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

Process interpretation $P(\cdot)$ 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

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

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

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

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

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

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

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

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

$\mathbf{0} \;\overset{P}{\longmapsto}\;$ deadlock δ, no termination

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

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

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

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

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

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

Process interpretation P (formal definition)

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

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

Process interpretation P (formal definition)

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 (formal definition)

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

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

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

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

Process interpretation P (formal definition)

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 (formal definition)

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

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

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

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

Definition

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

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

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)

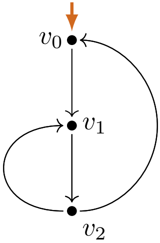
Definition

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)

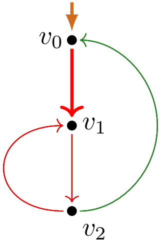
Definition

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)

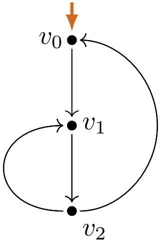
Definition

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)

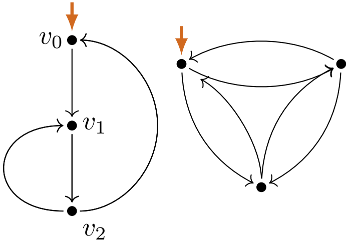
Definition

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)

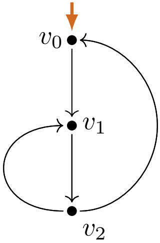
Definition

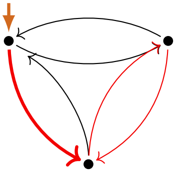
A process graph is a loop graph if:

(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)

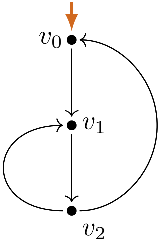
Definition

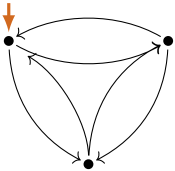
A process graph is a loop graph if:

(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)

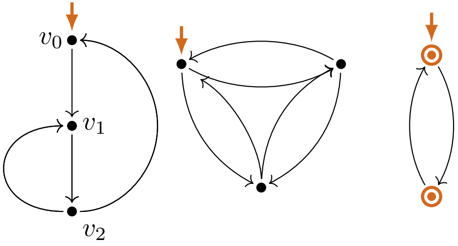
Definition

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)

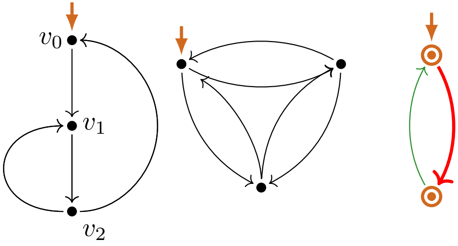
Definition

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)

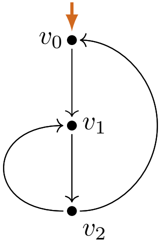
Definition

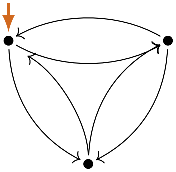
A process graph is a loop graph if:

(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)

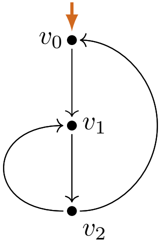
Definition

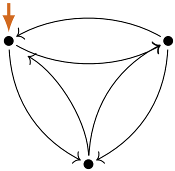
A process graph is a loop graph if:

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

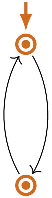
(L2) Every infinite path from the start vertex returns to it.

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





(L1),~~(L2)~~



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

Loop graphs (interpretations of innermost iterations without 1)

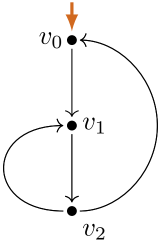
Definition

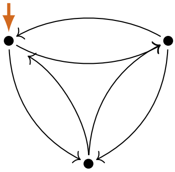
A process graph is a loop graph if:

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

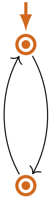
(L2) Every infinite path from the start vertex returns to it.

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

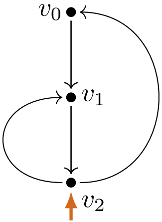




(L1),~~(L2)~~



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



Loop graphs (interpretations of innermost iterations without 1)

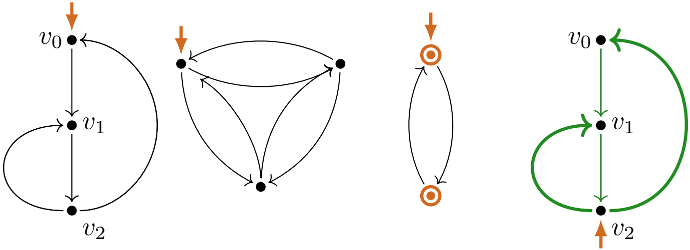
Definition

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)

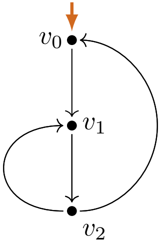
Definition

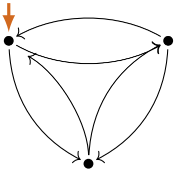
A process graph is a loop graph if:

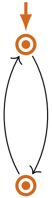
(L1) There is an infinite path from the start vertex.

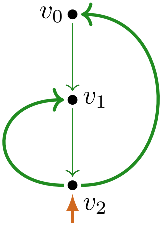
(L2) Every infinite path from the start vertex returns to it.

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









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

Loop graphs (interpretations of innermost iterations without 1)

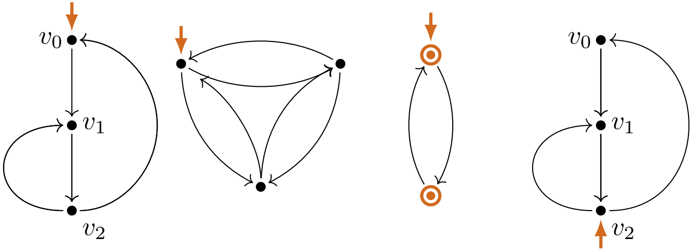
Definition

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)

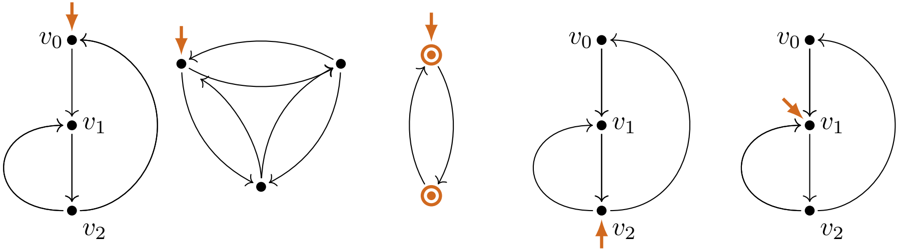
Definition

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)

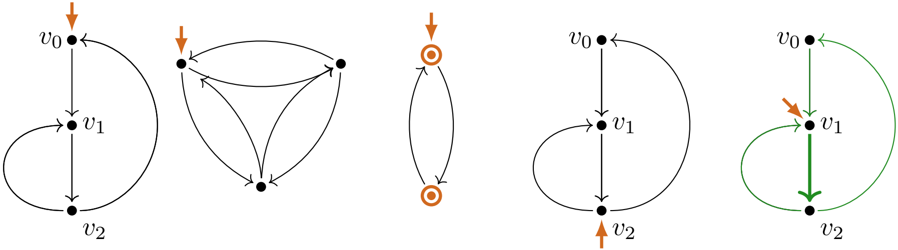
Definition

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)

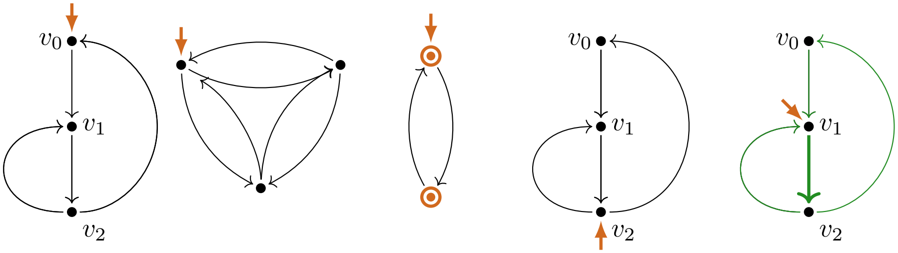
Definition

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)

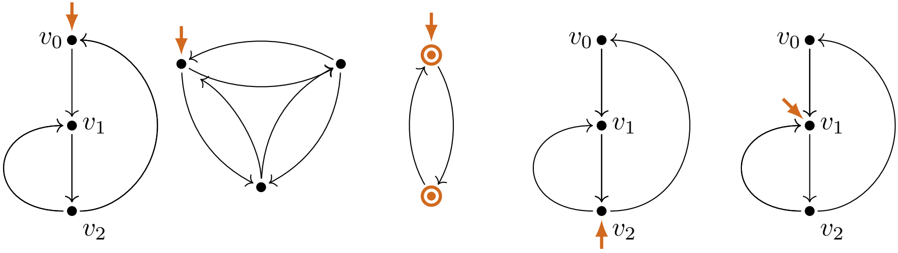
Definition

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)

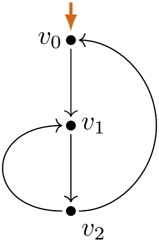
Definition

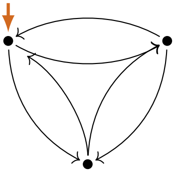
A process graph is a loop graph if:

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

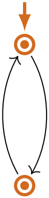
(L2) Every infinite path from the start vertex returns to it.

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

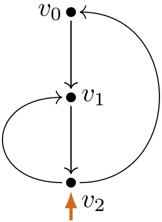




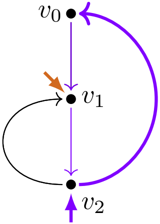
(L1),~~(L2)~~



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



loop chart



loop subchart

Loop elimination, 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.*

Loop elimination, and properties

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

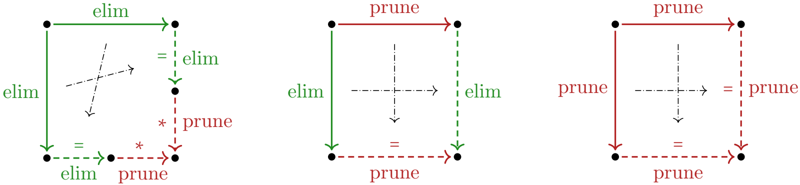
- removing the loop-entry transition(s)
- garbage collection

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

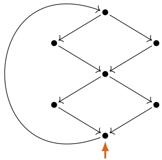
Lemma

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

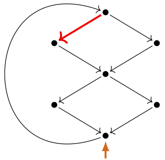
(ii) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is decreasing* [Van Oostrom, de Bruijn]



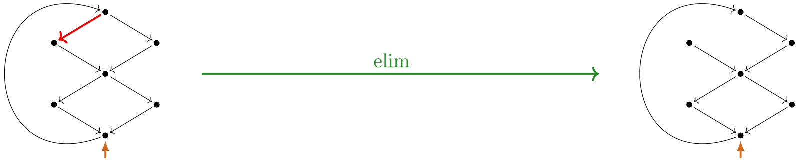
'Critical pair': bi-loop elimination



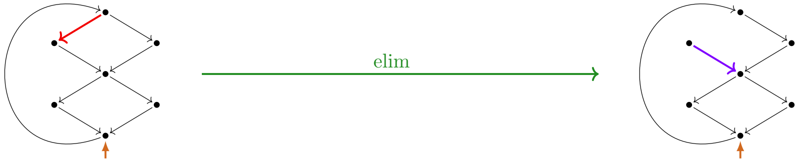
'Critical pair': bi-loop elimination



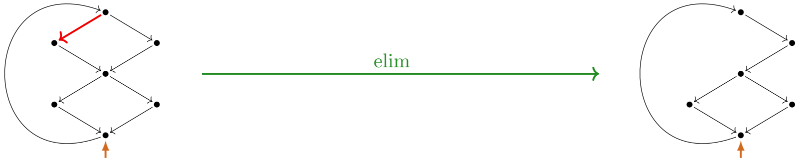
'Critical pair': bi-loop elimination



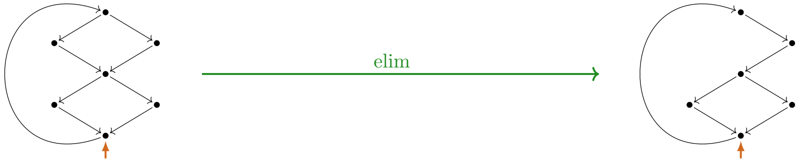
'Critical pair': bi-loop elimination



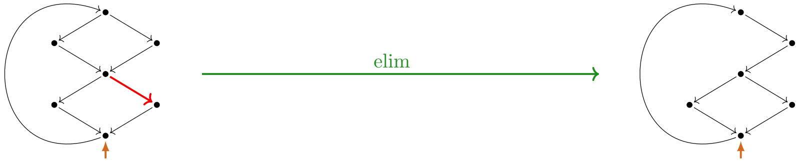
'Critical pair': bi-loop elimination



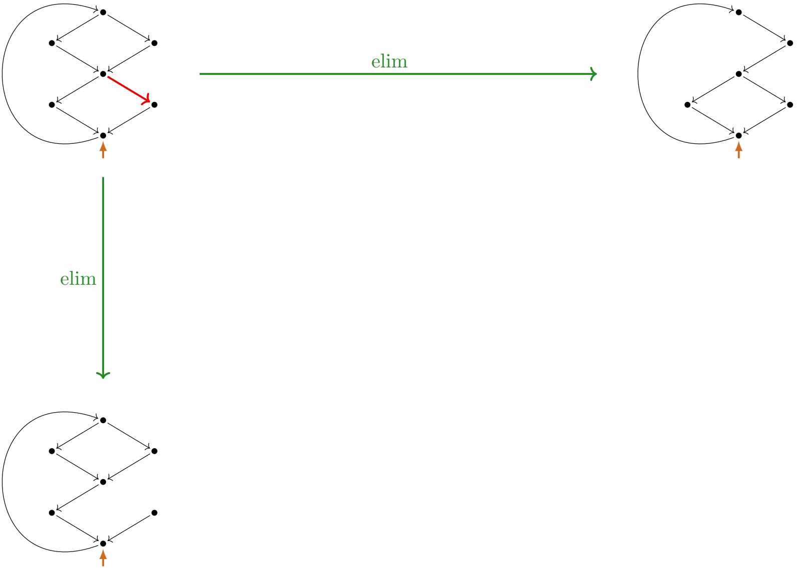
'Critical pair': bi-loop elimination



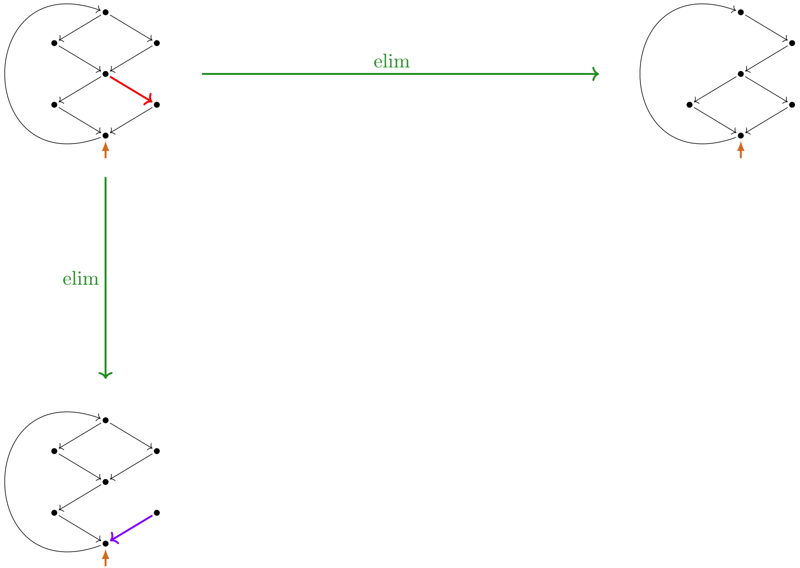
'Critical pair': bi-loop elimination



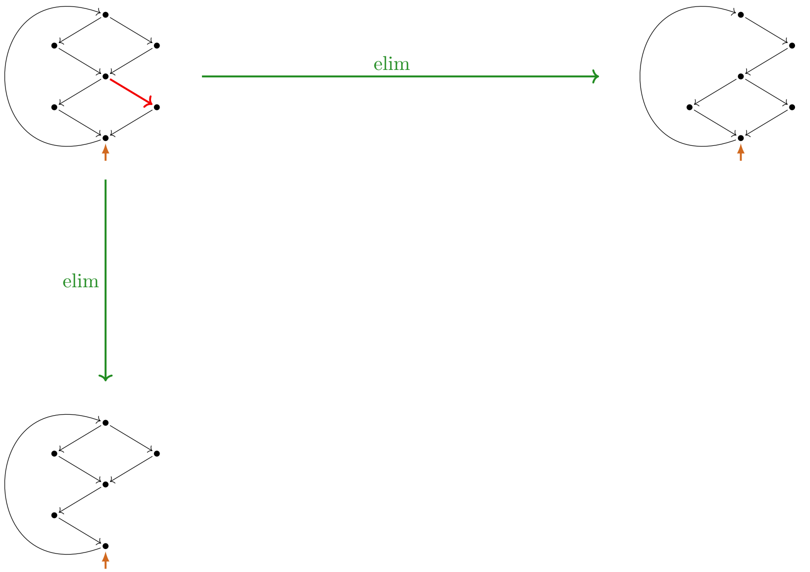
'Critical pair': bi-loop elimination



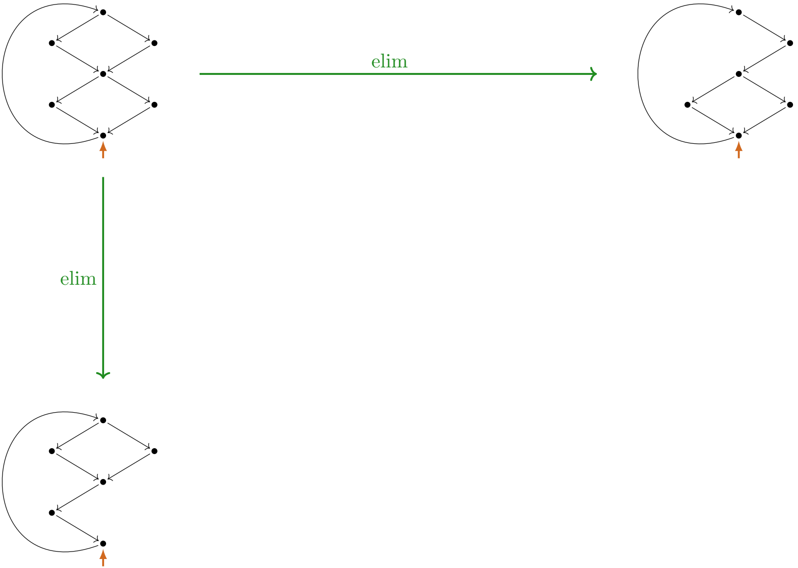
‘Critical pair’: bi-loop elimination



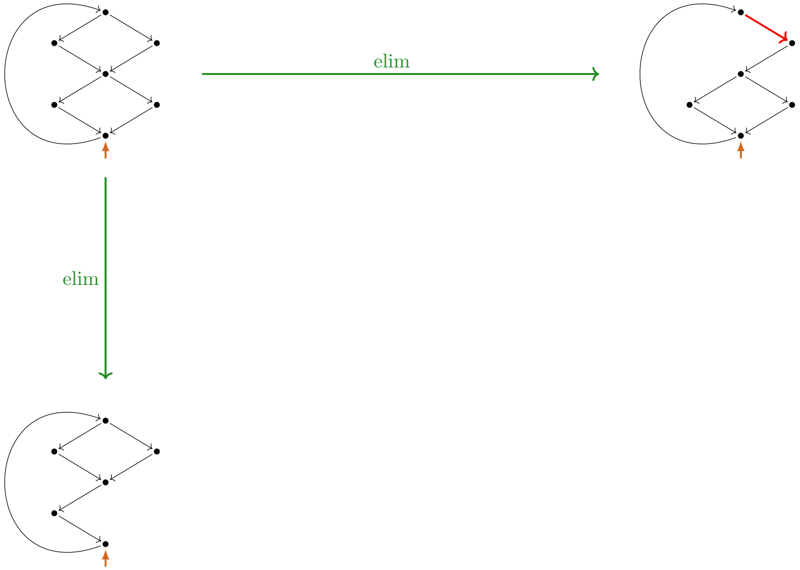
'Critical pair': bi-loop elimination



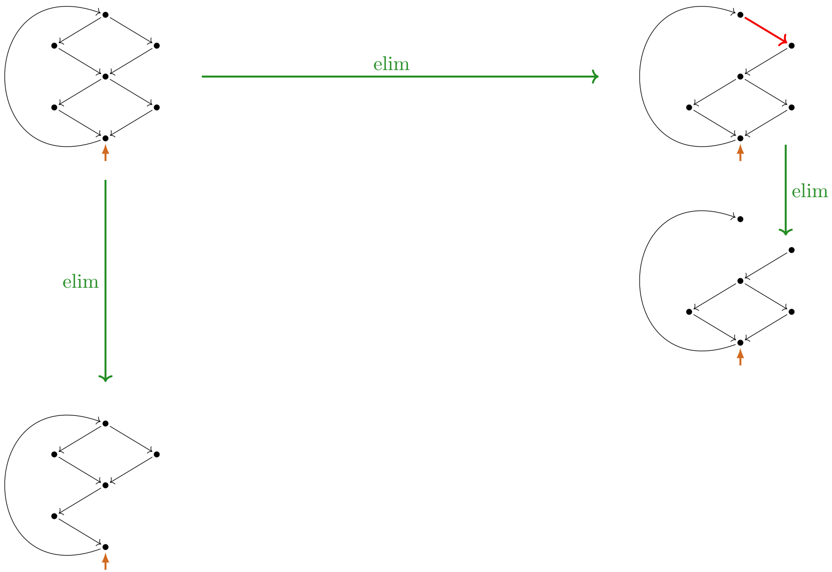
'Critical pair': bi-loop elimination



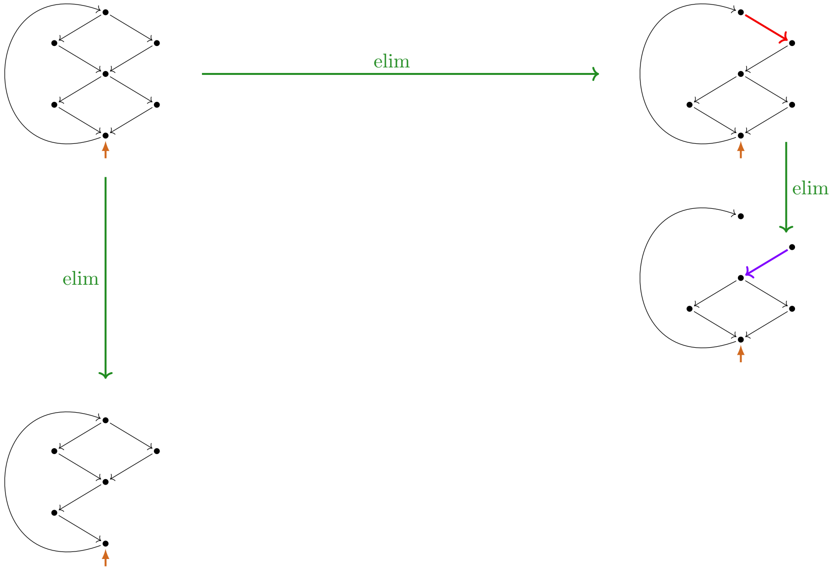
'Critical pair': bi-loop elimination



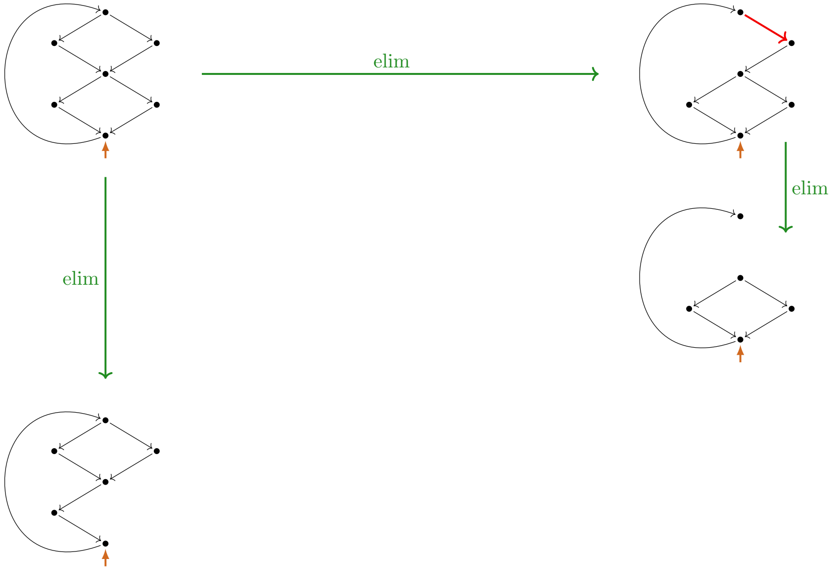
'Critical pair': bi-loop elimination



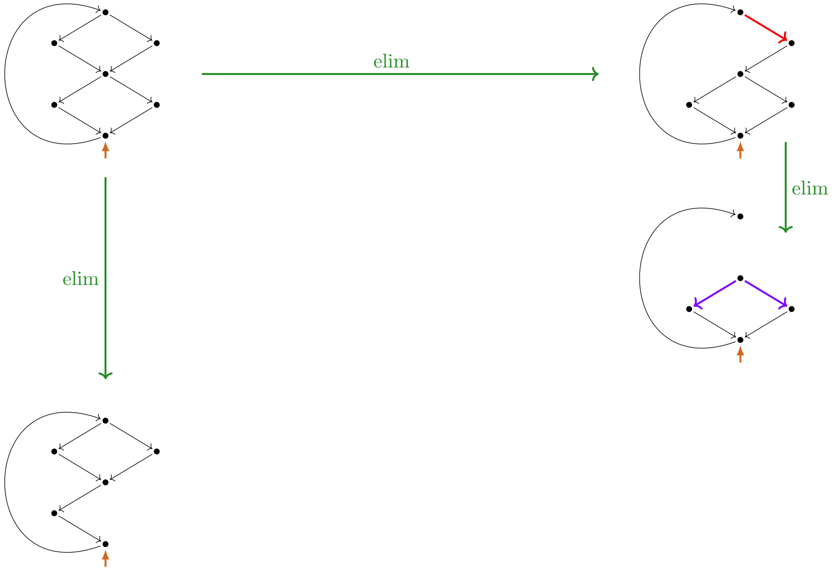
'Critical pair': bi-loop elimination



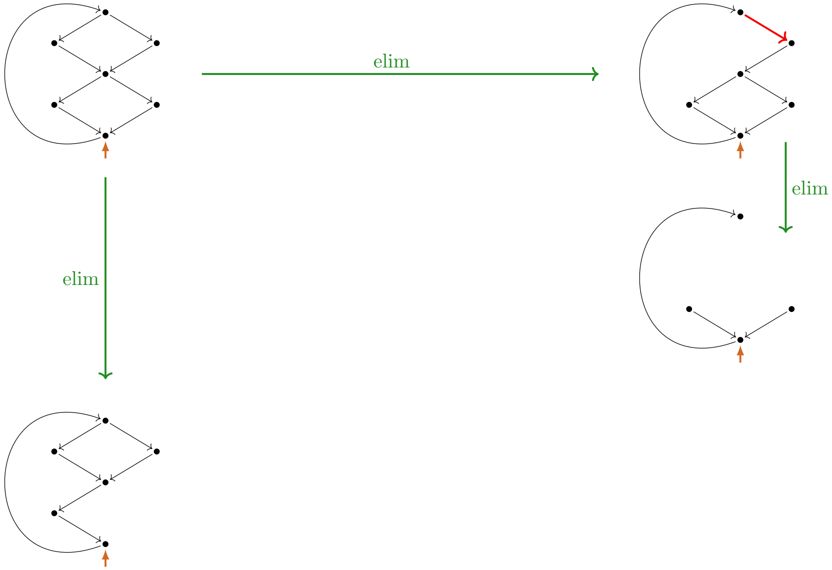
'Critical pair': bi-loop elimination



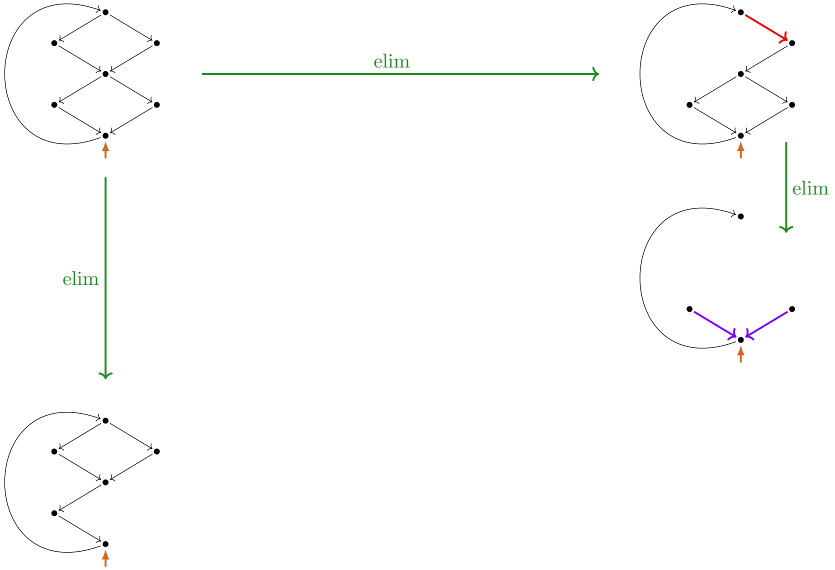
'Critical pair': bi-loop elimination



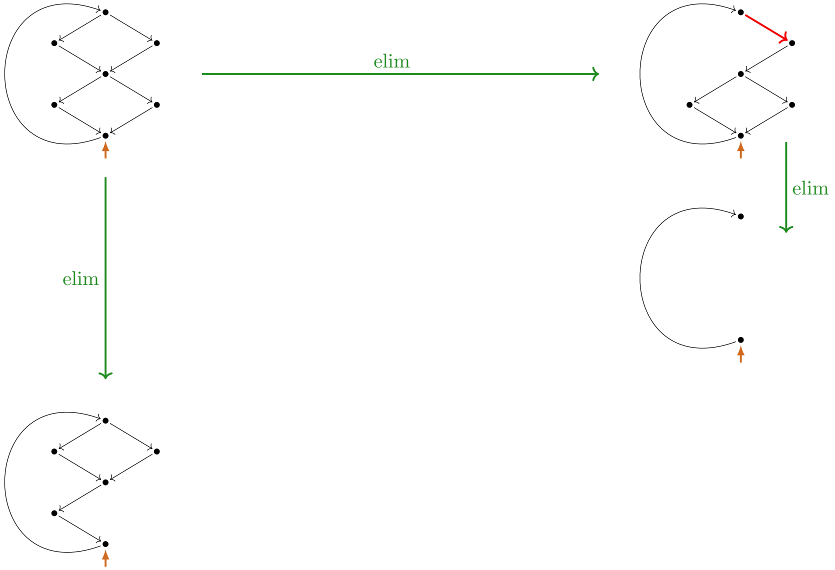
‘Critical pair’: bi-loop elimination



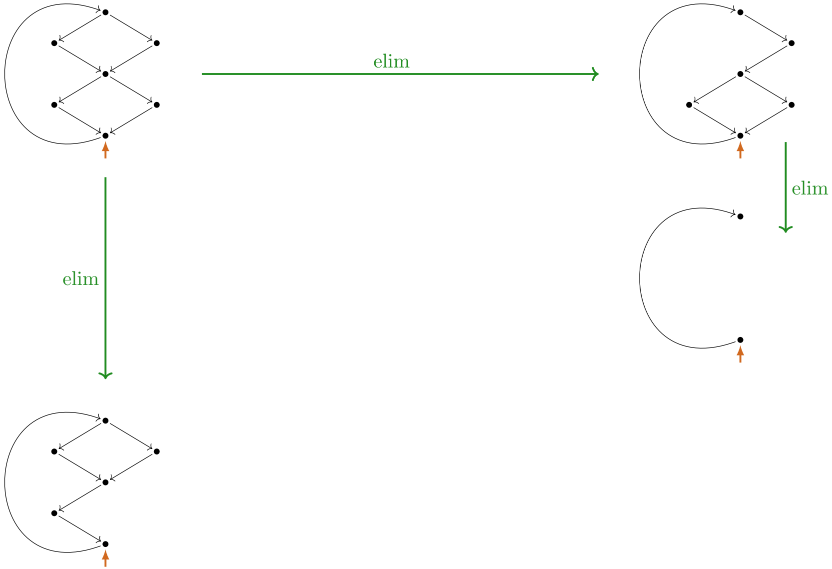
'Critical pair': bi-loop elimination



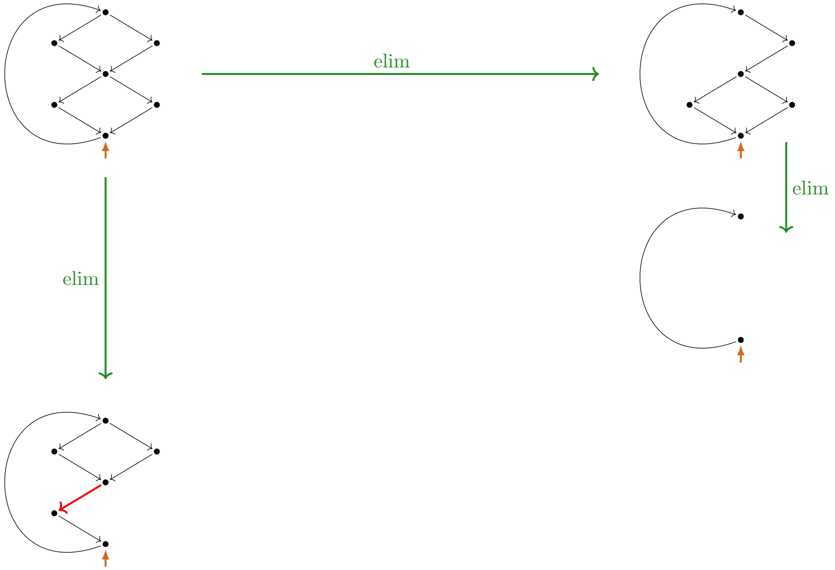
‘Critical pair’: bi-loop elimination



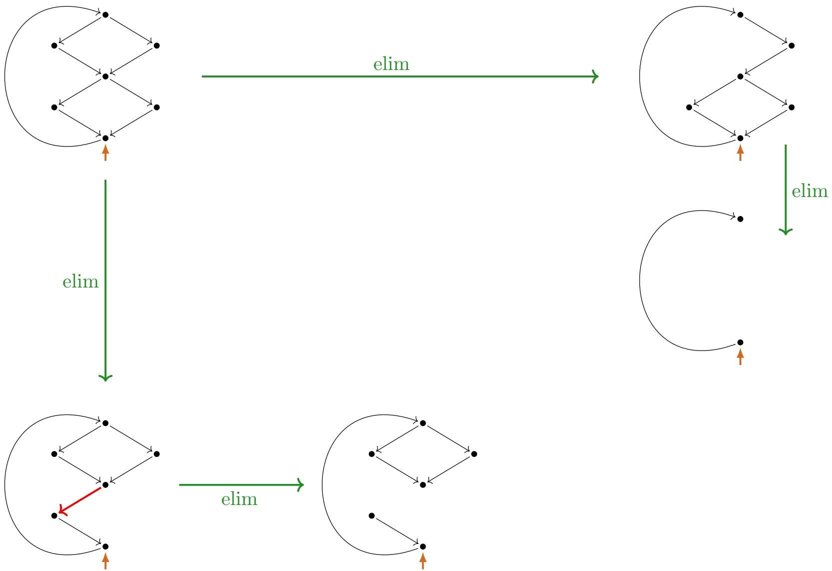
‘Critical pair’: bi-loop elimination



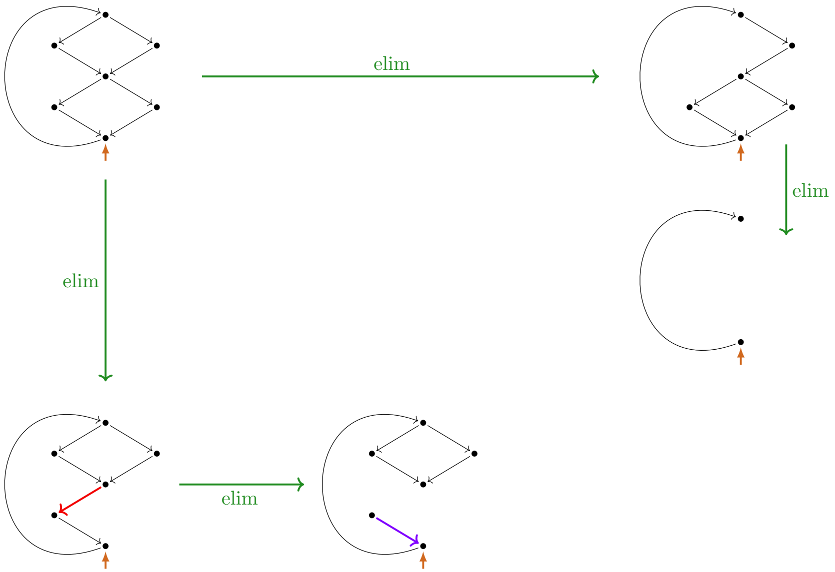
'Critical pair': bi-loop elimination



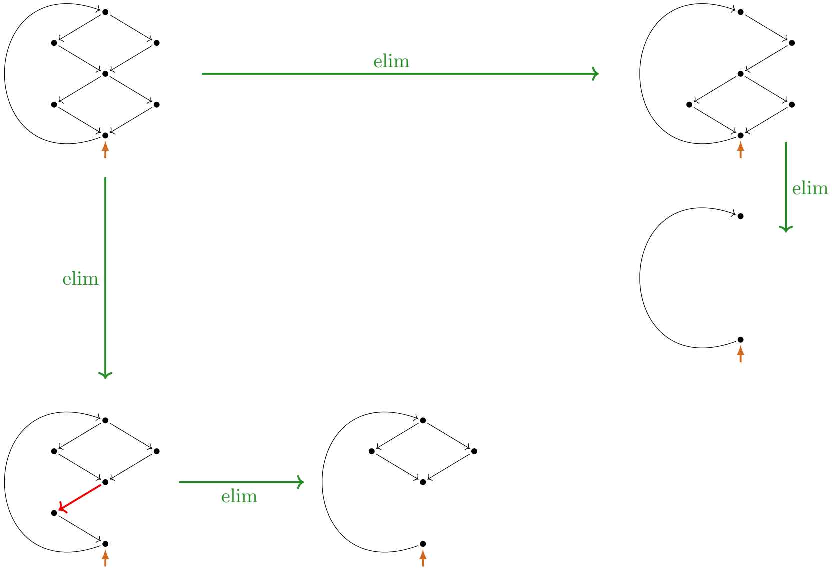
'Critical pair': bi-loop elimination



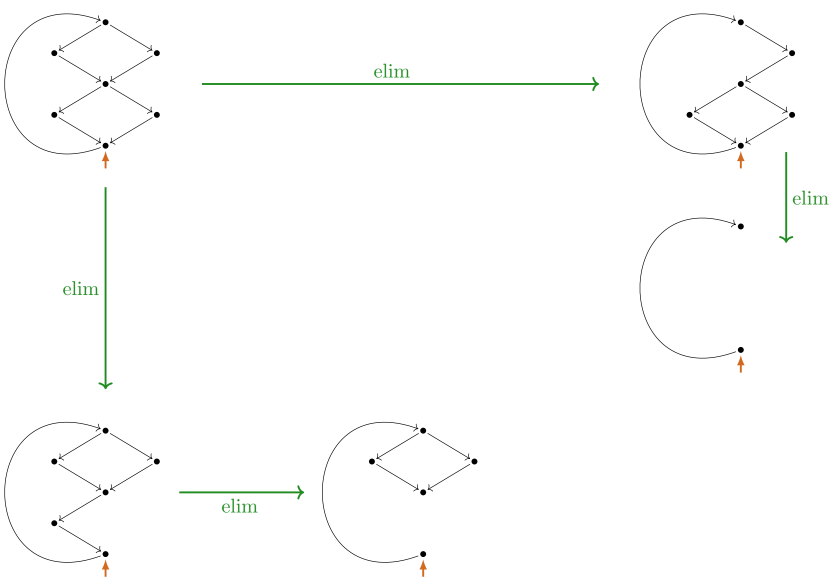
'Critical pair': bi-loop elimination



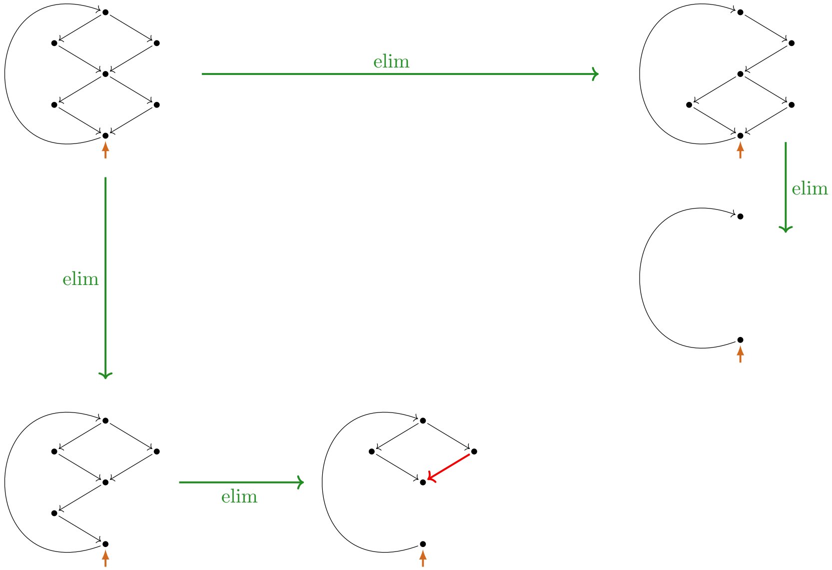
‘Critical pair’: bi-loop elimination



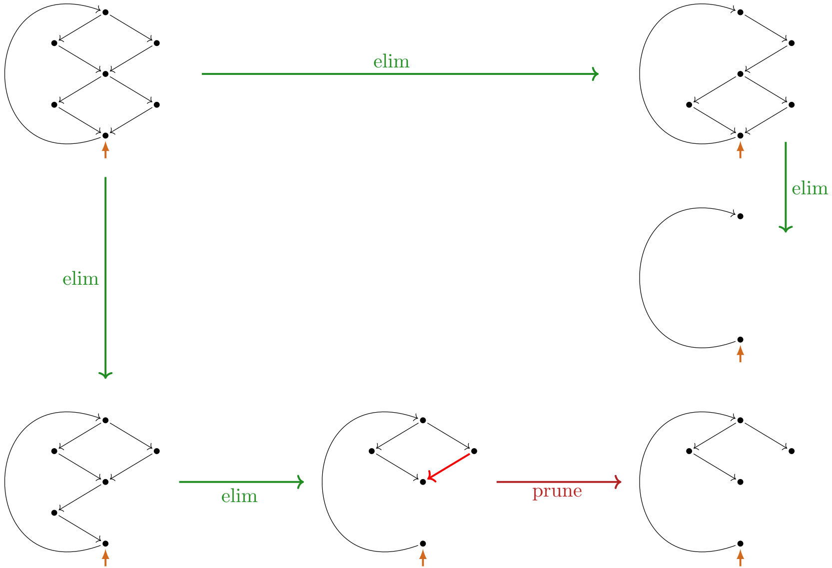
'Critical pair': bi-loop elimination



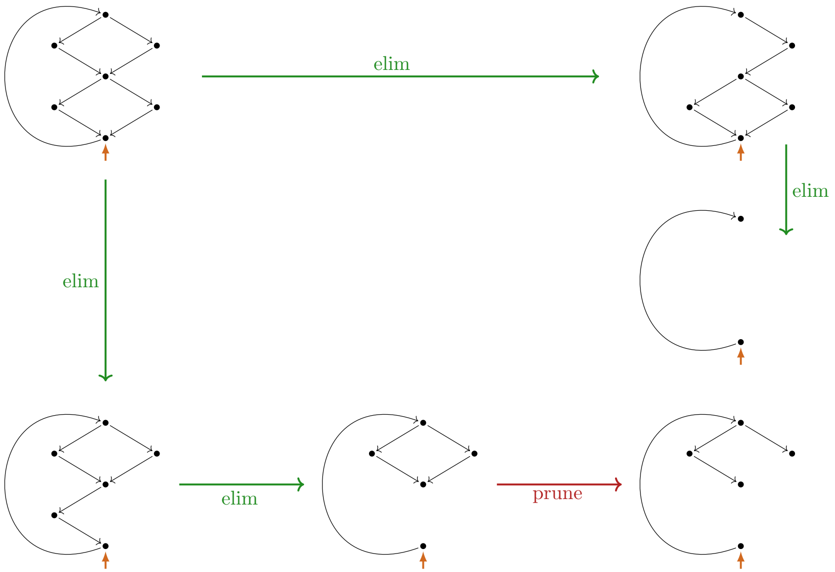
‘Critical pair’: bi-loop elimination



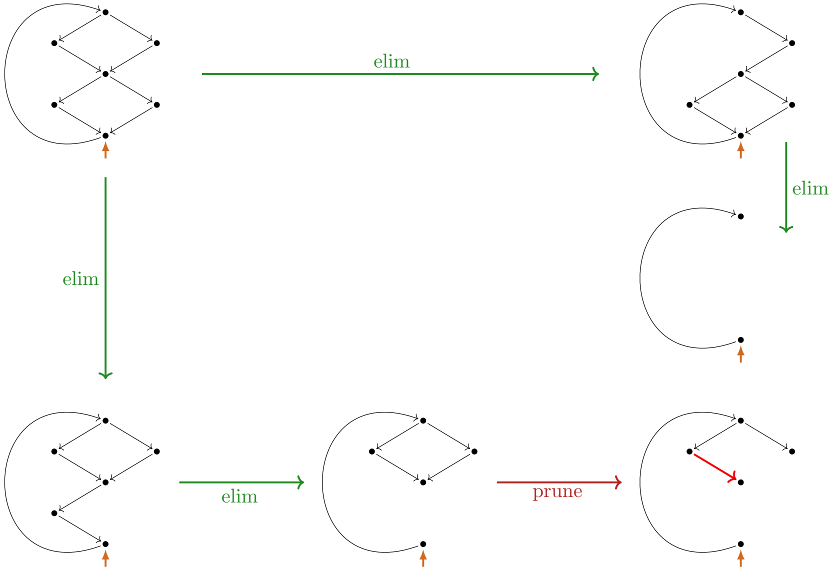
'Critical pair': bi-loop elimination



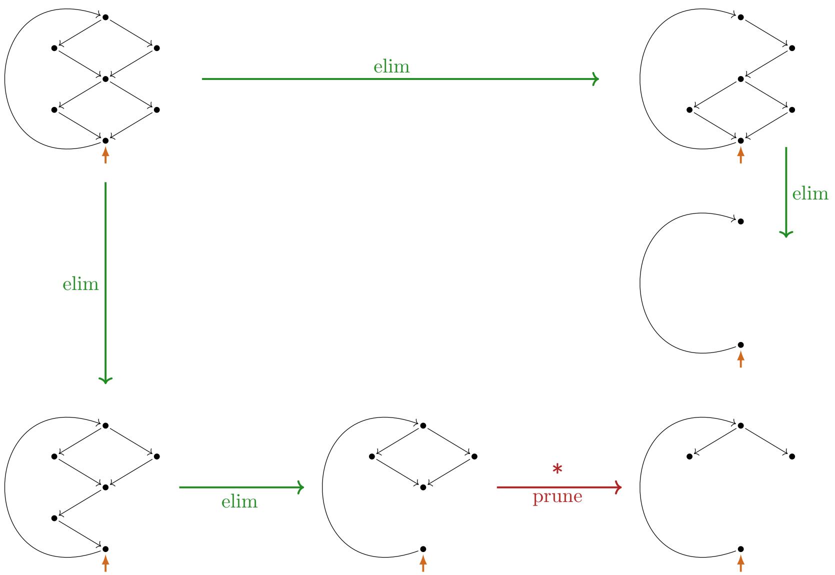
‘Critical pair’: bi-loop elimination



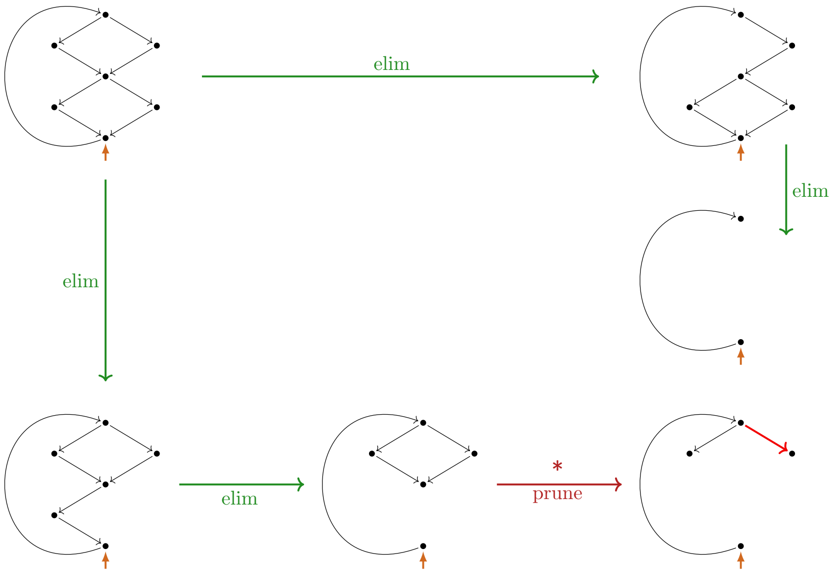
'Critical pair': bi-loop elimination



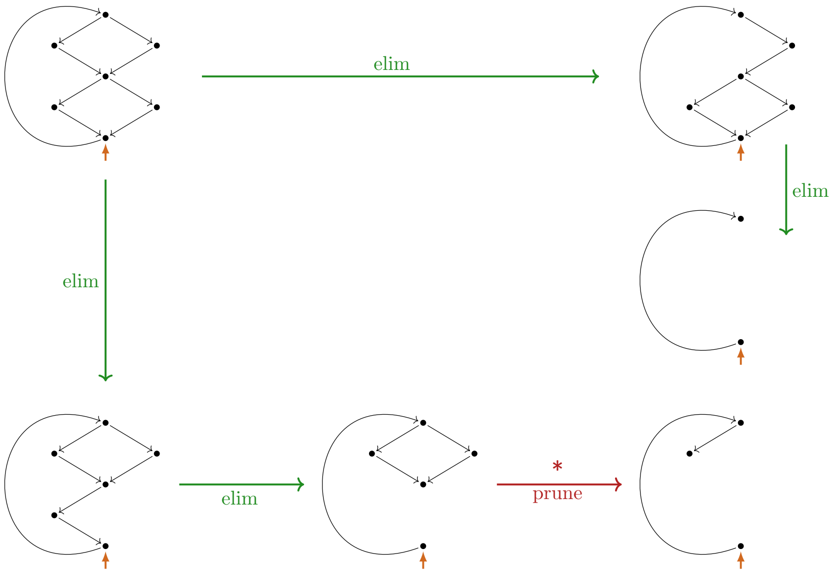
'Critical pair': bi-loop elimination



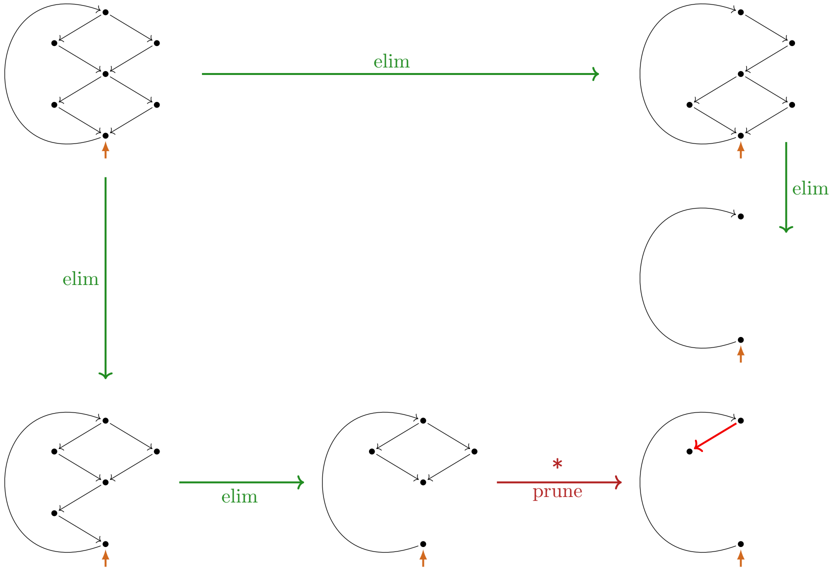
'Critical pair': bi-loop elimination



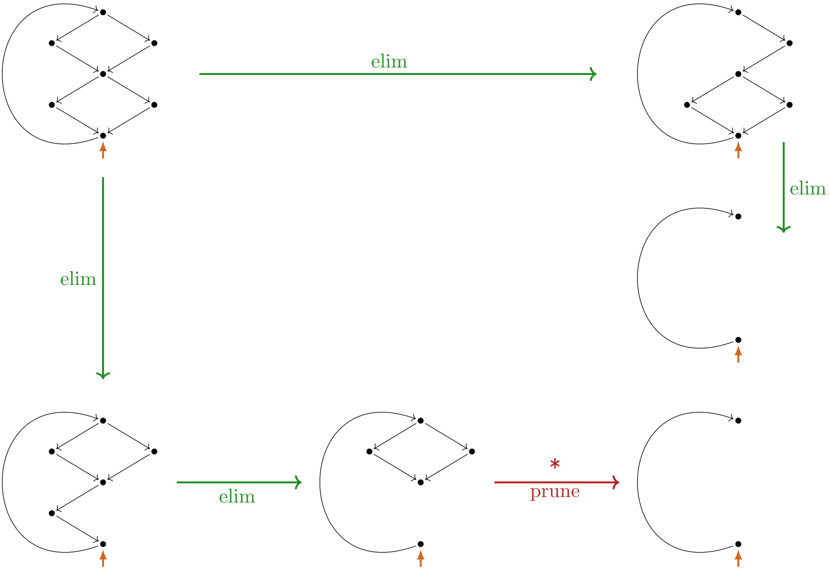
'Critical pair': bi-loop elimination



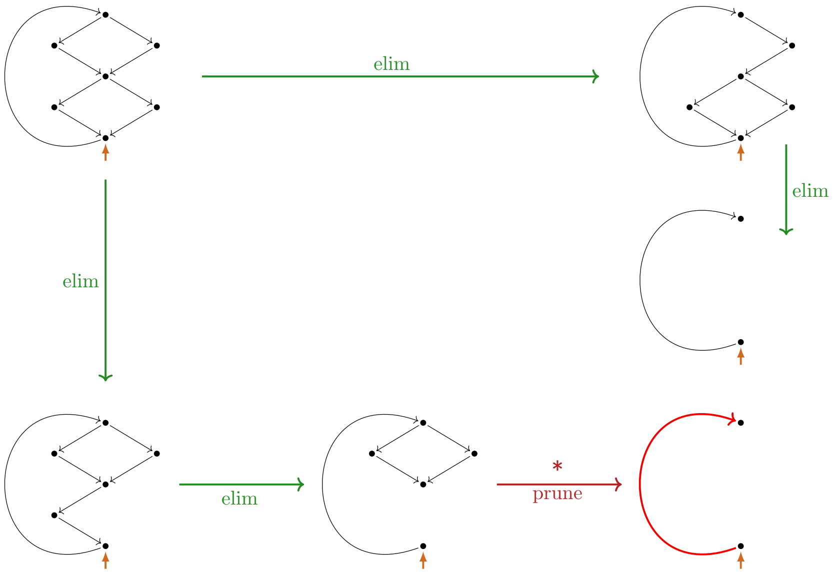
'Critical pair': bi-loop elimination



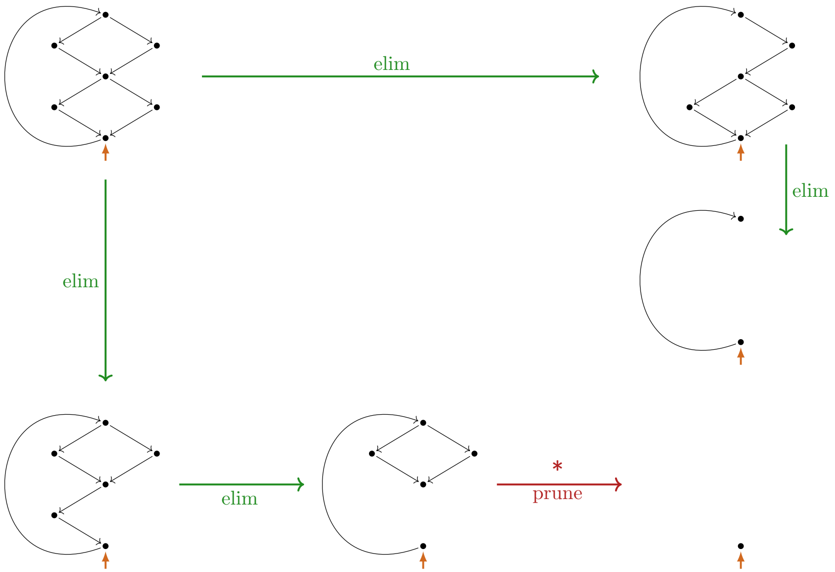
'Critical pair': bi-loop elimination



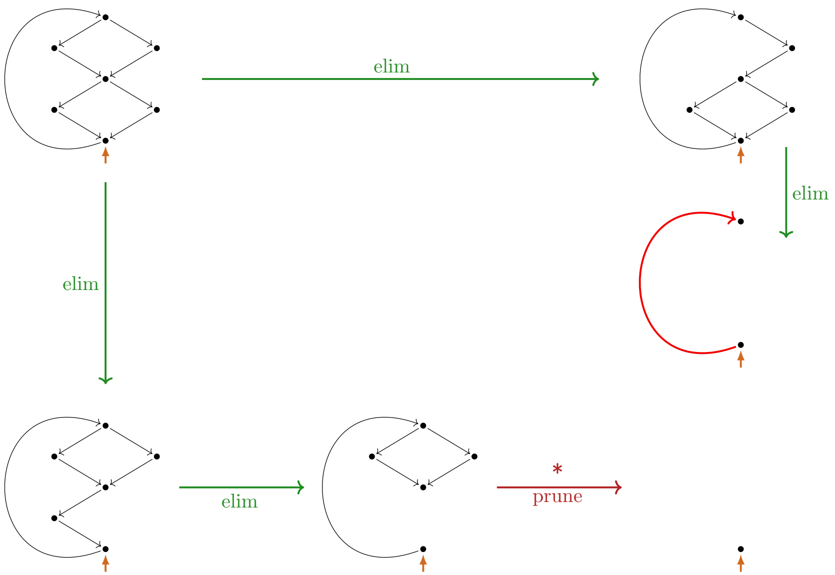
'Critical pair': bi-loop elimination



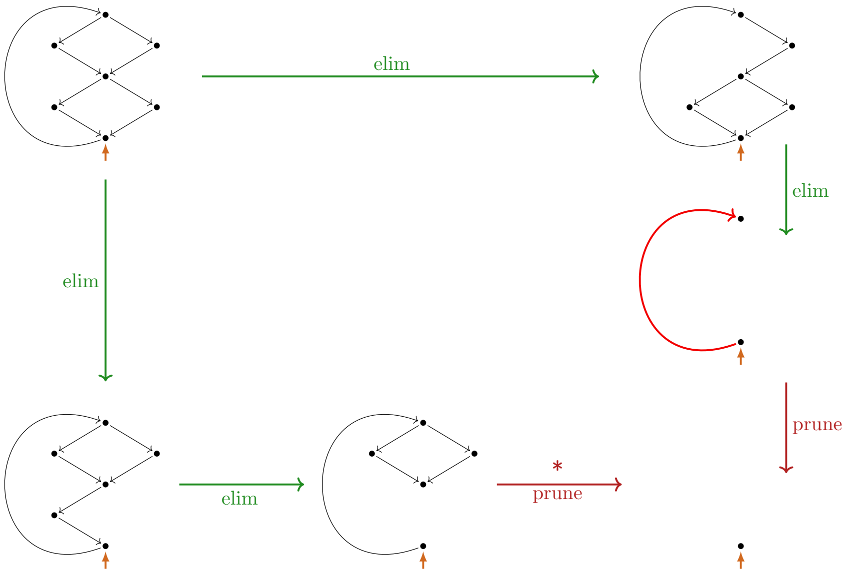
'Critical pair': bi-loop elimination



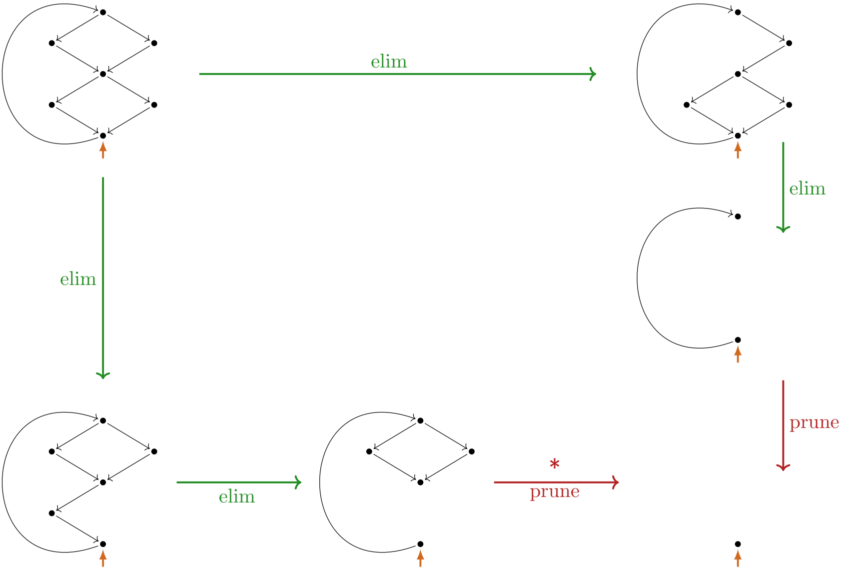
'Critical pair': bi-loop elimination



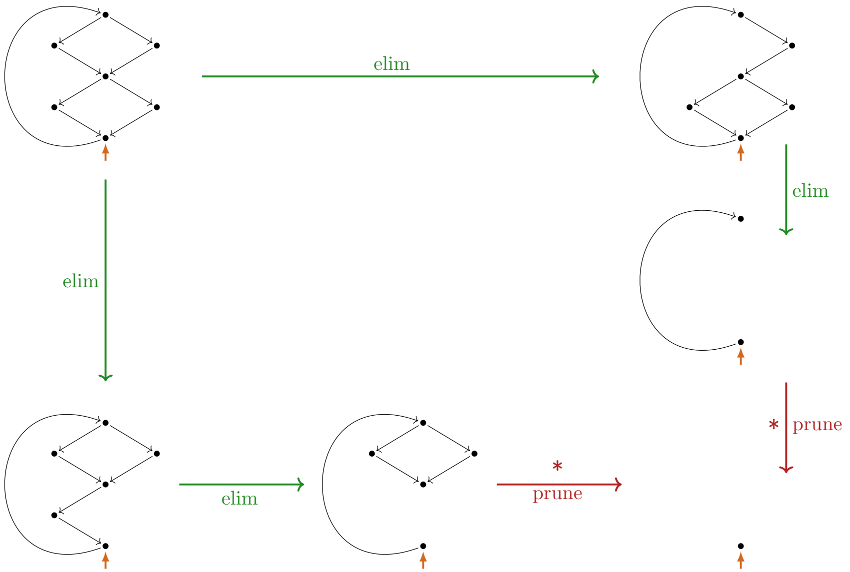
'Critical pair': bi-loop elimination



‘Critical pair’: bi-loop elimination



‘Critical pair’: bi-loop elimination



Loop elimination, and properties

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

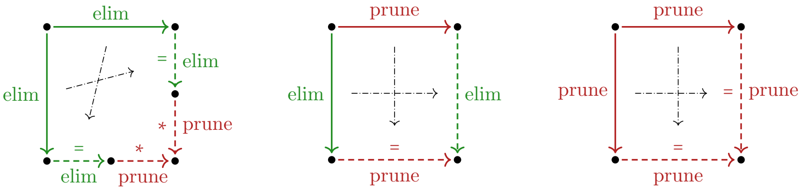
- removing the loop-entry transition(s)
- garbage collection

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

Lemma

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

(ii) $\longrightarrow_{\mathrm{elim}} \cup \longrightarrow_{\mathrm{prune}}$ *is decreasing, and hence locally confluent.*



Loop elimination, and properties

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

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

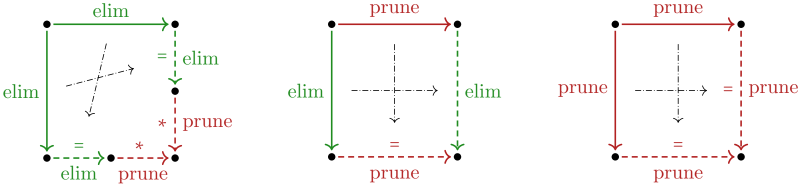
$\longrightarrow_{\text{prune}}$: remove a transition to a deadlocking state

Lemma

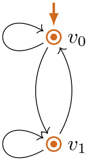
(i) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is terminating.*

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

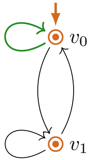
(iii) $\longrightarrow_{\text{elim}} \cup \longrightarrow_{\text{prune}}$ *is confluent.*



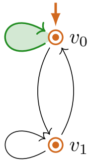
Loop elimination



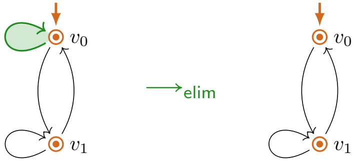
Loop elimination



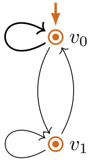
Loop elimination



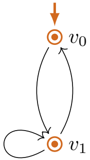
Loop elimination



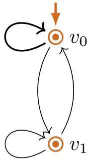
Loop elimination



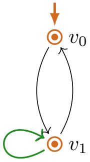
$\longrightarrow_{\text{elim}}$



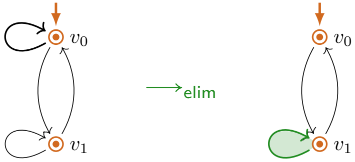
Loop elimination



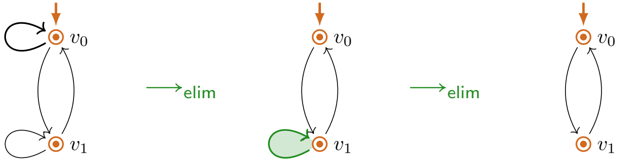
$\longrightarrow_{\text{elim}}$



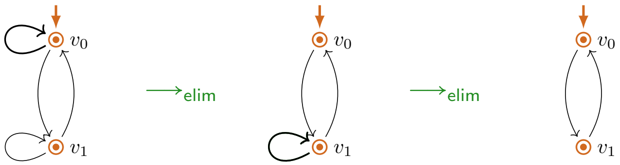
Loop elimination



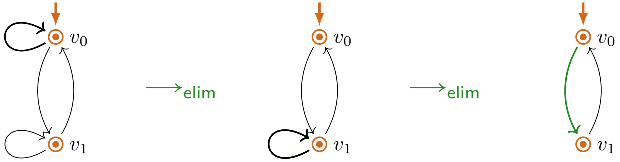
Loop elimination



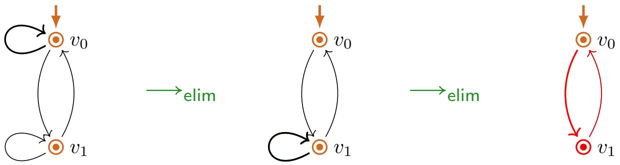
Loop elimination



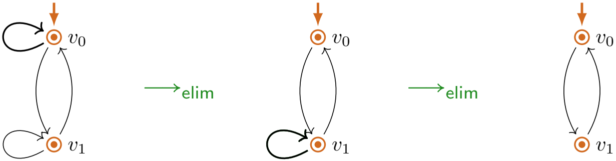
Loop elimination



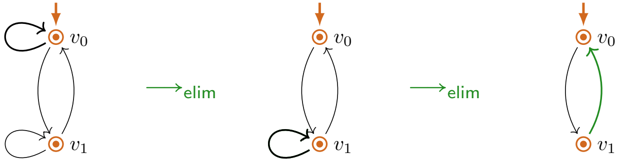
Loop elimination



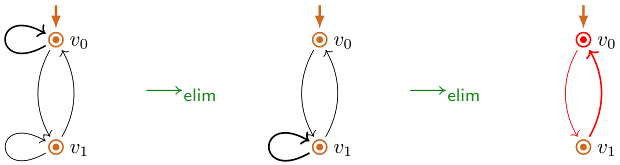
Loop elimination



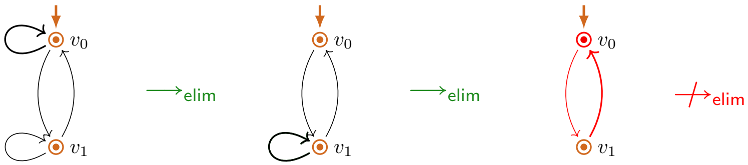
Loop elimination



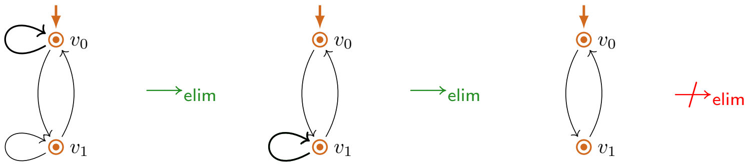
Loop elimination



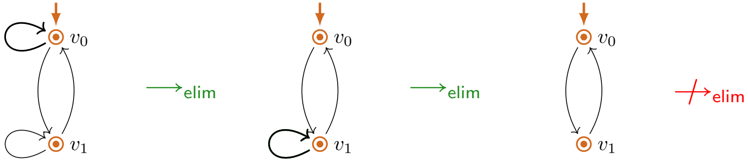
Loop elimination

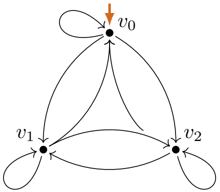


Loop elimination

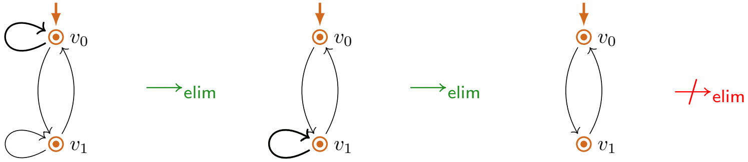


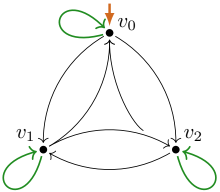
Loop elimination



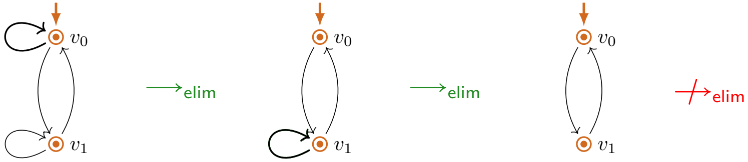


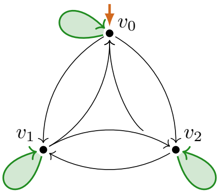
Loop elimination



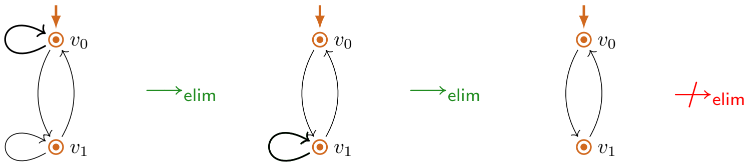


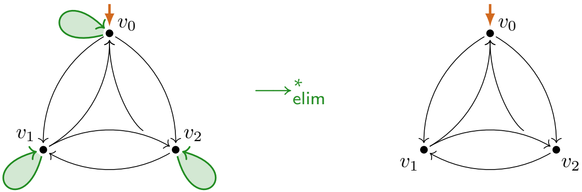
Loop elimination



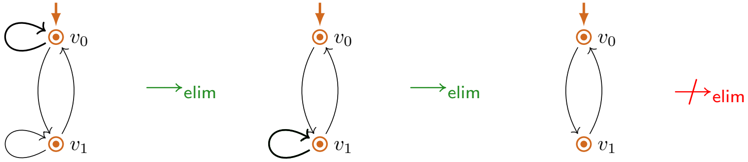


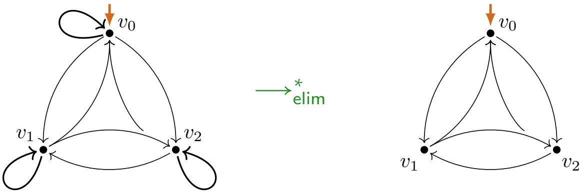
Loop elimination



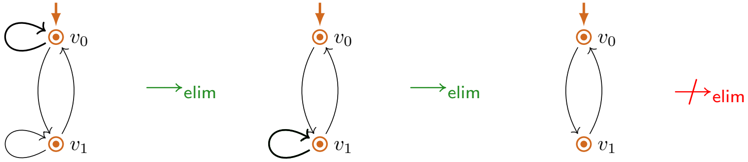


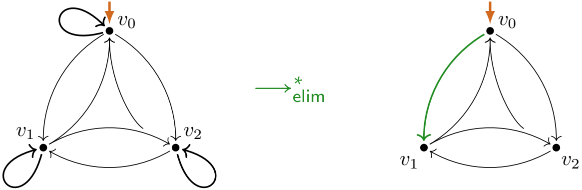
Loop elimination



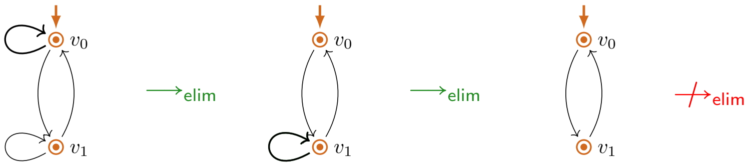


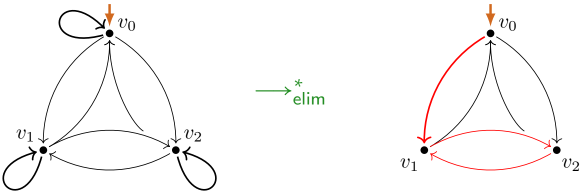
Loop elimination



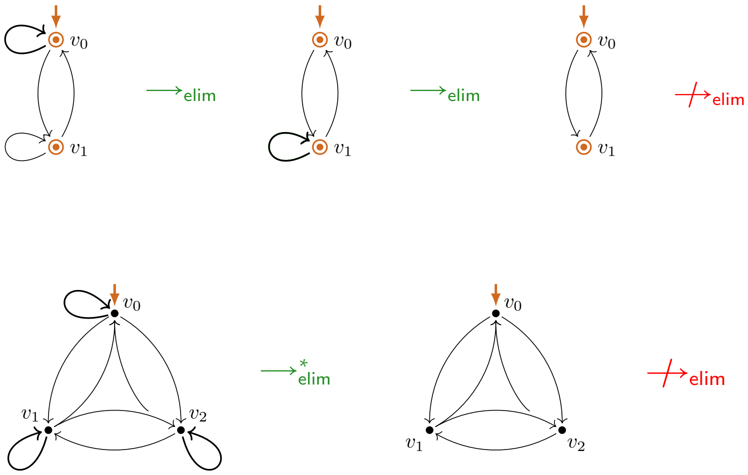


Loop elimination





Loop elimination



Loop elimination

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

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

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

Lemma

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

Loop elimination

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

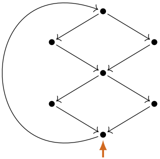
- removing the loop-entry transition(s)
- garbage collection

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

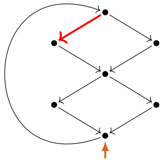
Lemma

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

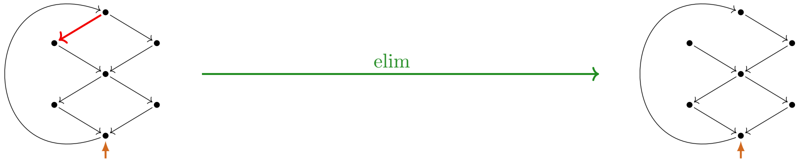
'Critical pair': bi-loop elimination



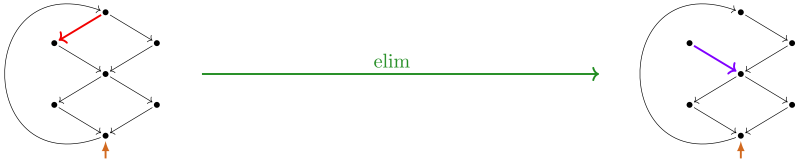
'Critical pair': bi-loop elimination



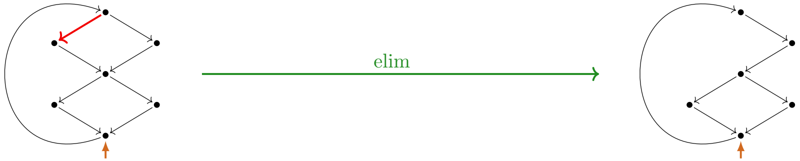
'Critical pair': bi-loop elimination



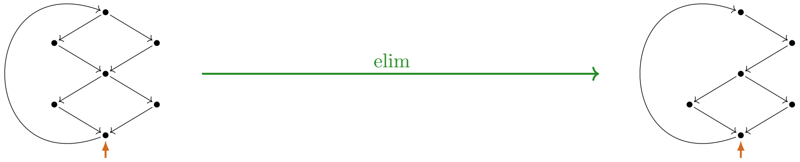
'Critical pair': bi-loop elimination



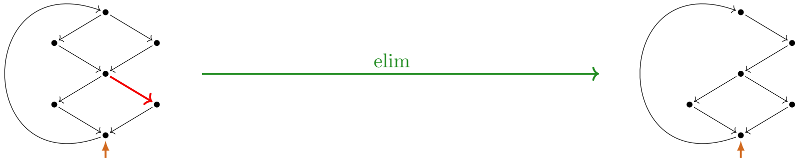
'Critical pair': bi-loop elimination



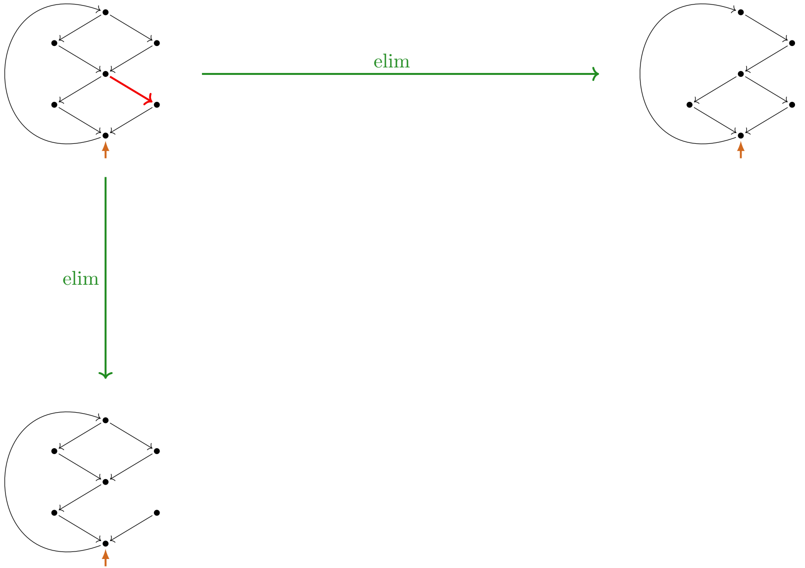
'Critical pair': bi-loop elimination



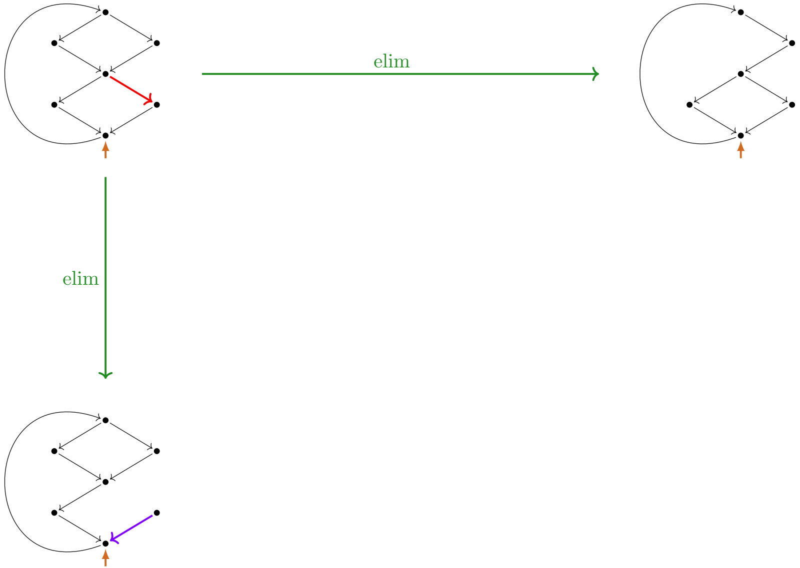
'Critical pair': bi-loop elimination



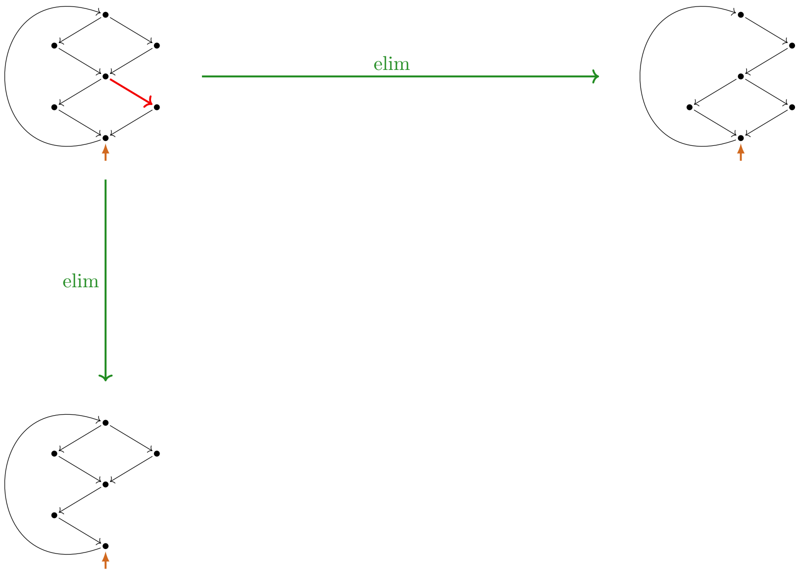
‘Critical pair’: bi-loop elimination



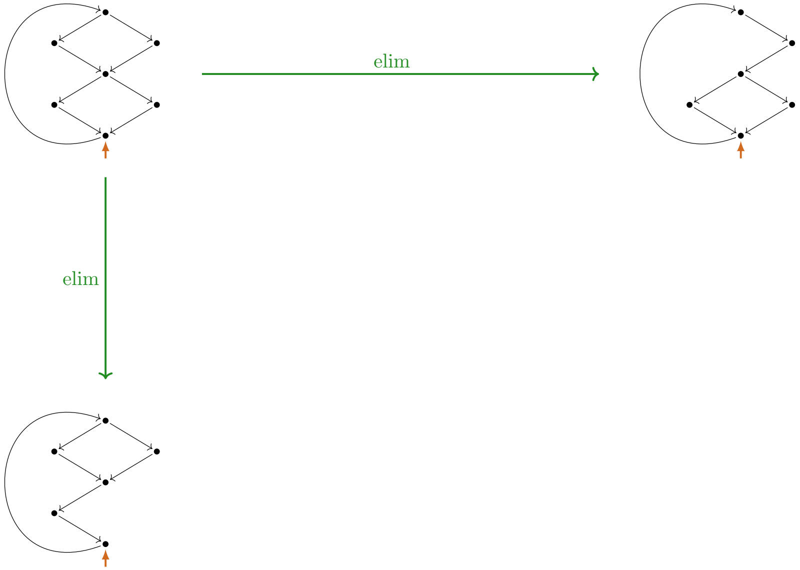
‘Critical pair’: bi-loop elimination



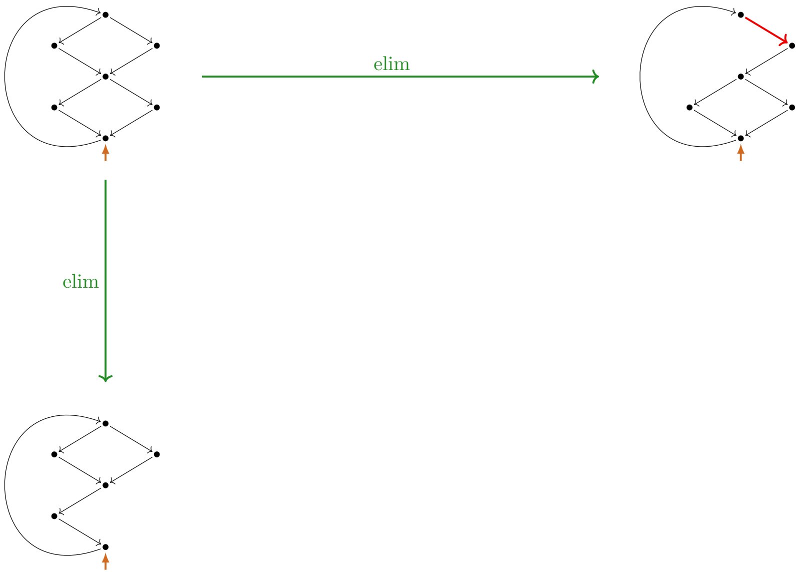
'Critical pair': bi-loop elimination



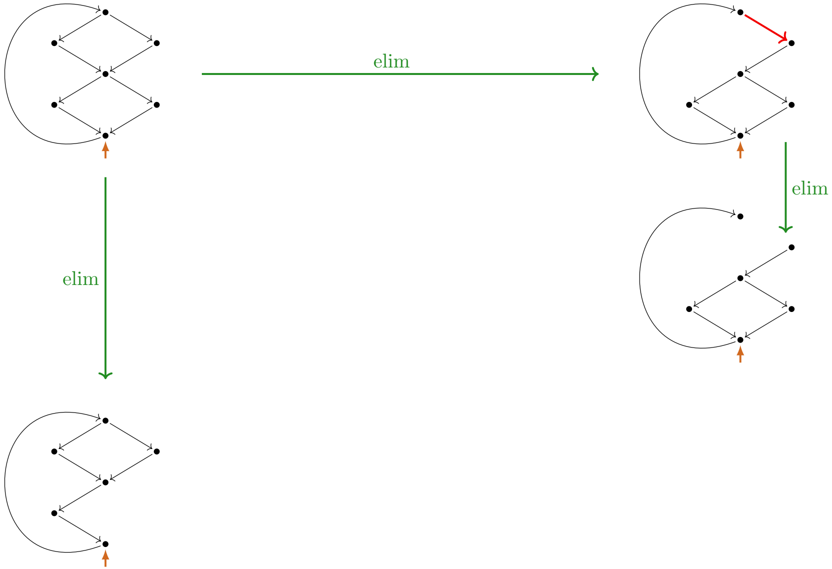
'Critical pair': bi-loop elimination



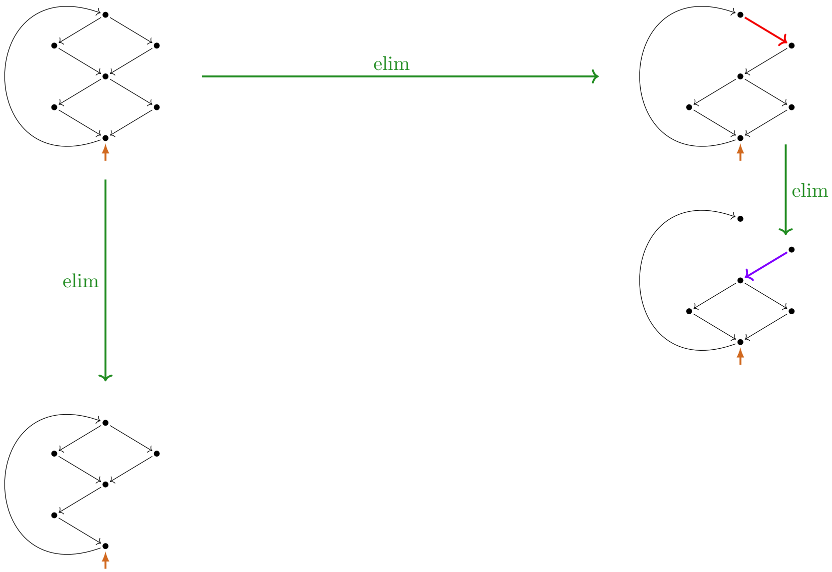
'Critical pair': bi-loop elimination



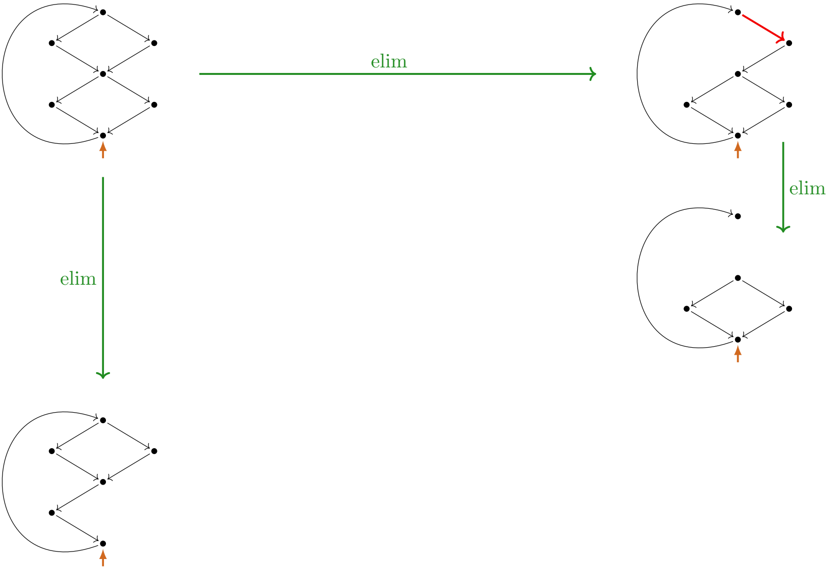
'Critical pair': bi-loop elimination



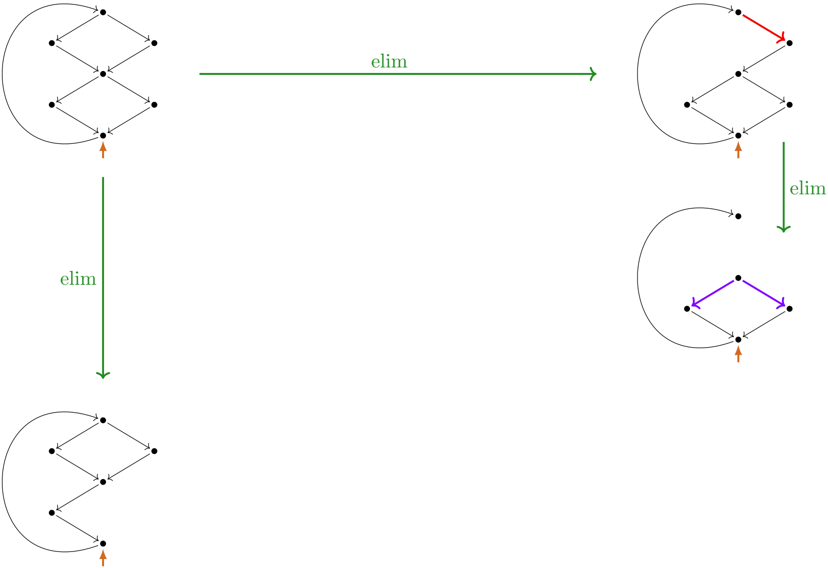
'Critical pair': bi-loop elimination



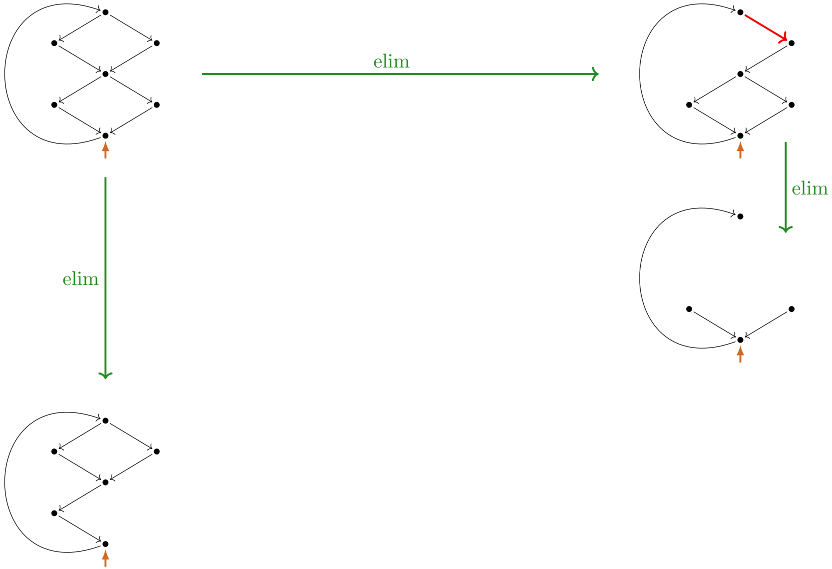
‘Critical pair’: bi-loop elimination



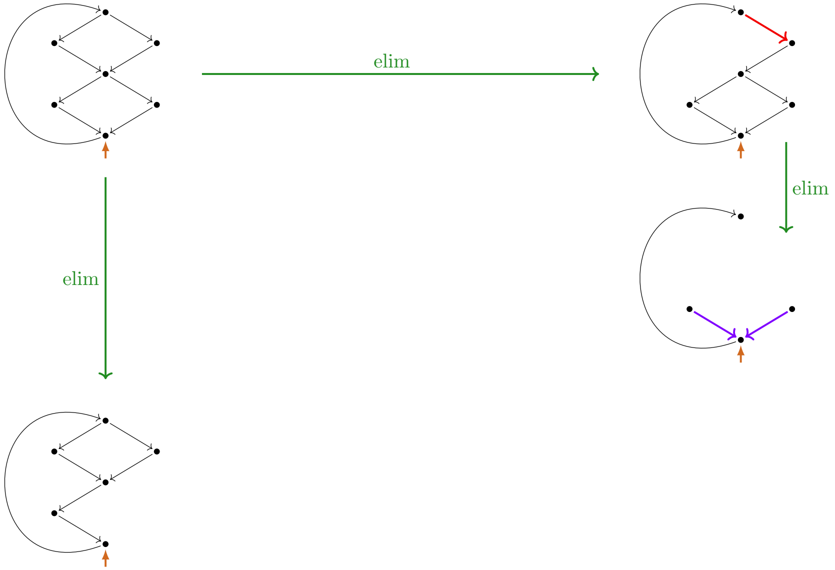
'Critical pair': bi-loop elimination



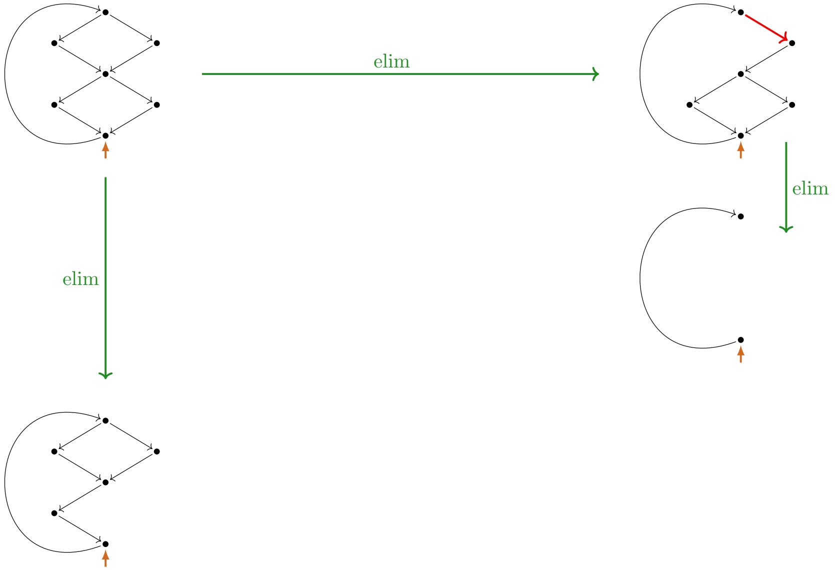
‘Critical pair’: bi-loop elimination



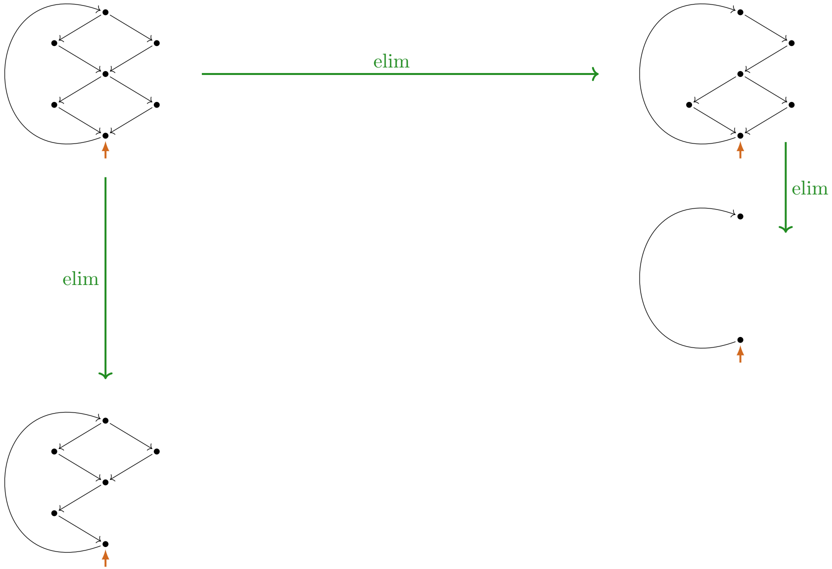
'Critical pair': bi-loop elimination



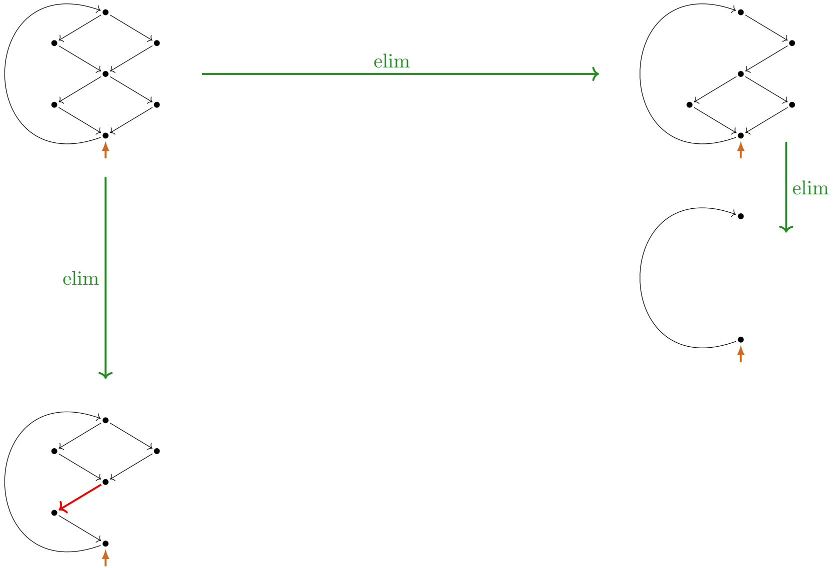
‘Critical pair’: bi-loop elimination



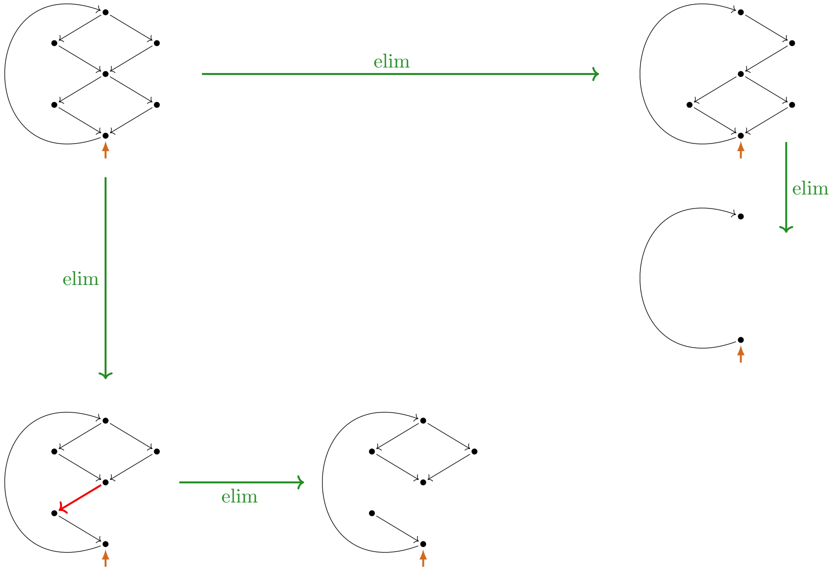
‘Critical pair’: bi-loop elimination



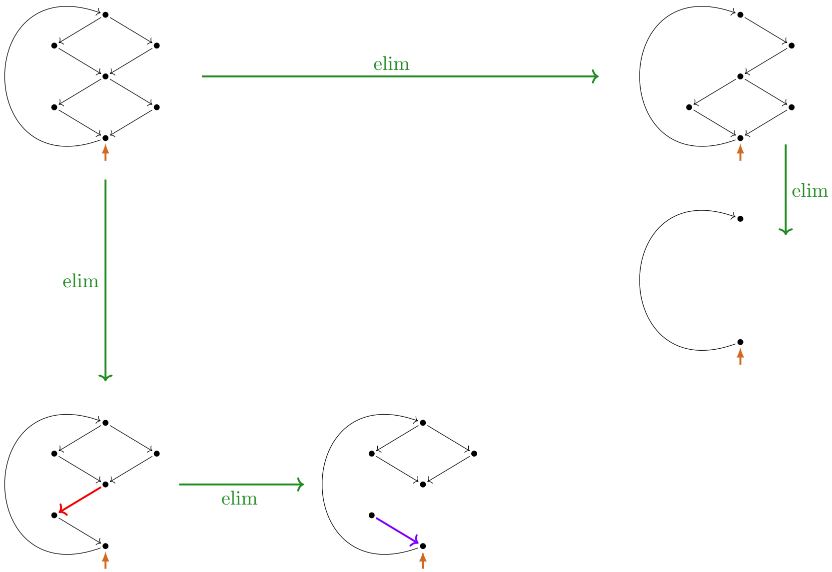
‘Critical pair’: bi-loop elimination



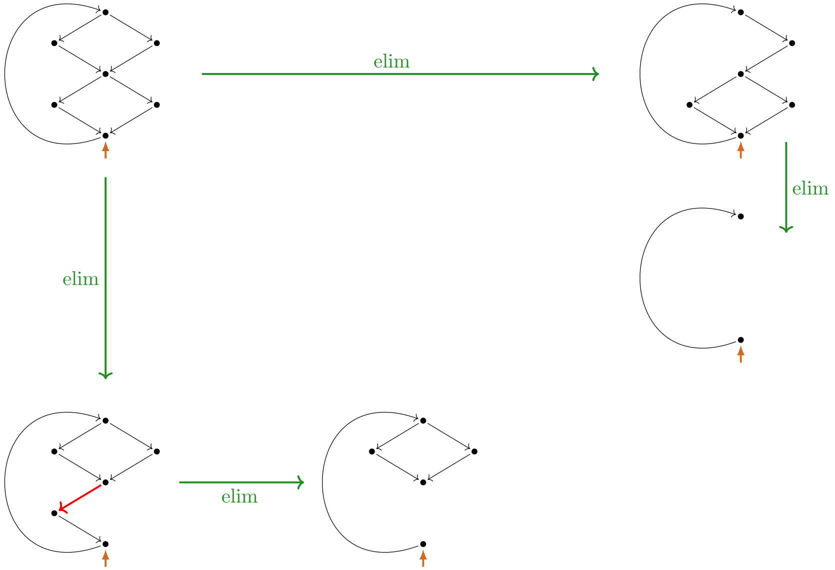
'Critical pair': bi-loop elimination



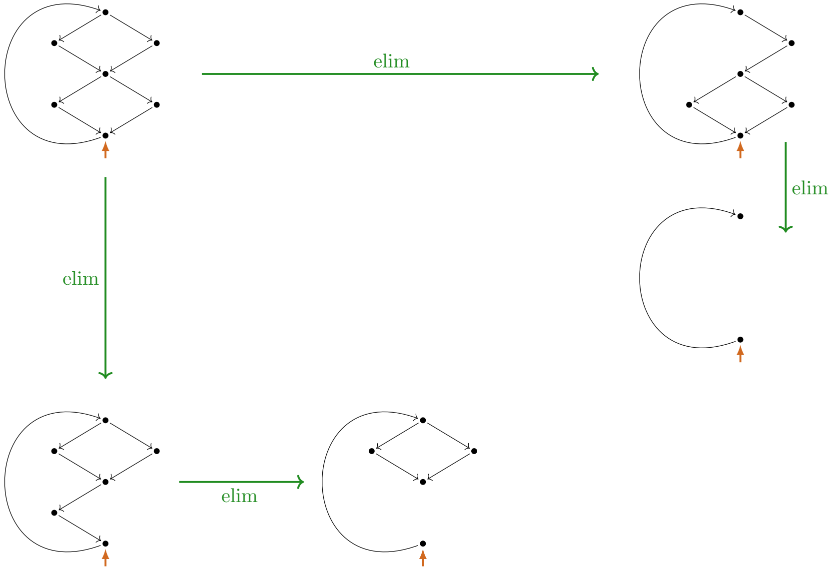
‘Critical pair’: bi-loop elimination



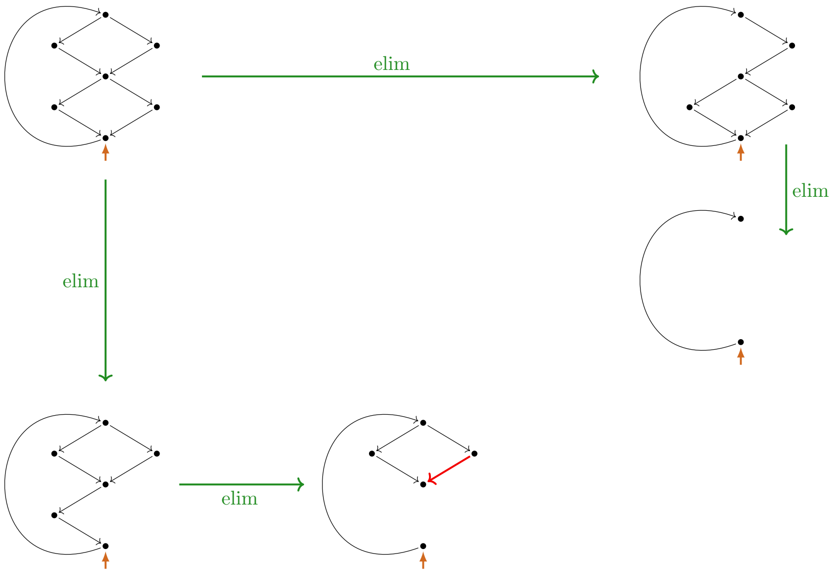
'Critical pair': bi-loop elimination



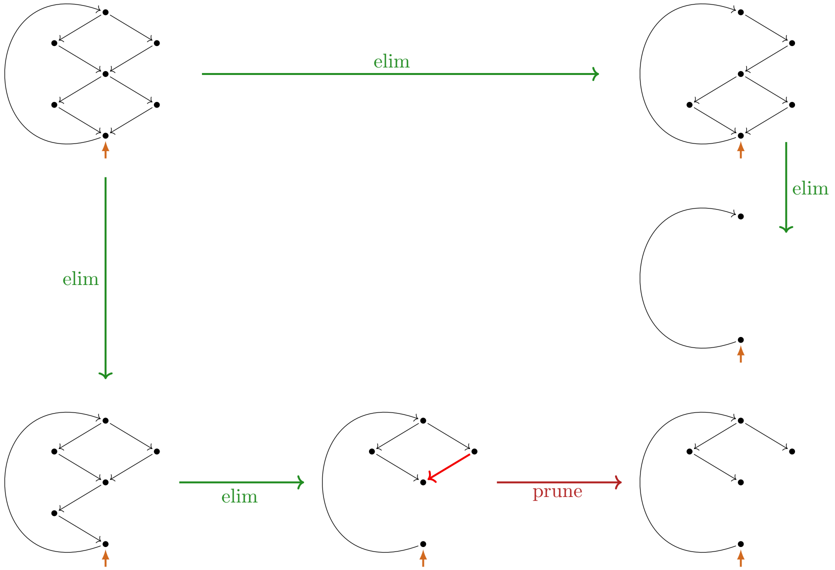
'Critical pair': bi-loop elimination



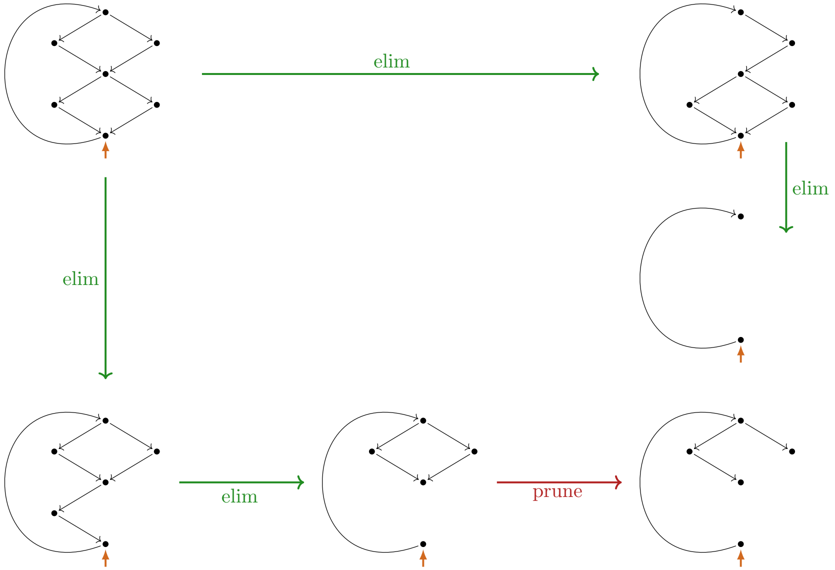
'Critical pair': bi-loop elimination



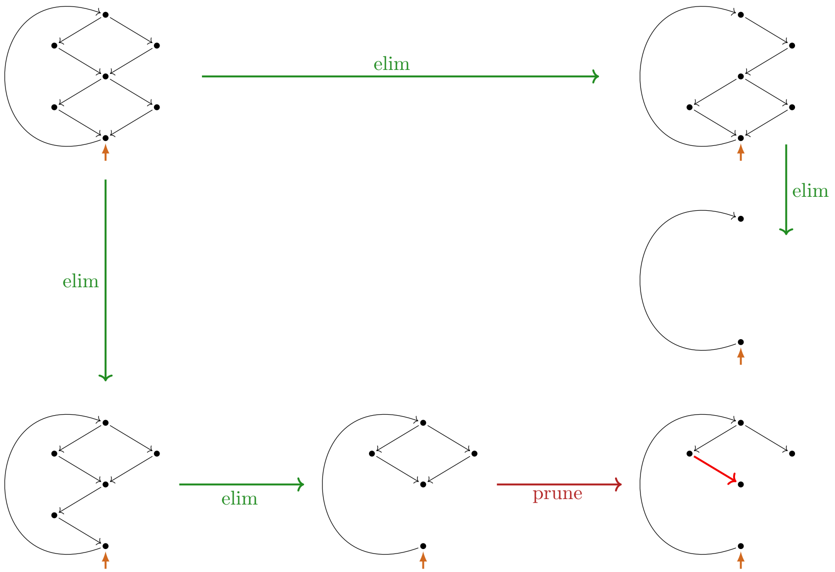
'Critical pair': bi-loop elimination



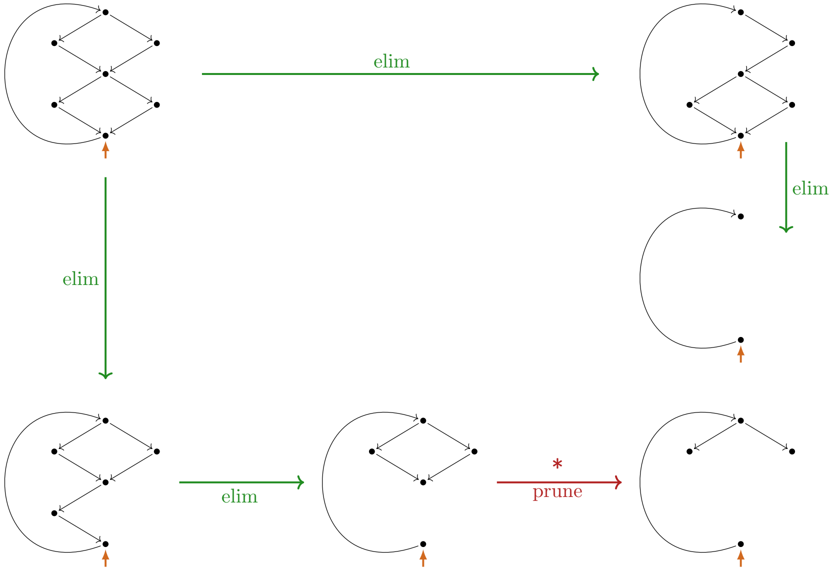
'Critical pair': bi-loop elimination



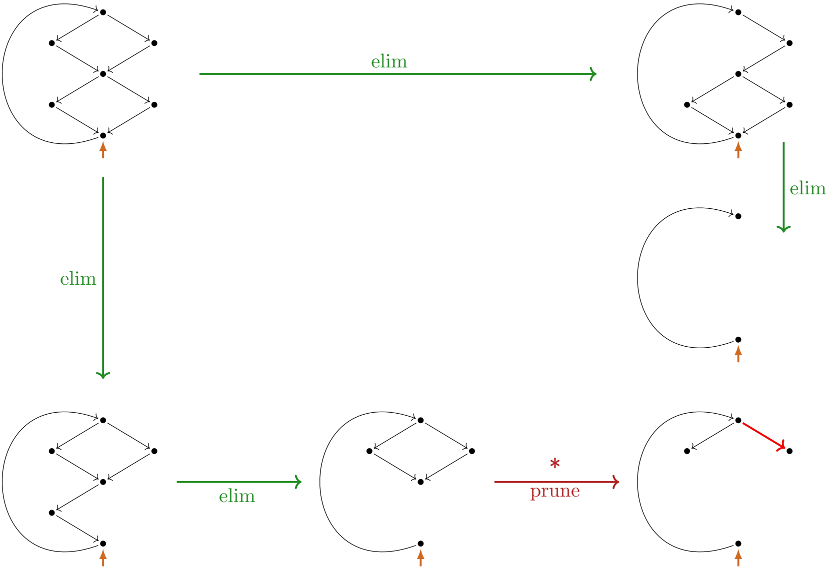
'Critical pair': bi-loop elimination



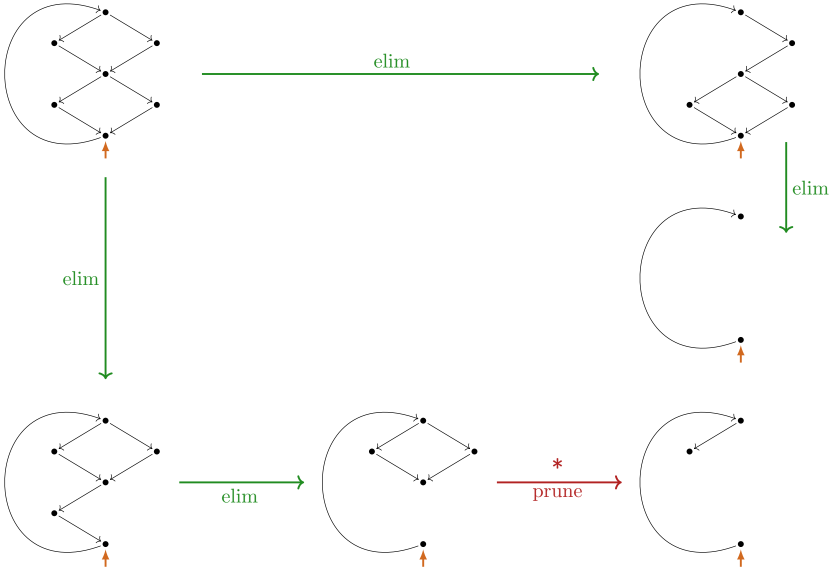
‘Critical pair’: bi-loop elimination



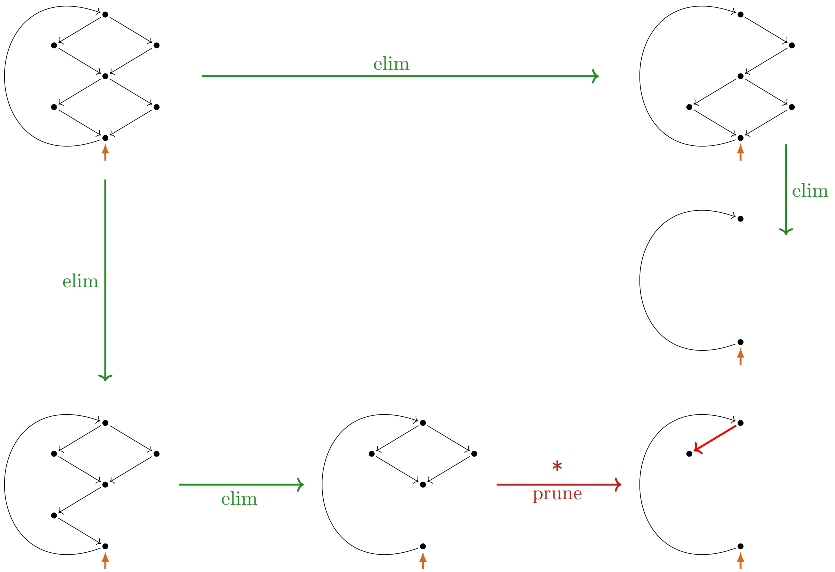
'Critical pair': bi-loop elimination



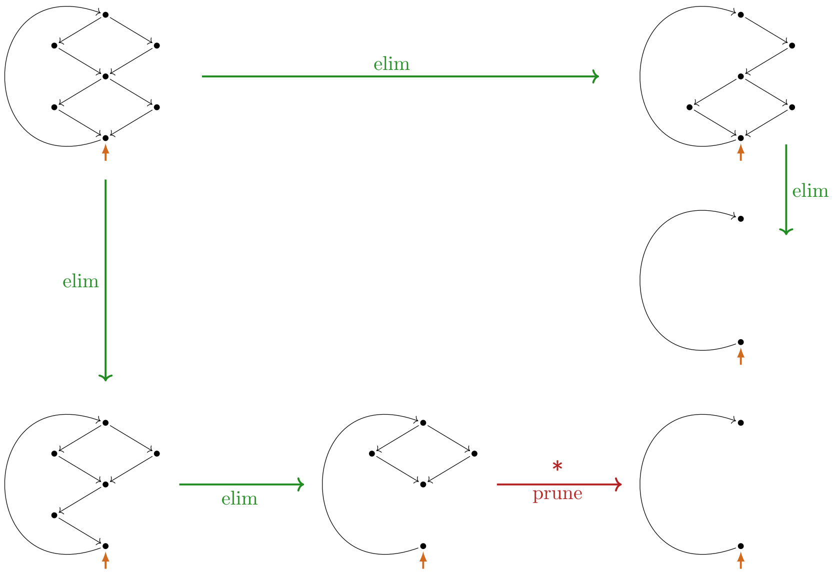
‘Critical pair’: bi-loop elimination



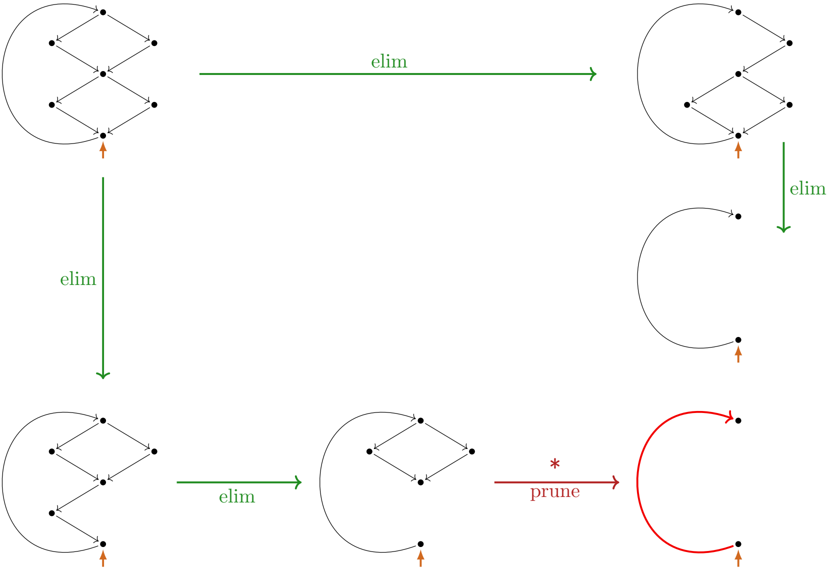
'Critical pair': bi-loop elimination



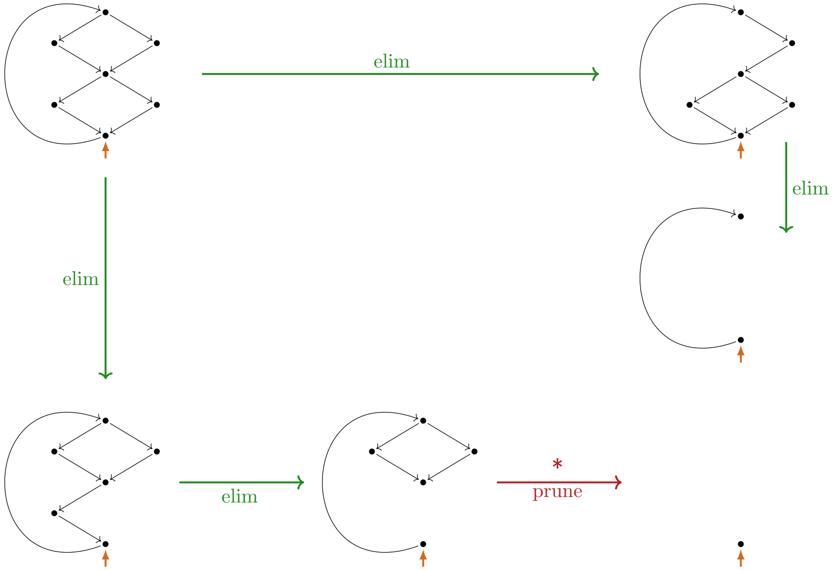
'Critical pair': bi-loop elimination



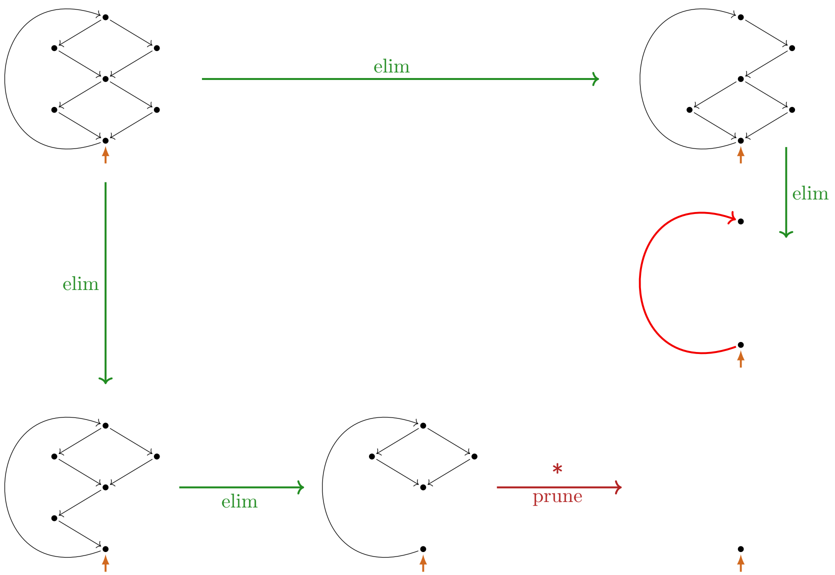
'Critical pair': bi-loop elimination



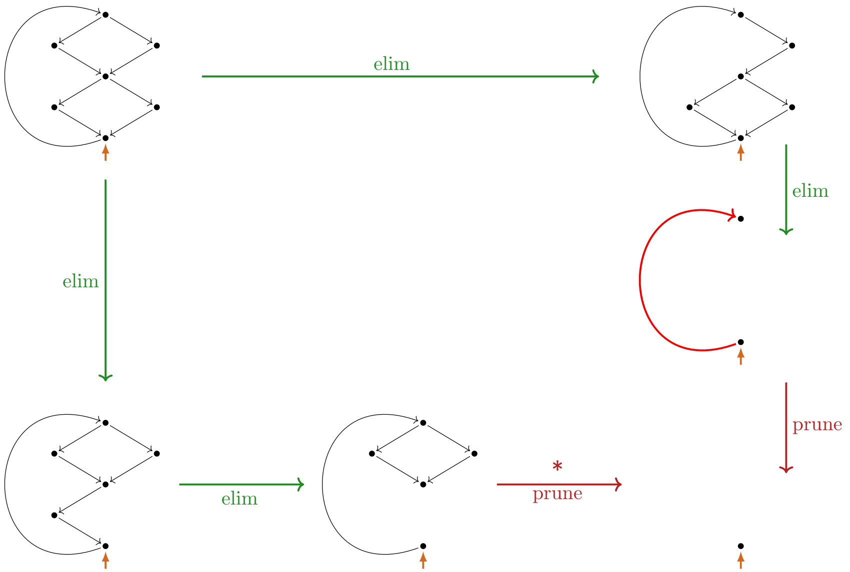
‘Critical pair’: bi-loop elimination



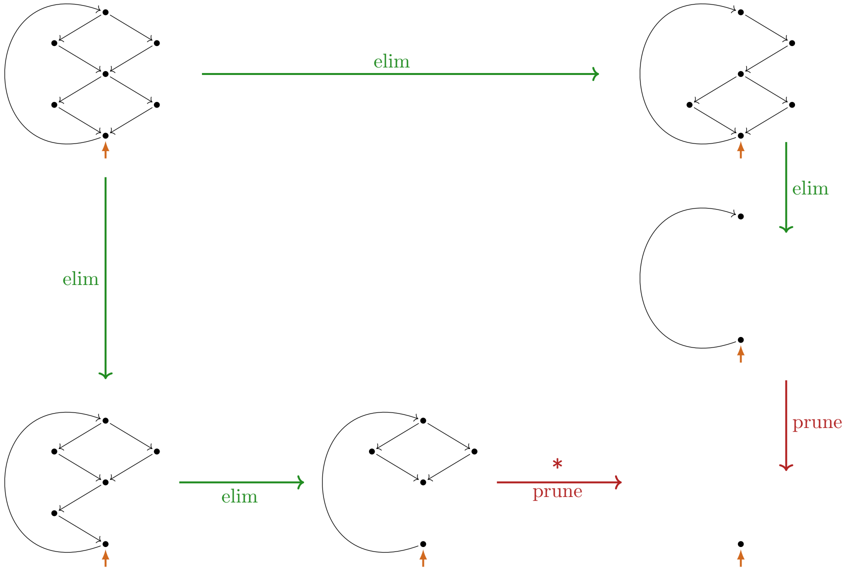
‘Critical pair’: bi-loop elimination



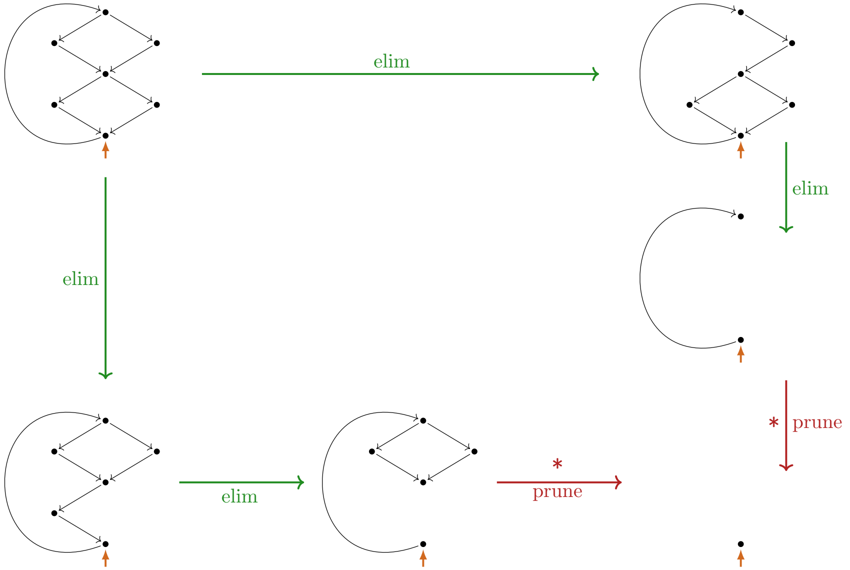
'Critical pair': bi-loop elimination



'Critical pair': bi-loop elimination



‘Critical pair’: bi-loop elimination



Loop elimination, and properties

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

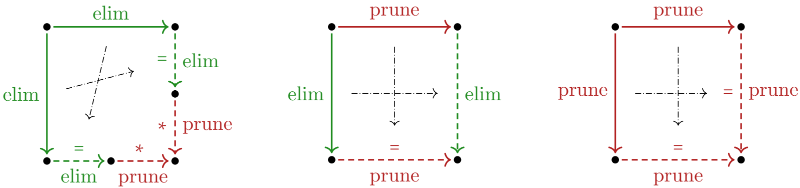
- removing the loop-entry transition(s)
- garbage collection

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

Lemma

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

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



Loop elimination, and properties

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

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

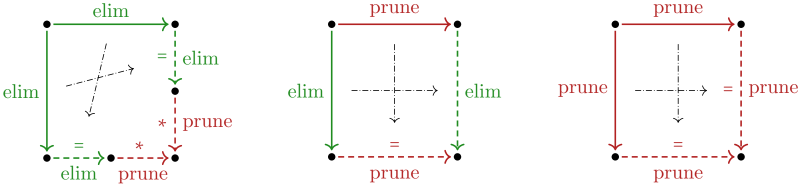
$\longrightarrow_{\text{prune}}$: remove a transition to a deadlocking state

Lemma

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

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

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



Structure property LEE

Definition

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

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

Structure property LEE

Definition

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

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

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

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

Structure property LEE

Definition

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

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

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

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

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

Structure property LEE

Definition

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

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

Lemma (by using termination and confluence)

For every process graph G the following are equivalent:

(i) LEE(G).

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

(iii) There is an $\longrightarrow_{\text{elim,prune}}$ 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:

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

Lemma (by using termination and confluence)

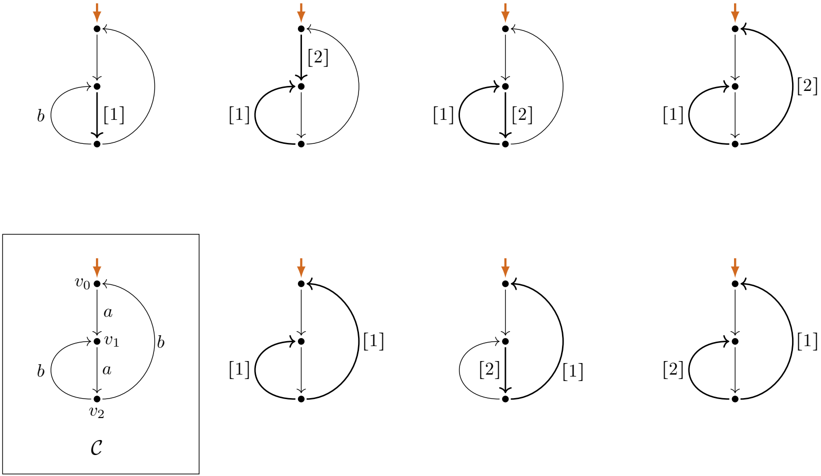
For every process graph G the following are equivalent:

- *(i)* LEE(G).
- *(ii) There is an $\longrightarrow_{\mathsf{elim}}$ normal form without an infinite trace.*
- *(iii) There is an $\longrightarrow_{\mathsf{elim,prune}}$ normal form without an infinite trace.*
- *(iv) Every $\longrightarrow_{\mathsf{elim}}$ normal form is without an infinite trace.*
- *(v) Every $\longrightarrow_{\mathsf{elim,prune}}$ normal form is without an infinite trace.*

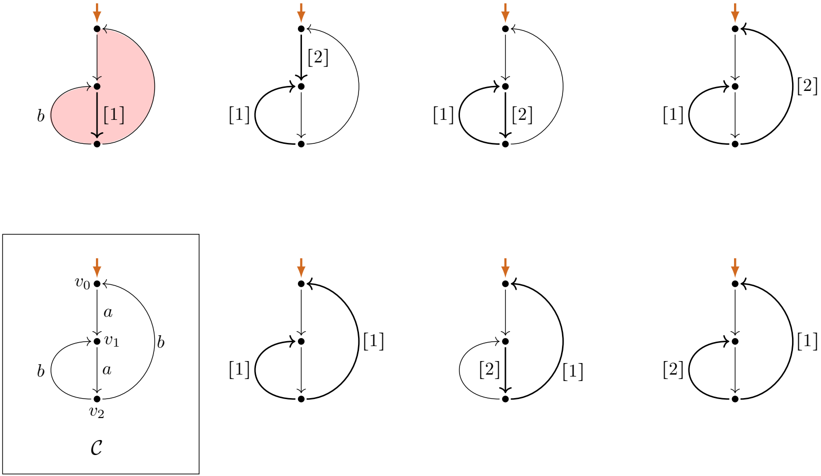
Theorem (efficient decidability)

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

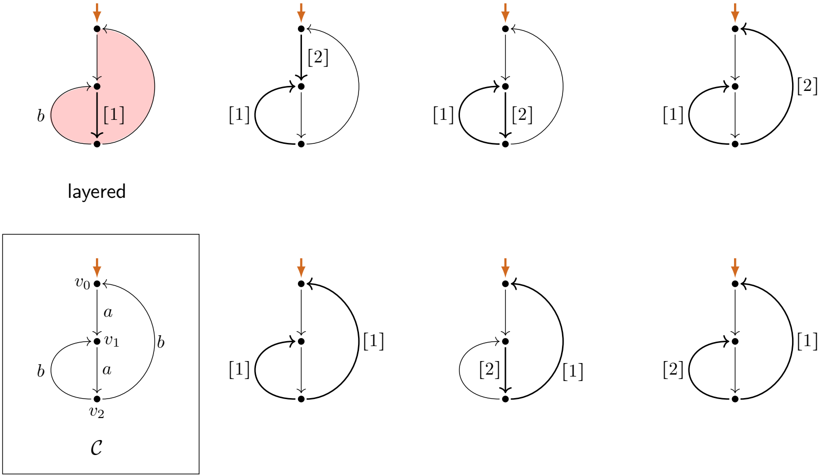
7 LEE-witnesses



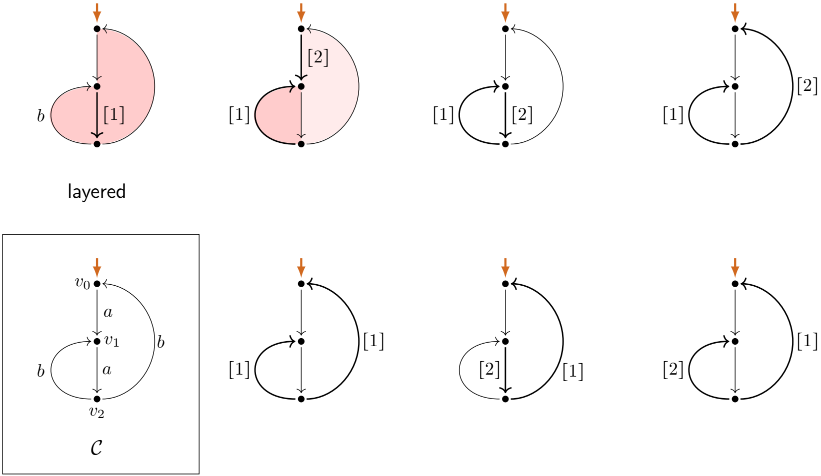
7 LEE-witnesses



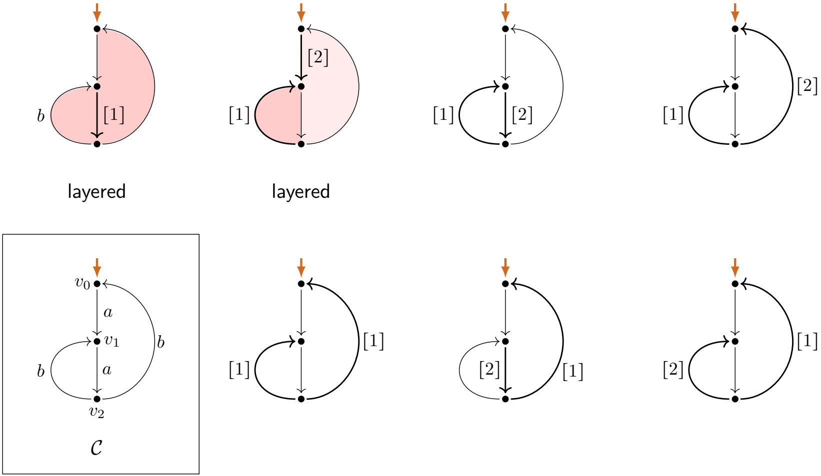
7 LEE-witnesses



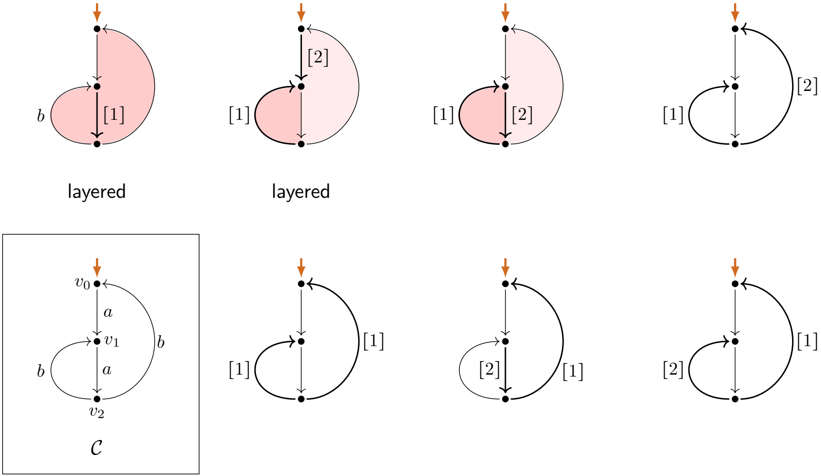
7 LEE-witnesses



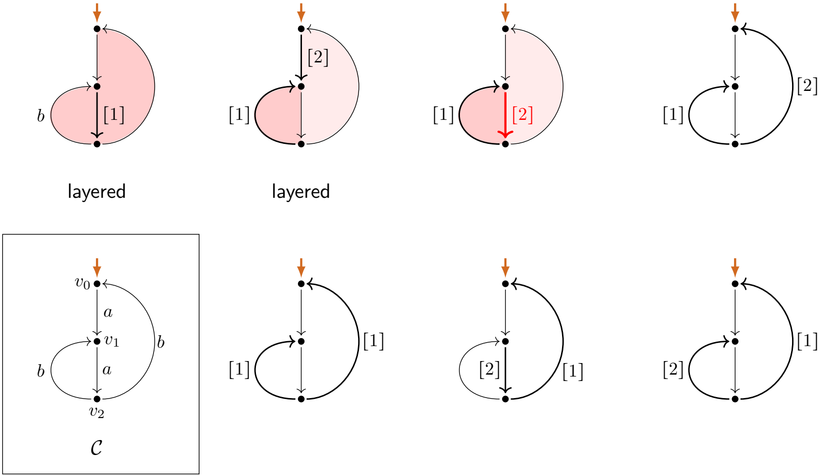
7 LEE-witnesses



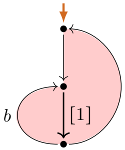
7 LEE-witnesses



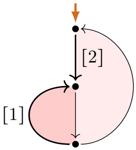
7 LEE-witnesses



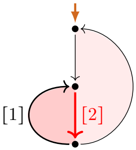
7 LEE-witnesses



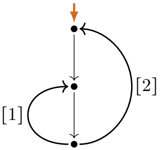
layered

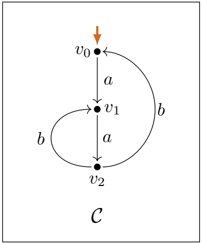


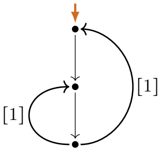
layered

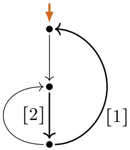


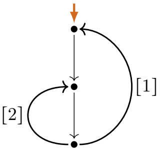
not layered



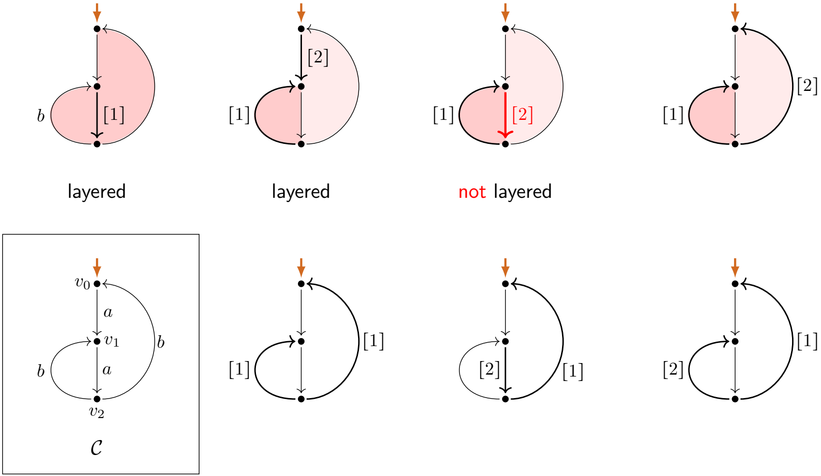




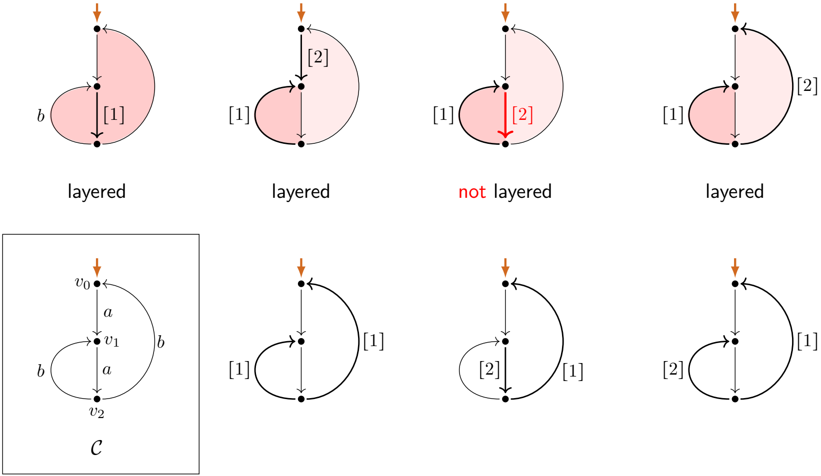




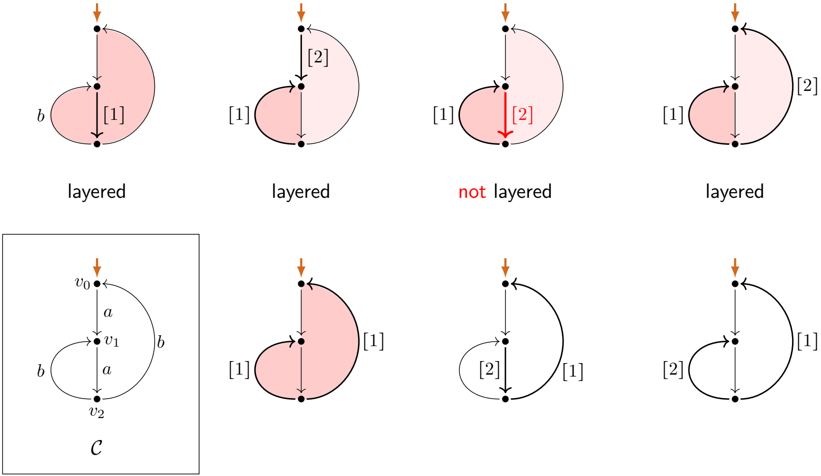
7 LEE-witnesses



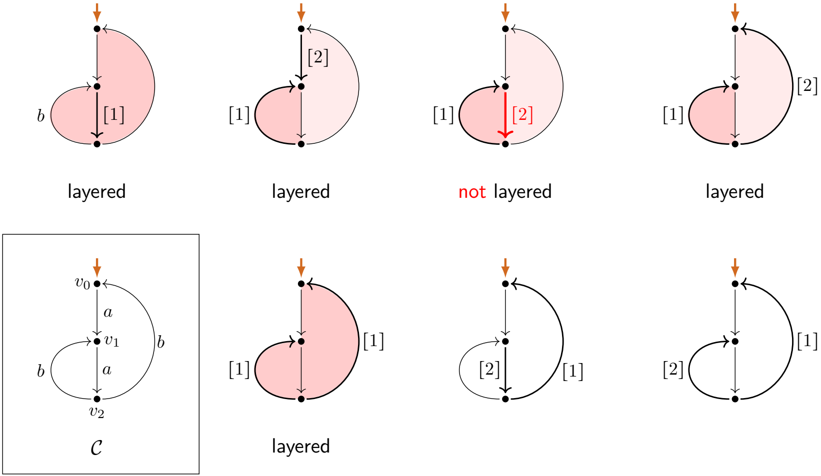
7 LEE-witnesses



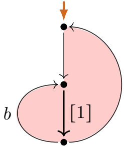
7 LEE-witnesses



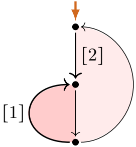
7 LEE-witnesses



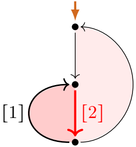
7 LEE-witnesses



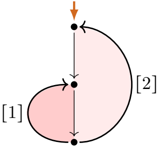
layered



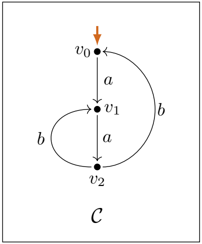
layered

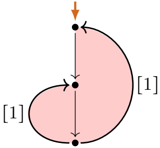


not layered

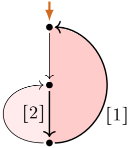


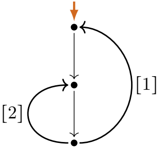
layered



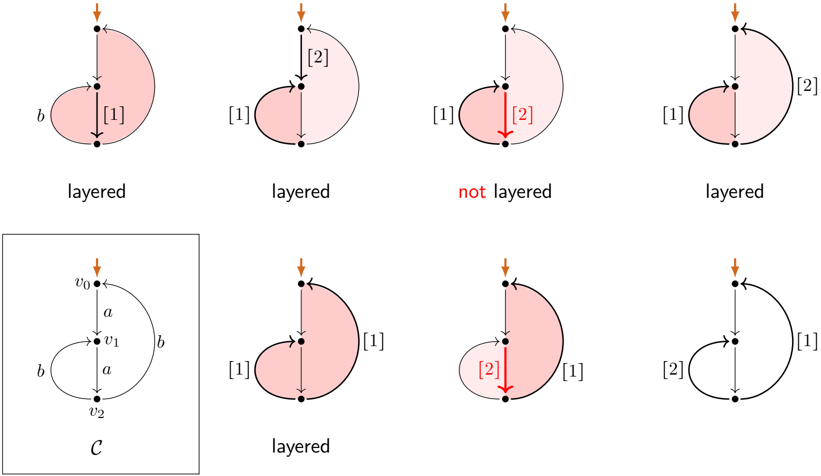


layered

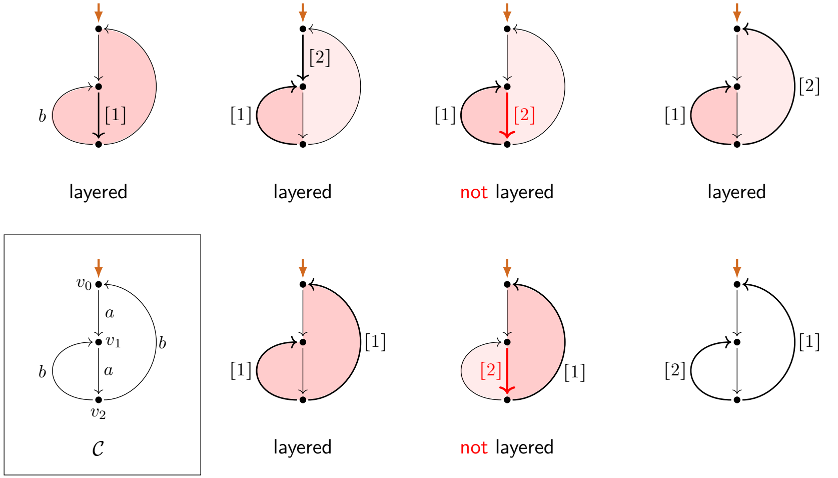




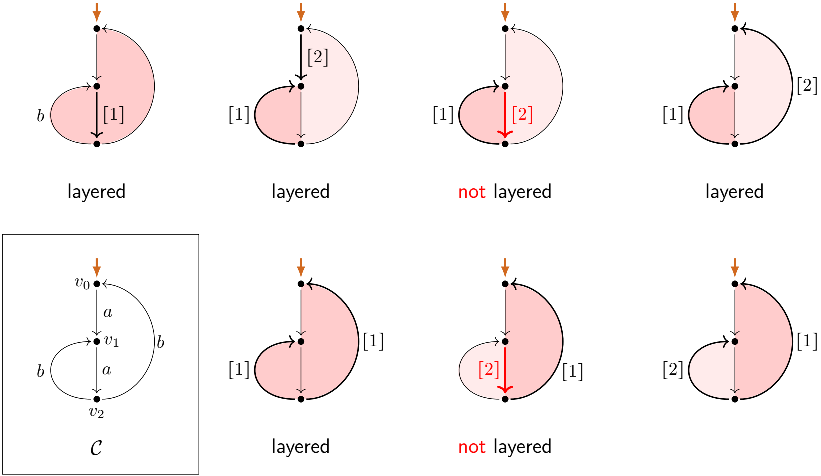
7 LEE-witnesses



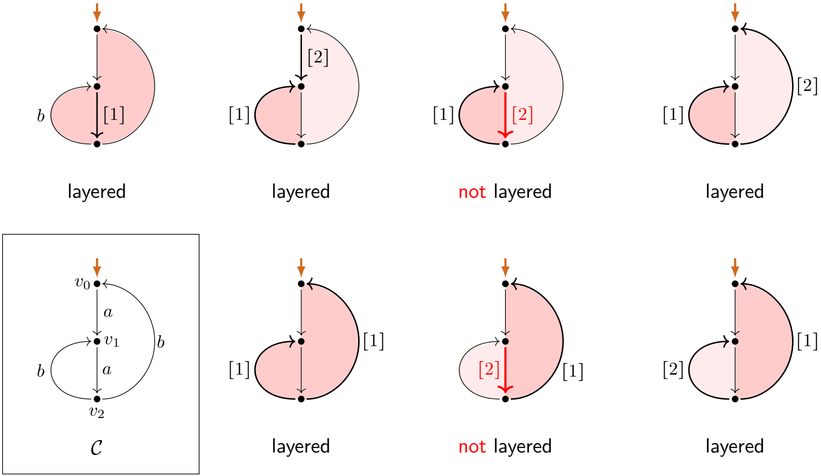
7 LEE-witnesses



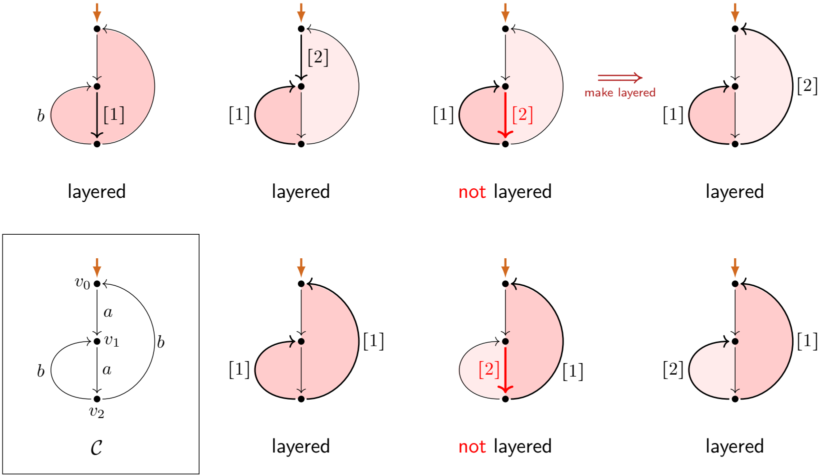
7 LEE-witnesses



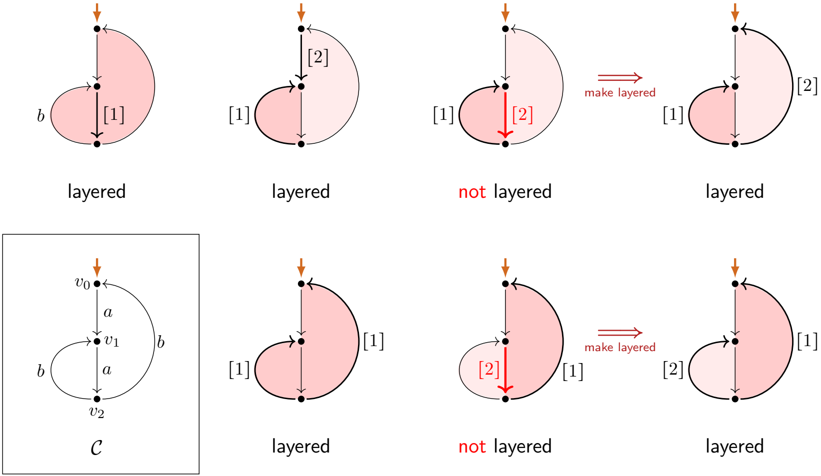
7 LEE-witnesses



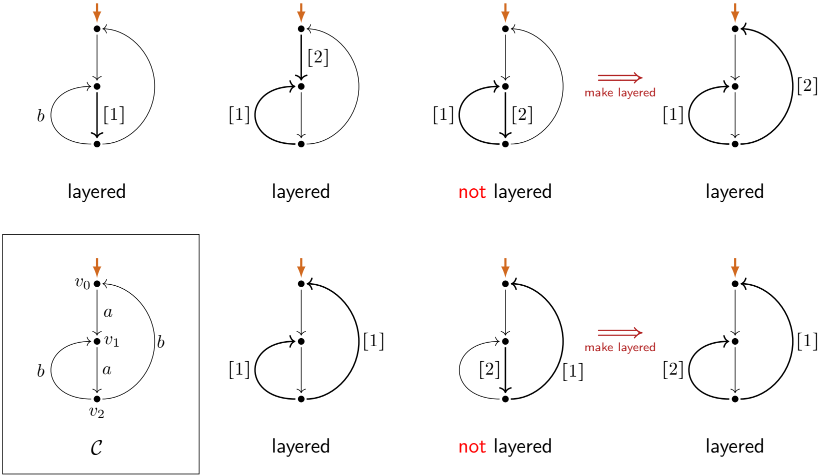
7 LEE-witnesses



7 LEE-witnesses



7 LEE-witnesses



Interpretation/extraction correspondences with LEE

(⇐ G/Fokkink 2020, G 2021)

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

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

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

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

Interpretation/extraction correspondences with LEE

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

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

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

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

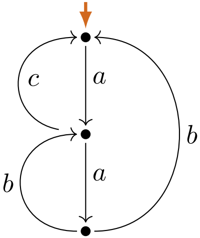
From every finite process graph G with LEE
a regular expression e can be **e**xtracted
such that $G \leftrightarrow P(e)$.

(Coll): LEE *is preserved under collapse*

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

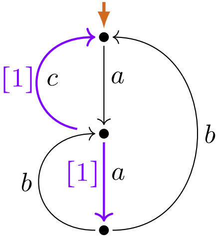
Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

G_4



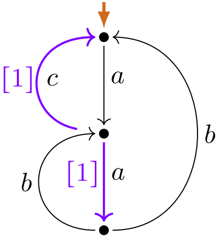
Expression extraction using LLEE (G/Fokkink 2020, G 2021/22)

$\widehat{G_4}$



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

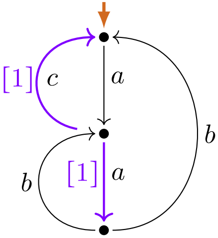
$\widehat{G_4}$



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

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

$\widehat{G_4}$

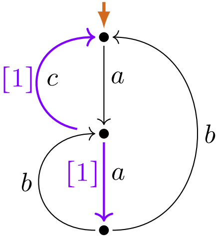


$\downarrow a$

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

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

$\widehat{G_4}$



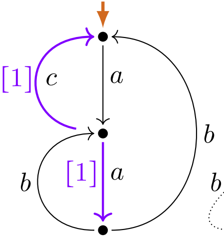
$$(a \cdot 1) \cdot ((\qquad\qquad\qquad)^* \cdot 0)$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



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

a

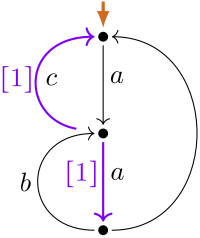
$(\qquad)^{*} \cdot 0$

b

b

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

$\widehat{G_4}$



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

$\downarrow a$

$(\qquad)^* \cdot 0$

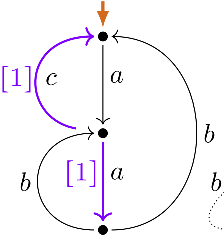
b

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

b

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

$\widehat{G_4}$



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

a $\quad$ c $\quad$ b

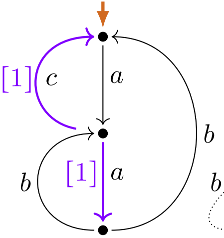
$(\qquad\qquad)^{*}\cdot 0$

a $\quad$ b

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

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

$\widehat{G_4}$



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

$$\downarrow a$$

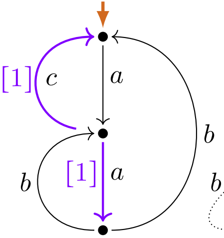
$$\Big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \Big)^* \cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



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

$$\downarrow a$$

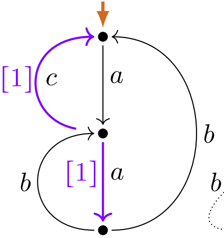
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



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

$$\downarrow a$$

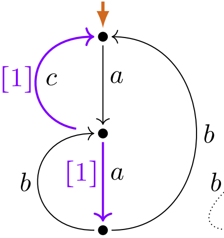
$$\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



$P(e) \rightleftharpoons G_4$

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

$$\downarrow a$$

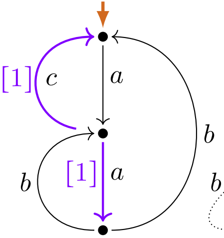
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^{*} \cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



$P(e) \mathrel{\underline{\to}} G_4$

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

$$\cancel{\downarrow}\, a$$

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

$$\downarrow a$$

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

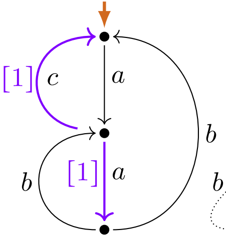
c

b

b

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

$\widehat{G_4}$



$$\mathit{P}(e) \trianglerighteq G_4$$

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

$$\downarrow a$$

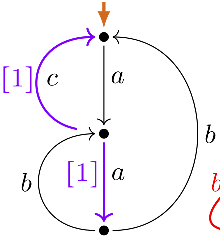
$$\big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \big)^* \cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



$P(e) \rightleftharpoons G_4$

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

$$\downarrow a$$

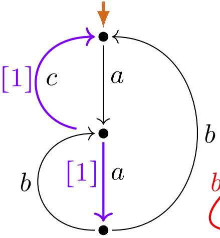
$$\left(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)) \right)^* \cdot 0$$

$$\downarrow a$$

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

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

$\widehat{G_4}$



$$P(e) \mathrel{\underline{\to}} G_4 \mathrel{\not\underline{\leftrightarrow}} P(e)$$

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

$$\Big\downarrow a$$

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

$$\Big\downarrow a$$

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

Interpretation of extracted expression

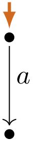
G_5

$$\mathsf{P}(e) = G_5$$

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

Interpretation of extracted expression

G_5



$$P(e) = G_5$$

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

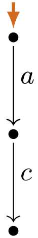
$$\downarrow a$$

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

Interpretation of extracted expression

G_5

$P(e) = G_5$



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

$$\downarrow a$$

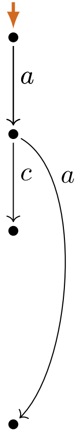
$$(1\cdot 1)\cdot\Big(\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\cdot 0\Big)$$

$$\downarrow c$$

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

Interpretation of extracted expression

G_5



$$P(e) = G_5$$

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

$$\downarrow a$$

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

$$\downarrow c$$

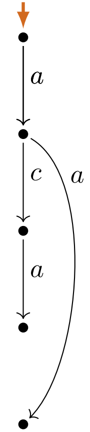
$$\Big((1 \cdot (a \cdot 1)) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\big)^*\Big) \cdot 0$$

$$a$$

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

Interpretation of extracted expression

G_5



$P(e) = G_5$

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

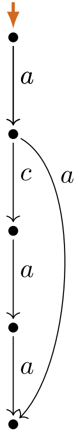
$$\big((1\cdot 1)\cdot\big(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1))\big)^{*}\big)\cdot 0\big)$$

$$a$$

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

Interpretation of extracted expression

G_5



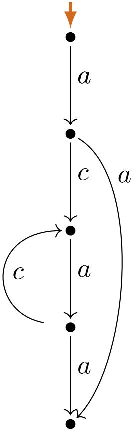
$$P(e) = G_5$$

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

a

Interpretation of extracted expression

G_5



$P(e) = G_5$

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

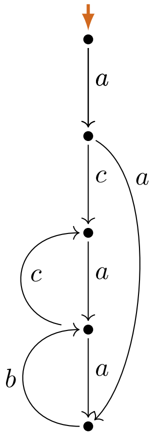
$$\Big((1 \cdot 1) \cdot \big(c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1))\big)^{*}\Big) \cdot 0\Big)$$

$$\downarrow a$$

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

Interpretation of extracted expression

G_5



$P(e) = G_5$

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

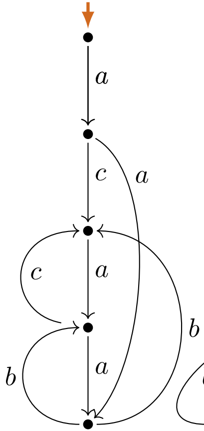
$$\big((1 \cdot 1) \cdot (c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)))^{*} \big) \cdot 0$$

$$\downarrow a$$

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

Interpretation of extracted expression

G_5



$\textsf{P}(e) = G_5$

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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

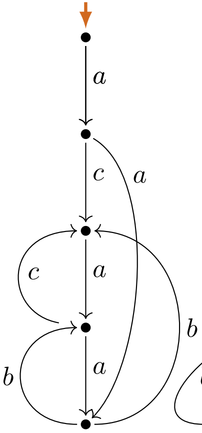
$$\big((1\cdot 1)\cdot(c\cdot(a\cdot 1)+a\cdot(b\cdot 1+b\cdot(a\cdot 1)))^{*}\big)\cdot 0\big)$$

$$\downarrow a$$

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

Interpretation of extracted expression

G_5



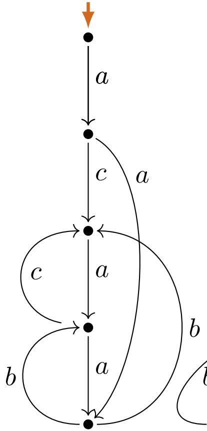
$$P(e) = G_5 \rightrightarrows G_4$$

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

Interpretation of extracted expression

G_5

$P(e) = G_5 \Rightarrow G_4 \not\simeq G_5$



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

$$\downarrow a$$

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

$$\downarrow c$$

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

$$\downarrow a$$

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

$$\downarrow a$$

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

LEE under bisimulation?

LEE under bisimulation

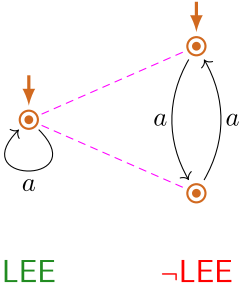
Observation

- LEE is not invariant under bisimulation.

LEE under bisimulation

Observation

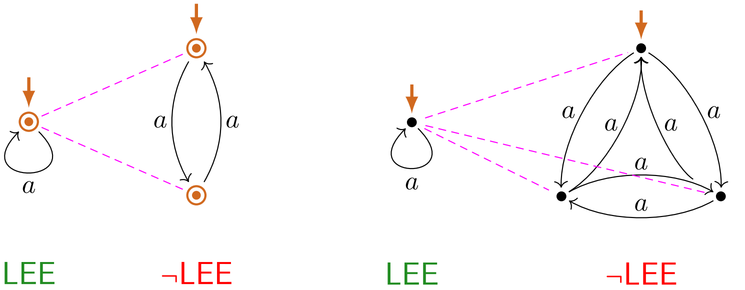
- LEE is **not** invariant under bisimulation.



LEE under bisimulation

Observation

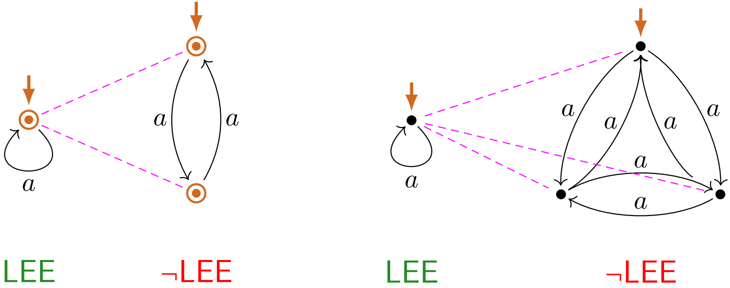
- LEE is not invariant under bisimulation.



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

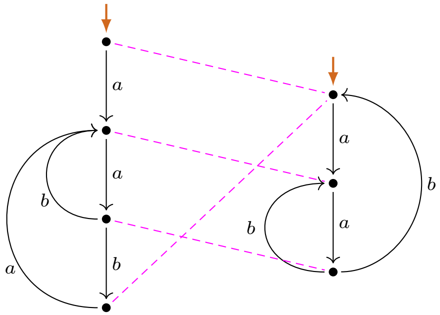
(i) LEE *is preserved by functional bisimulations:*

$$\mathsf{LEE}(G_1) \wedge G_1 \rightrightarrows G_2 \implies \mathsf{LEE}(G_2)\,.$$

Proof (Idea).

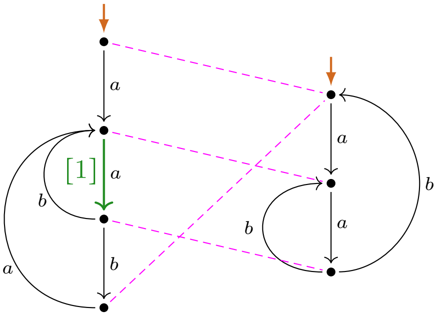
Use loop elimination in G_1 to carry out loop elimination in G_2.

Collapsing LEE-witnesses



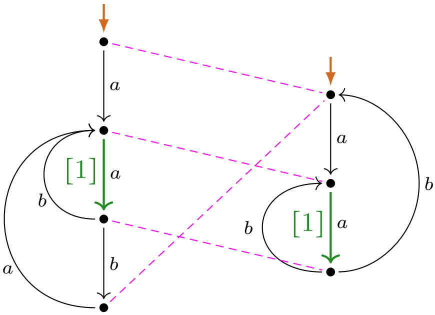
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



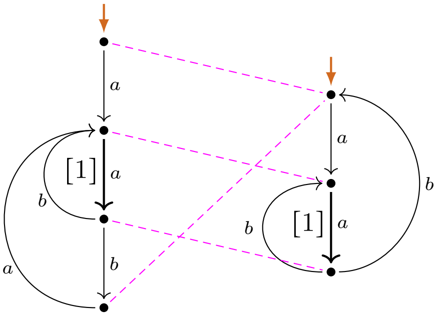
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



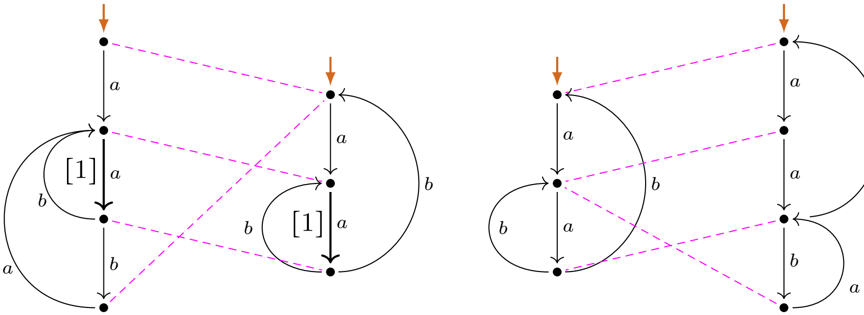
$P(a(a(b+ba))^* \cdot 0)$

Collapsing LEE-witnesses



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

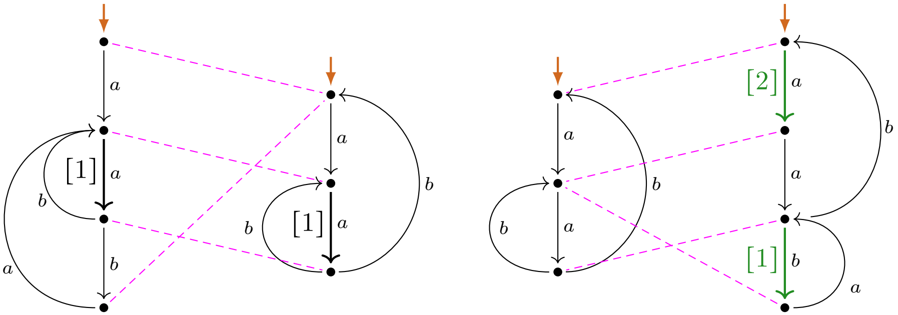
Collapsing LEE-witnesses



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

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

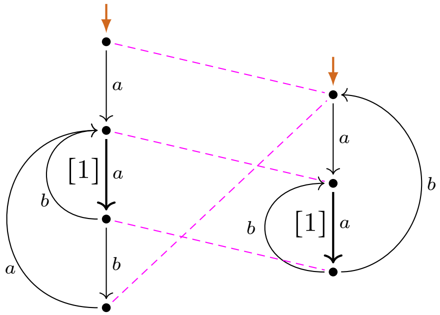
Collapsing LEE-witnesses

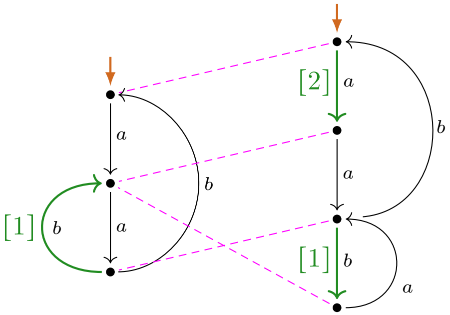


$\mathsf{P}(a(a(b+ba))^* \cdot 0)$

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

Collapsing LEE-witnesses

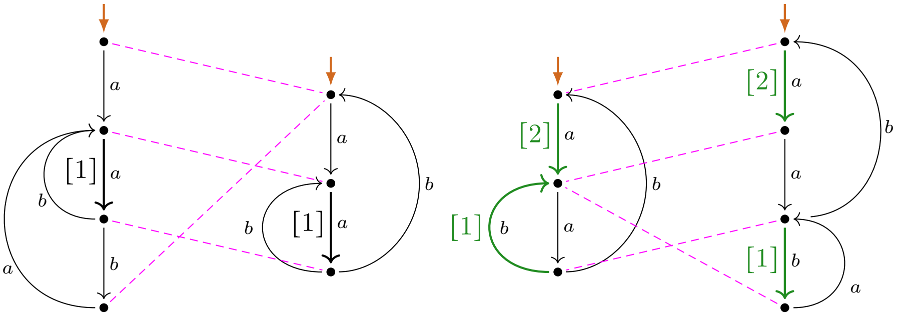




$P(a(a(b+ba))^* \cdot 0)$

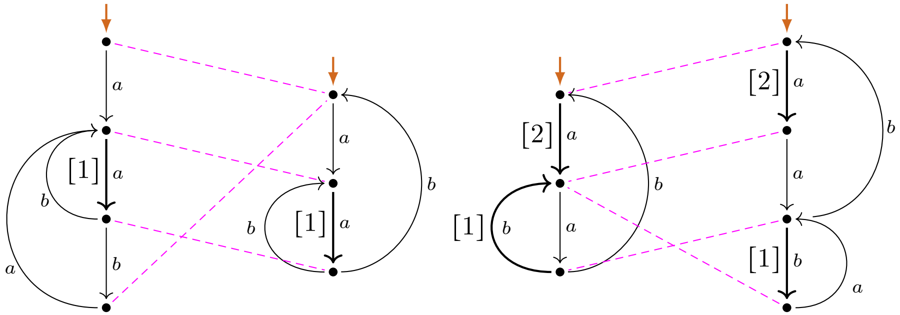
$P((aa(ba)^* \cdot b)^* \cdot 0)$

Collapsing LEE-witnesses



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

Collapsing LEE-witnesses



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

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

LEE under functional bisimulation

Lemma

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

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

Idea of Proof for (i)

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

LEE under functional bisimulation / bisimulation collapse

Lemma

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

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

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

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

Idea of Proof for (i)

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

LEE under functional bisimulation / bisimulation collapse

Lemma

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

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

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

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

Idea of Proof for (i)

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

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

LEE under functional bisimulation / bisimulation collapse

Lemma

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

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

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

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

Idea of Proof for (i)

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

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

LEE under functional bisimulation / bisimulation collapse

Lemma

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

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

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

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

Idea of Proof for (i)

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

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

LEE under functional bisimulation / bisimulation collapse

Lemma

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

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

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

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

Idea of Proof for (i)

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

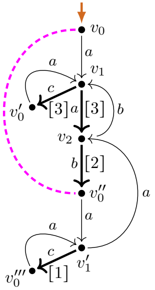
- images of loop subcharts in G_1 under $\mathrel{\underline{\rightarrow}}$ are loop subcharts of G_2.
- eliminating a loop subchart from G_2 amounts, via $\mathrel{\underline{\rightarrow}}$, to eliminating a transition induced subgraph from G_1.
- LEE is preserved by dropping transition-induced subgraphs.

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

LLEE-preserving collapse (example, corollary)

Lemma (C)

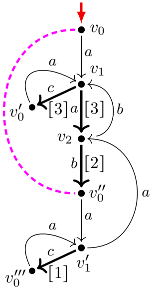
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

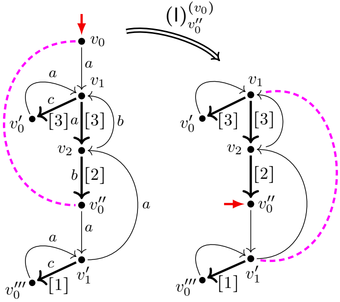
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

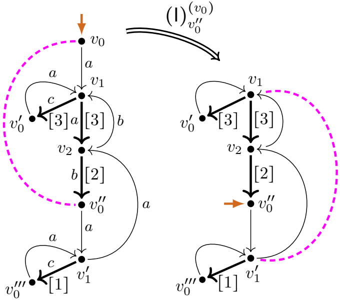
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

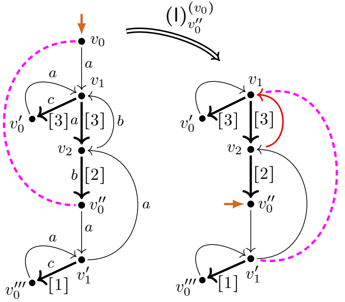
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

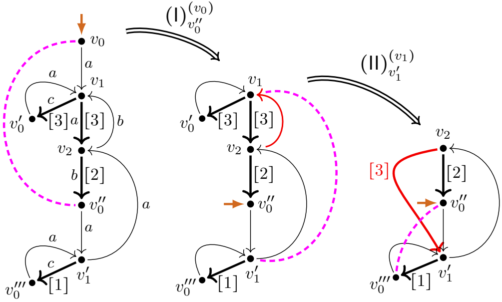
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

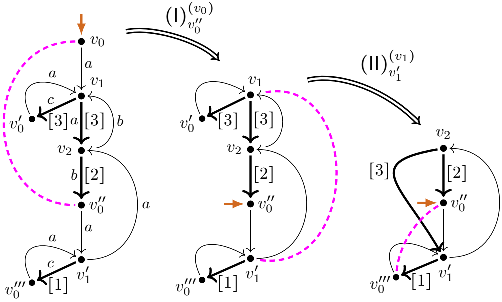
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

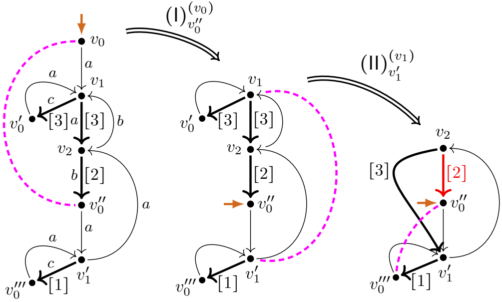
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

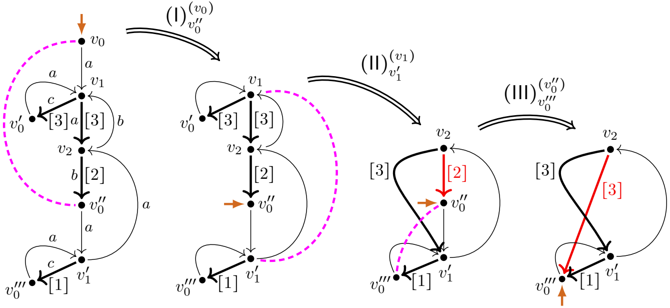
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

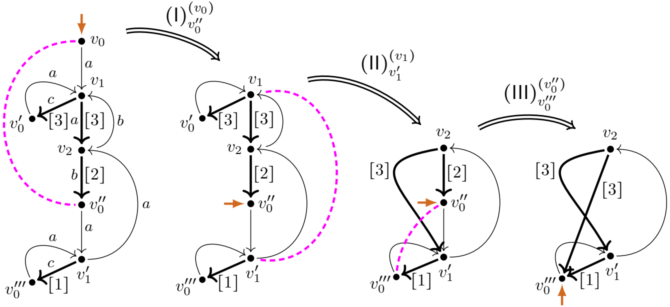
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

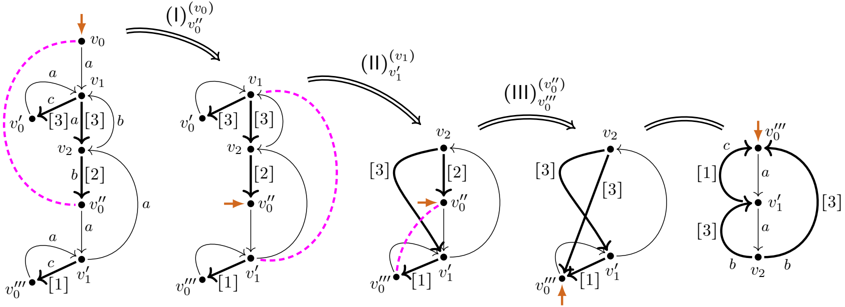
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

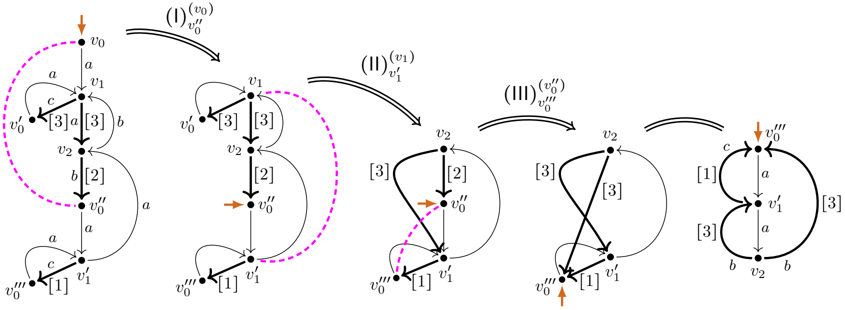
The bisimulation collapse of a LLEE-graph is again a LLEE-graph.



LLEE-preserving collapse (example, corollary)

Lemma (C)

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



Corollary

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

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

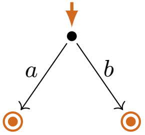
$P(uf)$

$P(uf)$

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

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

$P(uf)$



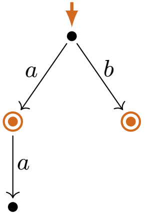
$P(uf)$

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

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

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

$P(uf)$



$P(uf)$

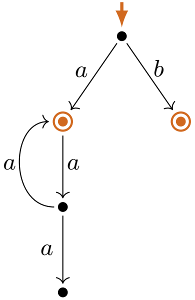
$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

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

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

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

$P(uf)$



$P(uf)$

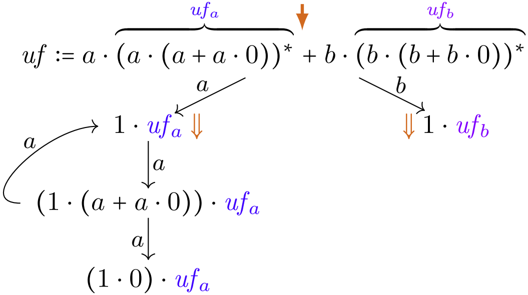
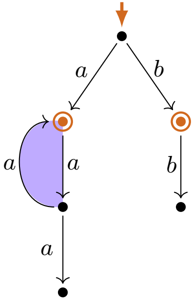


Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$

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

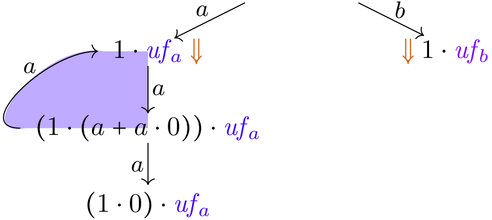
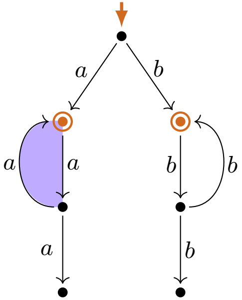


Image of *P* is **not** closed under bisimulation collapse

P(uf)



P(uf)

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

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

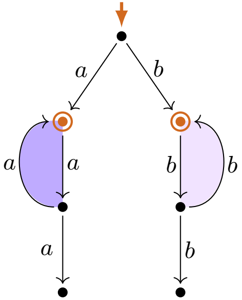
$$1 \cdot uf_a \xrightarrow{a} (1 \cdot (a + a \cdot 0)) \cdot uf_a \xrightarrow{a} (1 \cdot 0) \cdot uf_a$$

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

$$1 \cdot uf_b \xrightarrow{b} (1 \cdot (b + b \cdot 0)) \cdot uf_b$$

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

$P(uf)$



$P(uf)$

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

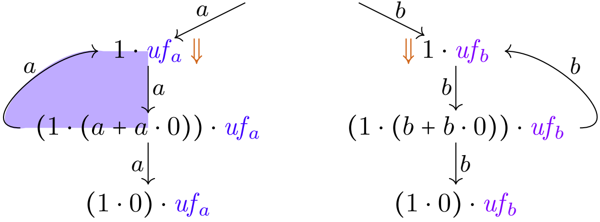
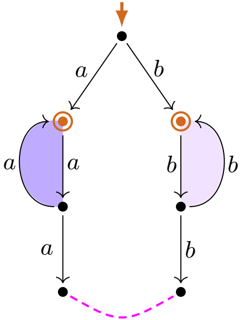
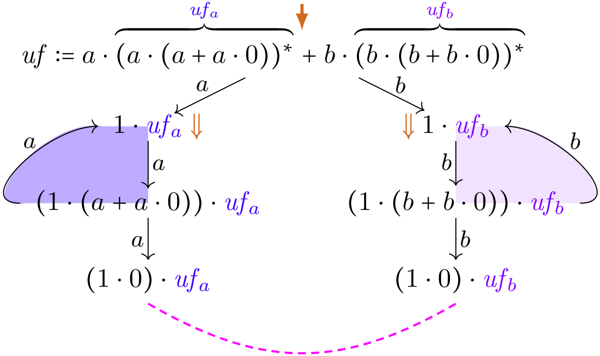


Image of P is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$



Compact process interpretation $P^{\bullet}$

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

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

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

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

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

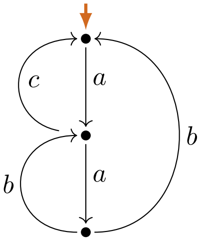
Lemma ($P^{\bullet}$ increases sharing; $P^{\bullet}$, P have same bisimulation semantics)

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

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

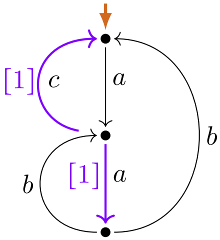
Refined extraction expression (example)

G_4



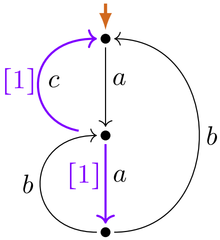
Refined extraction expression (example)

$\widehat{G_4}$



Refined extraction expression (example)

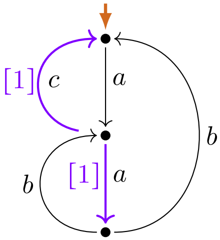
$\widehat{G_4}$



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

Refined extraction expression (example)

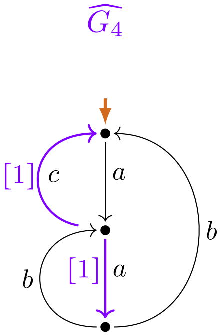
$\widehat{G_4}$



$$\overset{a}{\dashrightarrow}$$

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

Refined extraction expression (example)



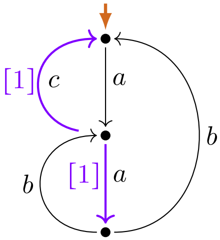
$$((1 \cdot a) \cdot (\qquad\qquad)^*) \cdot 0$$

$$\Big\downarrow a$$

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

Refined extraction expression (example)

$\widehat{G_4}$



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

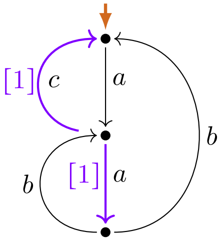
a

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

b $\quad b$

Refined extraction expression (example)

$\widehat{G_4}$



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

a

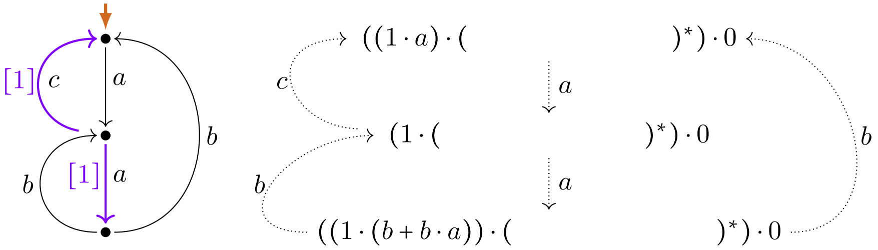
$(1 \cdot (\qquad\qquad)^*) \cdot 0$

$b \qquad b$

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

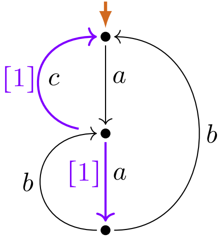
Refined extraction expression (example)

$\widehat{G_4}$



Refined extraction expression (example)

$\widehat{G_4}$



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

c $\quad$ $\downarrow a$ $\quad$ b

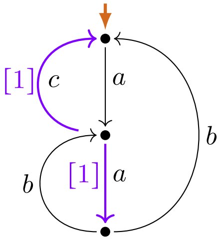
$$(1\cdot(c\cdot a + a\cdot(b+b\cdot a))^*)\cdot 0$$

b $\quad$ $\downarrow a$

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

Refined extraction expression (example)

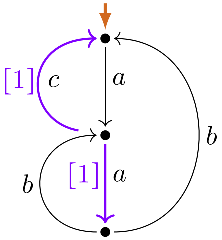
$\widehat{G_4}$



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

Refined extraction expression (example)

$\widehat{G_4}$

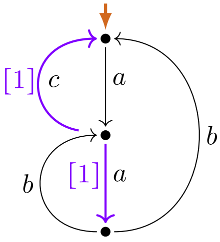


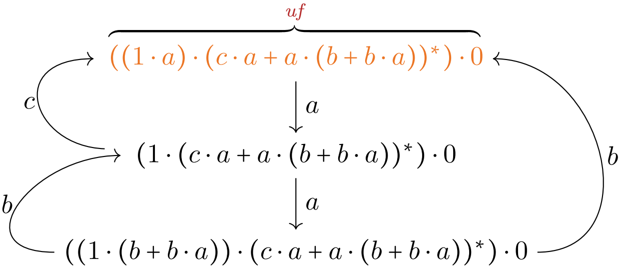
$$\begin{tikzcd}
((1 \cdot a) \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0 \arrow[d, "a"] \\
(1 \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0 \arrow[d, "a"] \arrow[u, bend left, "c"] \\
((1 \cdot (b + b \cdot a)) \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0 \arrow[u, bend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

Refined extraction expression (example)

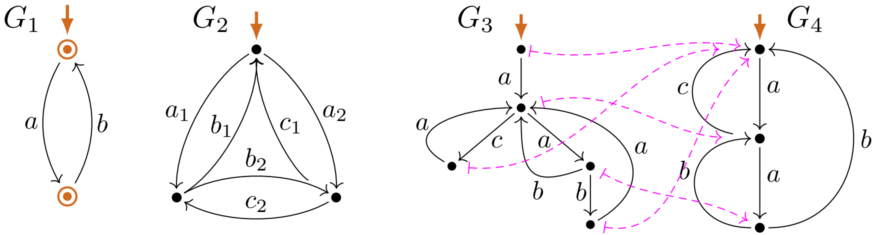
$\widehat{G_4}$

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





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



not P-expressible P-/$P^{\bullet}$-expressible $P^{\bullet}$-expressible

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

Summary and outlook

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

Summary and outlook

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

Summary and outlook

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

Summary and outlook

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

Summary and outlook

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

Outlook on an extension:

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

Summary and outlook

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

Outlook on an extension:

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

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

Summary and outlook

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

Outlook on an extension:

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

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

Resources

- Slides/extended abstract on `clegra.github.io`
  - slides: `.../lf/TG-2024.pdf`
  - extended abstract: `.../lf/closing-bs-i-pi-us1f.pdf`

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

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

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

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

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

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

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

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

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

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

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

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

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

$\mathbf{0} \xmapsto{L}$ 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

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