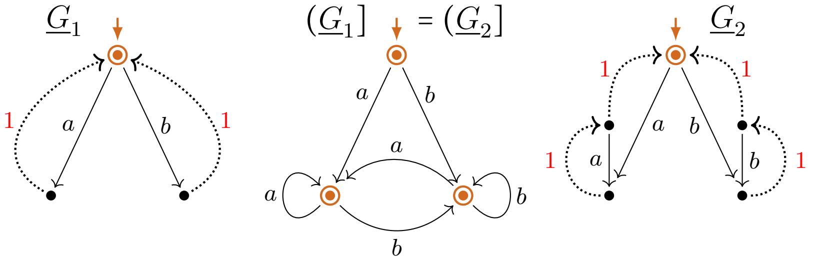
$v_1 \xrightarrow{1} \cdot \;\cdots\; \cdot \xrightarrow{1} \cdot \xrightarrow{a} v_2 \;\;\widehat{=}\;\; v_1 \xrightarrow{(a]} v_2$ induced a-transitions, for $a \in A$

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

Definition

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

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



1-LEE

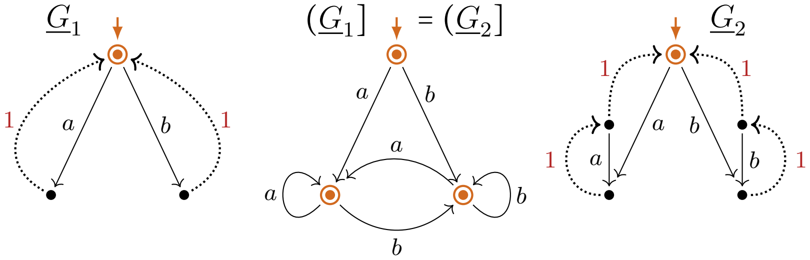
Definition

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

1-LEE

Definition

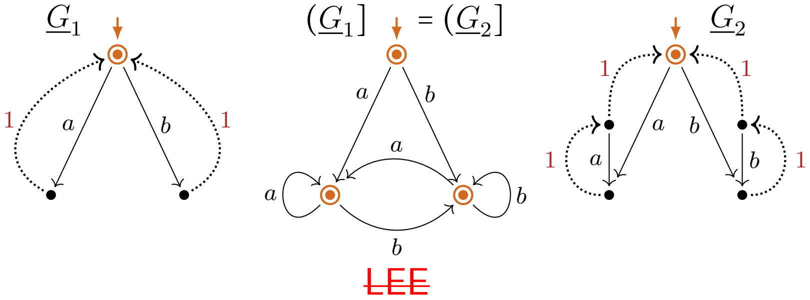
1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some weakly-guarded 1-graph $\underline{G}$.



1-LEE

Definition

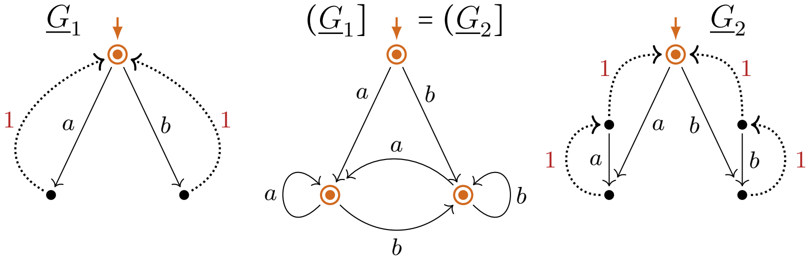
1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some weakly-guarded 1-graph $\underline{G}$.



1-LEE

Definition

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



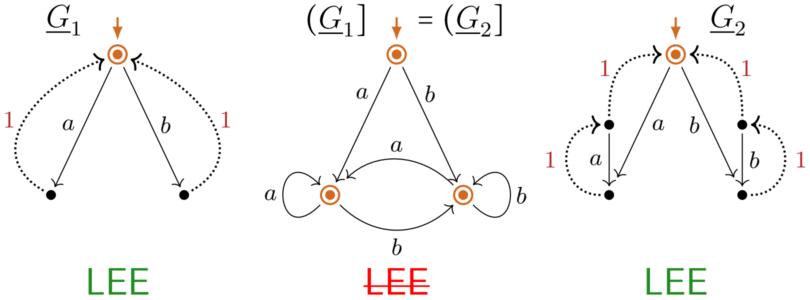
LEE

~~LEE~~

1-LEE

Definition

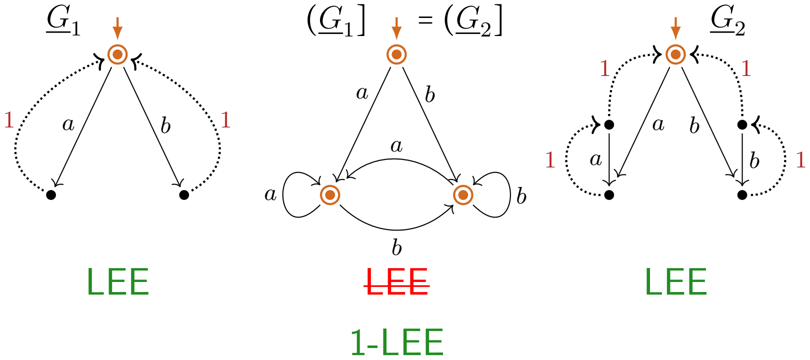
1-LEE(G) holds for a graph G,
if $G = (\underline{G}]$ for some weakly-guarded 1-graph $\underline{G}$.



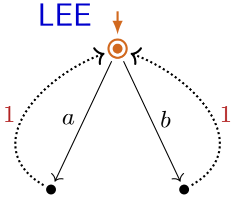
1-LEE

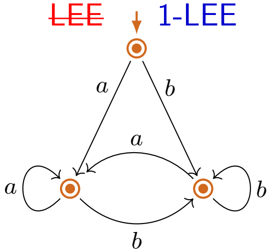
Definition

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



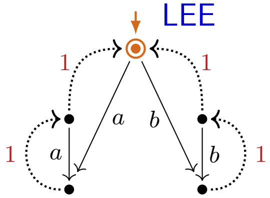
1-LEE holds for process interpretations





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

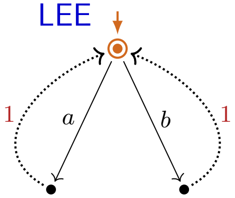
$$P((b^* \cdot a^*)^*)$$

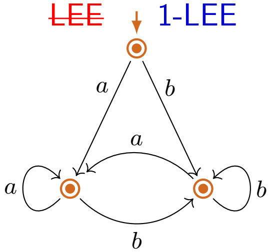


1-LEE holds for process interpretations

Lemma

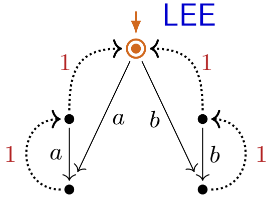
There is a 1-graph interpretation $\underline{P}$ of reg. expression e as 1-graphs $\underline{P}(e)$ such that for all $e \in RExp$: (i): LEE($\underline{P}(e)$), (ii): $(\underline{P}(e)] = P(e)$.





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

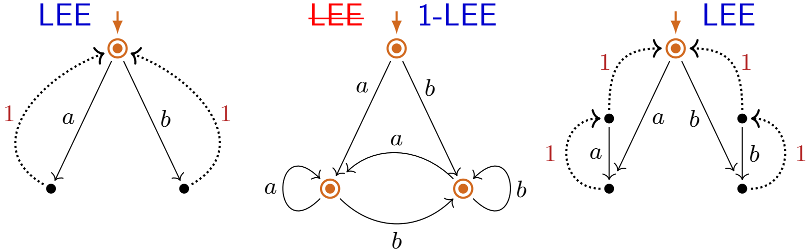
$P((b^* \cdot a^*)^*)$



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): $(\underline{P}(e)] = P(e)$.



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

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

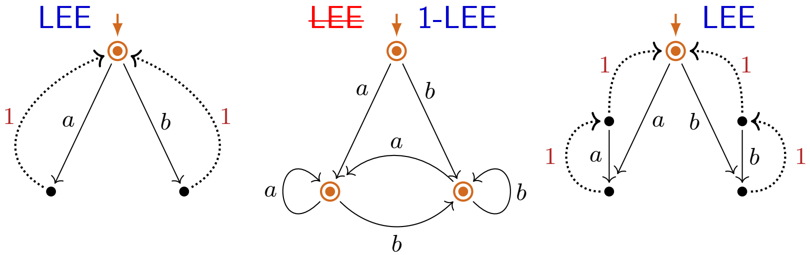
1-LEE holds for process interpretations

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): $\text{LEE}(\underline{P}(e))$, (ii): $(\underline{P}(e)] = P(e)$.

Theorem

$\text{1-LEE}(P(e))$ holds for all regular expressions e.



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

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

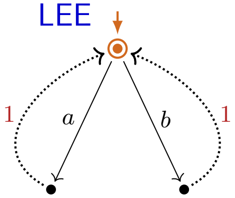
1-LEE holds for process interpretations

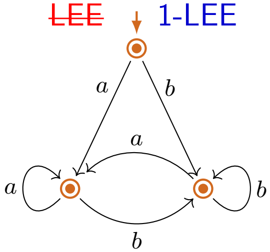
Lemma

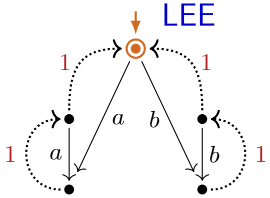
There is a 1-graph interpretation $\underline{P}$ of reg. expression e as 1-graphs $\underline{P}(e)$ such that for all $e \in RExp$: (i): LEE($\underline{P}(e)$), (ii): $(\underline{P}(e)] = P(e)$.

Theorem

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







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

Interpretation/extraction correspondences with 1-LEE

($\Leftarrow$ G 2021/22/23)

(Int)$_P$: *P-expressible graphs have the structural property* 1-LEE

Process **int**erpretations $P(e)$ of regular expressions e
are finite process graphs that satisfy 1-LEE.

(Extr)$_P$: 1-LEE *implies* $[\![\cdot]\!]_P$*-expressibility*

From every finite 1-process-graph $\underline{G}$ with 1-LEE
a regular expression e can be **extr**acted such that $\underline{G} \leftrightarrow P(e)$.

Interpretation/extraction correspondences with 1-LEE

($\Leftarrow$ G 2021/22/23)

(Int)$_P$: *P-expressible graphs have the structural property* 1-LEE

Process **int**erpretations $P(e)$ of regular expressions e
are finite process graphs that satisfy 1-LEE.

(Extr)$_P$: 1-LEE *implies* $[\![\cdot]\!]_P$*-expressibility*

From every finite 1-process-graph $\underline{G}$ with 1-LEE
a regular expression e can be **extr**acted such that $\underline{G} \leftrightarrow P(e)$.

~~**(Coll)**~~: 1-LEE *is not preserved under collapse*

The class of finite process graphs with 1-LEE
is not closed under bisimulation **coll**apse.

Interpretation/extraction correspondences of $P^{\bullet}$ with 1-LEE

(Int)$_{P^{\bullet}}$: *$P^{\bullet}$-expressible graphs satisfy* 1-LEE:

Compact process **i**nterpretations $P^{\bullet}(e)$ of regular expressions e
are finite process graphs that satisfy 1-LEE.

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

From every finite process graph G with 1-LEE
an regular expression uf can be **e**xtracted
such that $G \rightrightarrows P^{\bullet}(uf)$.

From every finite collapsed process graph G with 1-LEE
an regular expression e can be **e**xtracted
such that $G \simeq P^{\bullet}(e)$.

Interpretation/extraction correspondences of $P^{\bullet}$ with 1-LEE

(Int)$_{P^{\bullet}}$: *$P^{\bullet}$-expressible graphs satisfy* 1-LEE:

Compact process **i**nterpretations $P^{\bullet}(e)$ of regular expressions e
are finite process graphs that satisfy 1-LEE.

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

From every finite process graph G with 1-LEE
an regular expression uf can be **e**xtracted
such that $G \mathrel{\underline{\to}} P^{\bullet}(uf)$.

From every finite collapsed process graph G with 1-LEE
an regular expression e can be **e**xtracted
such that $G \simeq P^{\bullet}(e)$.

~~**(ImColl)**$_{P^{\bullet}}$~~: The image of $P^{\bullet}$ is not closed under bisimulation **c**ollapse .

1-LEE/ LEE characterize

the un-/restricted image of $P^{\bullet}$

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

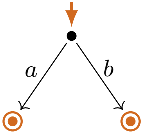
$P(uf)$

$P(uf)$

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

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

$P(uf)$



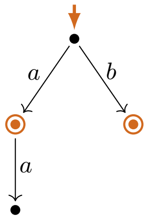
$P(uf)$

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

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

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

$P(uf)$



$P(uf)$

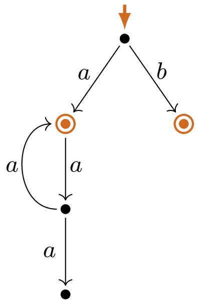
$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

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

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

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

$P(uf)$



$P(uf)$

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

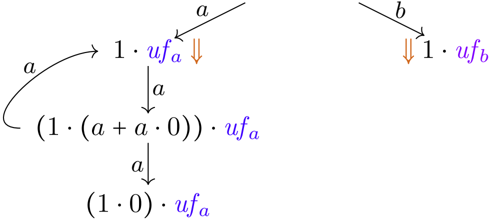
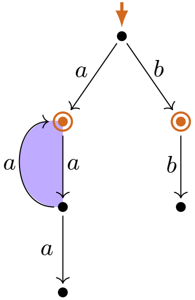


Image of P is **not** closed under bisimulation collapse

$P(uf)$



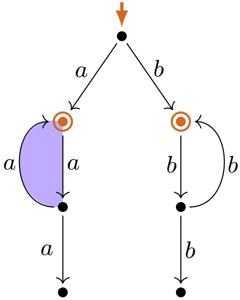
$P(uf)$

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

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

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

$P(uf)$



$P(uf)$

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

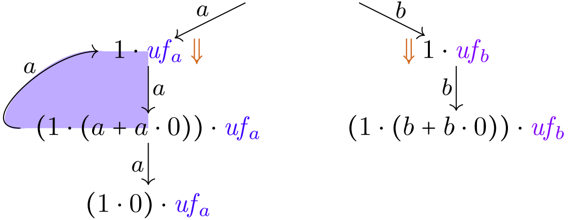
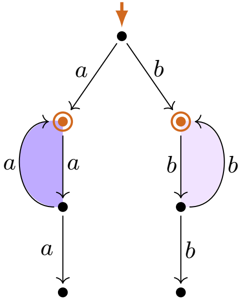


Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$

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

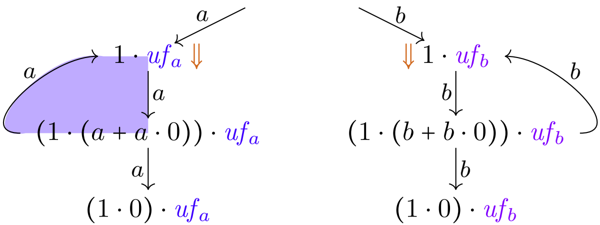
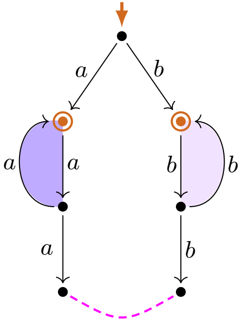
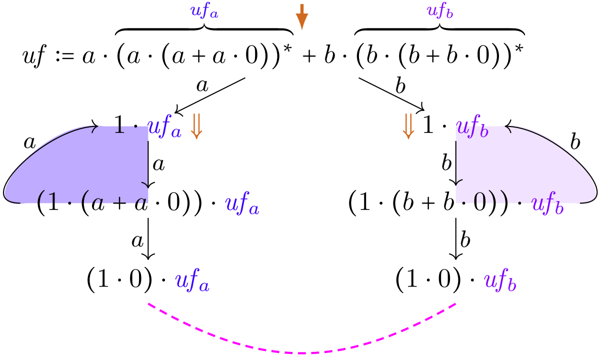


Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$



Compact process interpretation $P^{\bullet}$

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

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

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

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

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 $\mathcal{T}^{\bullet}$.

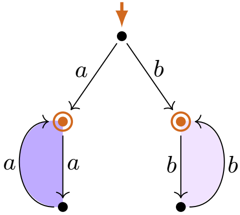
Lemma ($P^{\bullet}$ increases sharing; $P^{\bullet}$, P have same bisimulation semantics)

(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.

Image of $P^{\bullet}$ under bisimulation collapse ...

$P^{\bullet}(uf)$



$P^{\bullet}(uf)$

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

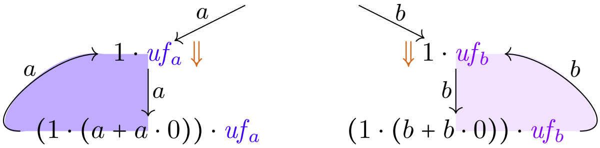
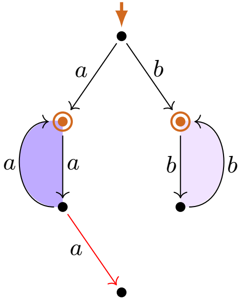


Image of $P^{\bullet}$ under bisimulation collapse . . .

$P^{\bullet}(uf)$



$P^{\bullet}(uf)$

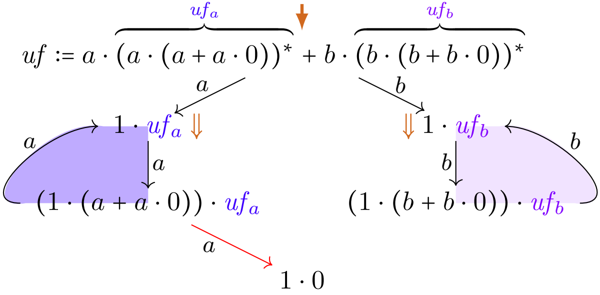
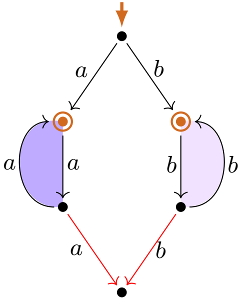


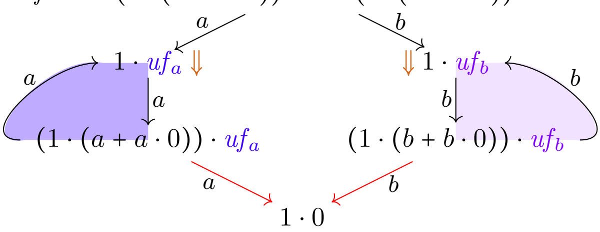
Image of $P^\bullet$ under bisimulation collapse . . .

$P^\bullet(uf)$



$P^\bullet(uf)$

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



Interpretation correspondence of $P^{\bullet}$ with LEE

$(\textbf{Int})^{(*/\cancel{1})}_{P^{\bullet}}$: *By under-star-1-free expressions $P^{\bullet}$-expressible graphs satisfy* LEE:

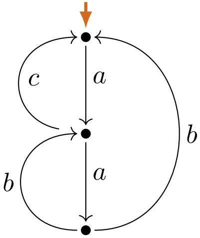
Compact process **i**nterpretations $P^{\bullet}(uf)$
of under-star-1-free regular expressions uf
are finite process graphs that satisfy LEE.

$(\textbf{Extr})^{(*/\cancel{1})}_{P^{\bullet}}$: LEE *implies $[\![\cdot]\!]_P$-expressibility by under-star-1-free reg. expr's*:

From every finite process graph G with LEE
an under-star-1-free regular expression uf can be **e**xtracted
such that $G \Rightarrow P(uf)$.

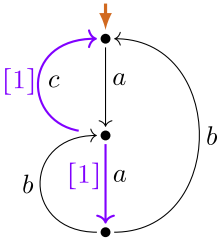
Refined extraction expression (example)

G_4



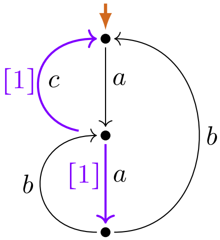
Refined extraction expression (example)

$\widehat{G_4}$



Refined extraction expression (example)

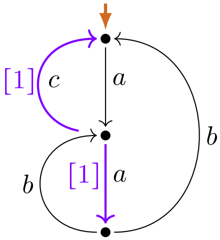
$\widehat{G_4}$



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

Refined extraction expression (example)

$\widehat{G_4}$

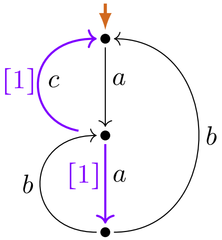


a

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

Refined extraction expression (example)

$\widehat{G_4}$



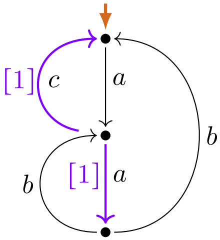
$$((1 \cdot a) \cdot (\qquad\qquad)^{*}) \cdot 0$$

$$\downarrow a$$

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

Refined extraction expression (example)

$\widehat{G_4}$



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

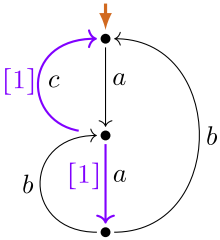
$$\downarrow a$$

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

b $\qquad$ b

Refined extraction expression (example)

$\widehat{G_4}$



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

$$\downarrow a$$

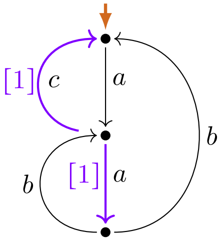
$$(1 \cdot (\qquad)^*) \cdot 0$$

$$b \qquad b$$

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

Refined extraction expression (example)

$\widehat{G_4}$



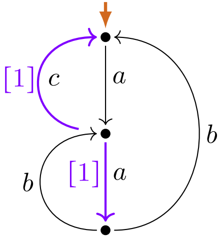
$$((1 \cdot a) \cdot (\qquad)^*) \cdot 0$$

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

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

Refined extraction expression (example)

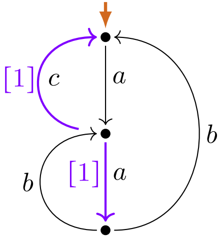
$\widehat{G_4}$



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

Refined extraction expression (example)

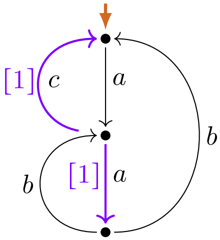
$\widehat{G_4}$



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

Refined extraction expression (example)

$\widehat{G_4}$

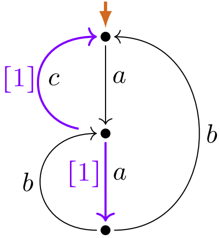


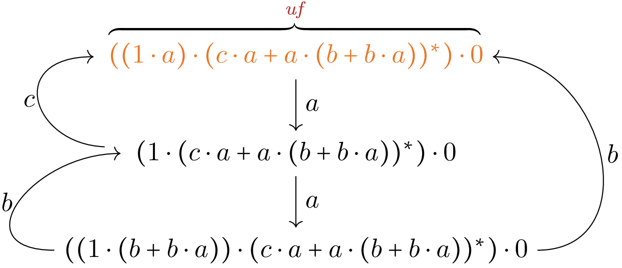
$$\begin{tikzcd}
((1\cdot a)\cdot(c\cdot a+a\cdot(b+b\cdot a))^{*})\cdot 0 \arrow[d, "a"] \\
(1\cdot(c\cdot a+a\cdot(b+b\cdot a))^{*})\cdot 0 \arrow[d, "a"] \arrow[u, bend left=60, "c"] \\
((1\cdot(b+b\cdot a))\cdot(c\cdot a+a\cdot(b+b\cdot a))^{*})\cdot 0 \arrow[u, bend left=60, "b"] \arrow[uu, bend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

Refined extraction expression (example)

$\widehat{G_4}$

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





Interpretation/extraction correspondences of $P^{\bullet}$ with LEE

(Int)$^{(*/\cancel{1})}_{P^{\bullet}}$: *By under-star-1-free expressions $P^{\bullet}$-expressible graphs satisfy* LEE:

Compact process **i**nterpretations $P^{\bullet}(uf)$
of under-star-1-free regular expressions uf
are finite process graphs that satisfy LEE.

(Extr)$^{(*/\cancel{1})}_{P^{\bullet}}$: LEE *implies $[\![\cdot]\!]_P$-expressibility by under-star-1-free reg. expr's*:

From every finite process graph G with LEE
an under-star-1-free regular expression uf can be **e**xtracted
such that $G \mathrel{\underline{\to}} P^{\bullet}(uf)$.

From every finite collapsed process graph G with LEE
an under-star-1-free regular expression uf can be **e**xtracted
such that $G \simeq P^{\bullet}(uf)$.

Interpretation/extraction correspondences of $P^{\bullet}$ with LEE

(Int)$_{P^{\bullet}}^{(*/\cancel{1})}$: *By under-star-1-free expressions $P^{\bullet}$-expressible graphs satisfy* LEE:

Compact process **i**nterpretations $P^{\bullet}(uf)$
of under-star-1-free regular expressions uf
are finite process graphs that satisfy LEE.

(Extr)$_{P^{\bullet}}^{(*/\cancel{1})}$: LEE *implies $[\![\cdot]\!]_P$-expressibility by under-star-1-free reg. expr's*:

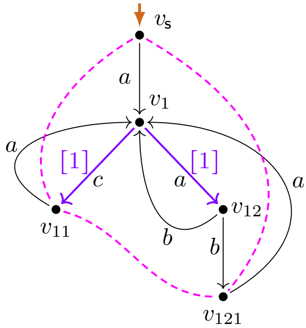
From every finite process graph G with LEE
an under-star-1-free regular expression uf can be **e**xtracted
such that $G \Rightarrow P^{\bullet}(uf)$.

From every finite collapsed process graph G with LEE
an under-star-1-free regular expression uf can be **e**xtracted
such that $G \simeq P^{\bullet}(uf)$.

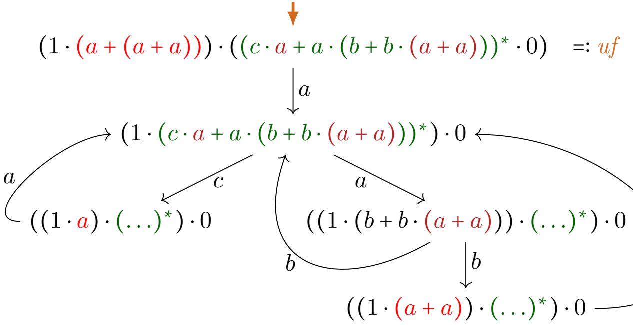
(ImColl)$_{P^{\bullet}}^{(*/\cancel{1})}$: The image of $P^{\bullet}$,
restricted to under-star-1-free regular expressions,
is closed under bisimulation **c**ollapse.

Adapted (refined) extraction from LLEE-graph

$G_1 \,/\, \widehat{G_1}$



$P^\bullet(uf) = P(uf) \simeq G_1$



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

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

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 $e \in RExp^{(*/\cancel{1})}$).

(iii) G is isomorphic to a graph in the image of $P^{\bullet}$ on $(*/\cancel{1})$ reg. expr's (i.e. $G \simeq G'$ for some $G' \in im(P^{\bullet}|_{RExp^{(*/\cancel{1})}})$).

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

My next aims

Completeness problem, solution (journal articles):

A1: graph structure of regular expression processes (LEE/1-LEE)

A2: motivation of crystallization

A4: details of crystallization procedure,
and completeness of Milner's proof system

Expressibility problem

A3: LEE is decidable in polynomial time (conference article).

Q: Is 1-LEE decidable in polynomial time**?**

P: Is expressibility by a regular expression, for a finite process graph,
decidable in polynomial time/fixed-parameter tractable time**?**

Resources

- Slides/abstract on `clegra.github.io`
  - slides: `.../lf/IFIP-1_6-2024.pdf`
  - abstract: `.../lf/abstract-IFIP-1_6-2024.pdf`
- CG: Closing the Image of the Process Interpretation of 1-Free Regular Expressions Under Bisimulation Collapse
  - TERMGRAPH 2024, extended abstract.
- CG: The Image of the Process Interpretation of Regular Expressions is Not Closed under Bisimulation Collapse,
  - arXiv:2303.08553, 2021/2023.
- CG: Milner's Proof System for Regular Expressions Modulo Bisimilarity is Complete,
  - LICS 2022, arXiv:2209.12188, poster.
- 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.

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)*

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

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

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

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

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

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

$$[\![e]\!]_L := L(e) \quad (\text{language defined by } e)$$