Towards a Characterisation of the Graph Structure of Process Interpretations of Regular Expressions

Clemens Grabmayer

https://clegra.github.io

Department of Computer Science

L'Aquila, Italy

TERMGRAPH 2026

Lisbon

July 19, 2026

Overview

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

Overview

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

Overview

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

Overview

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

Overview

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

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

$\mathbf{0} \overset{P}{\longmapsto}$ deadlock δ, no observable behaviour

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

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

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

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

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

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

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

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

$\mathbf{0} \overset{P}{\longmapsto}$ deadlock δ, no observable behaviour

$\mathbf{0}^* \overset{P}{\longmapsto}$ 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 $P(e)$)

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

$\mathbf{0} \overset{P}{\longmapsto}$ deadlock δ, no observable behaviour

$\mathbf{1} \overset{P}{\longmapsto}$ 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 $P(e)$)

Regular Expressions

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

Regular expressions over alphabet A with unary Kleene star:

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

- symbol $\mathbf{0}$ instead of $\varnothing$

Regular Expressions

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

Regular expressions over alphabet A with unary Kleene star:

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

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

Regular Expressions

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

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

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

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

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

Regular Expressions

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

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

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

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

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

Regular Expressions (1-free)

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

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

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

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

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

Definition (for process interpretation)

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

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

Regular Expressions (1-free)

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

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

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

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

Definition (for process interpretation)

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

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

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

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

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

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

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

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

Definition (for process interpretation)

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

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

Regular Expressions $((*/\not{1})/1\text{-free})$

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

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

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

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

Definition (for process interpretation)

The set $\mathit{RExp}^{(\not{1})}(A)$ of 1-free regular expressions over A is defined by:

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

the set $\mathit{RExp}^{(*/\not{1})}(A)$ of under-star-1-free $(*/\not{1})$ reg. expr. over A by:

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

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

$$\mathbf{0} \xmapsto{P} \text{deadlock } \delta\text{, no observable behaviour}$$

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

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

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

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

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

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

Process interpretation P (formally)

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

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

Process interpretation P (formally)

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

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

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

Process interpretation P (formally)

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

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

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

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

Process interpretation P (formally)

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

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

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

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

Process interpretation P (formally)

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

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

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

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

Definition

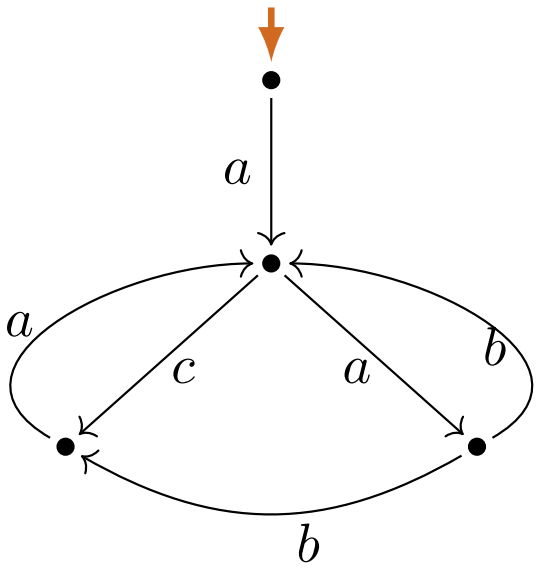
The process interpretation $P(e)$ of a regular expression e:

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

The process semantics $[\![e]\!]_P$ of e is: $[\![e]\!]_P := [P(e)]_{\underline{\leftrightarrow}}$.

P-expressibility and $[\![\cdot]\!]_P$-expressibility

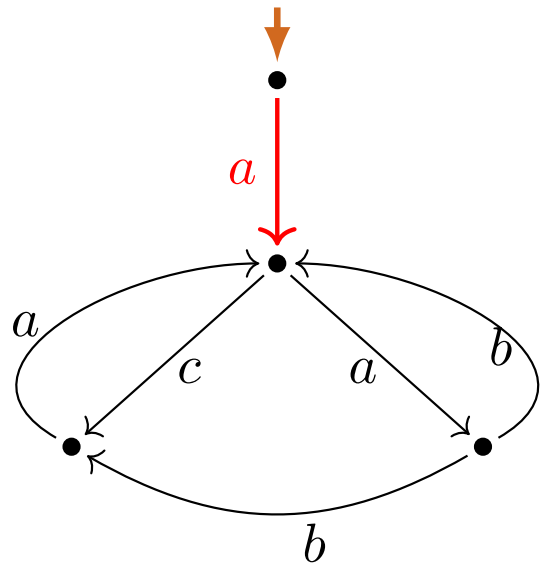
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

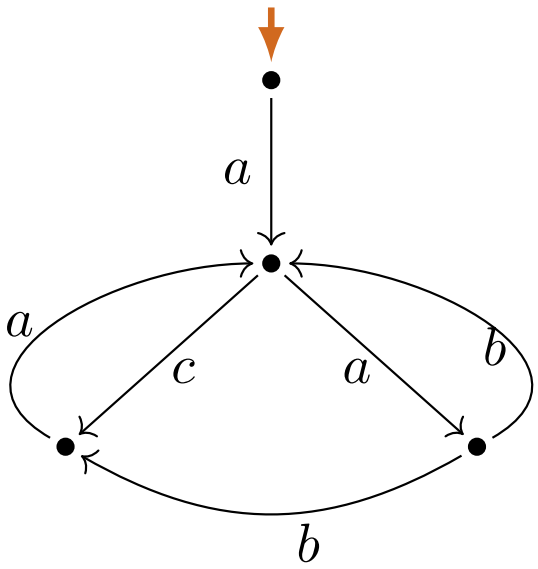
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

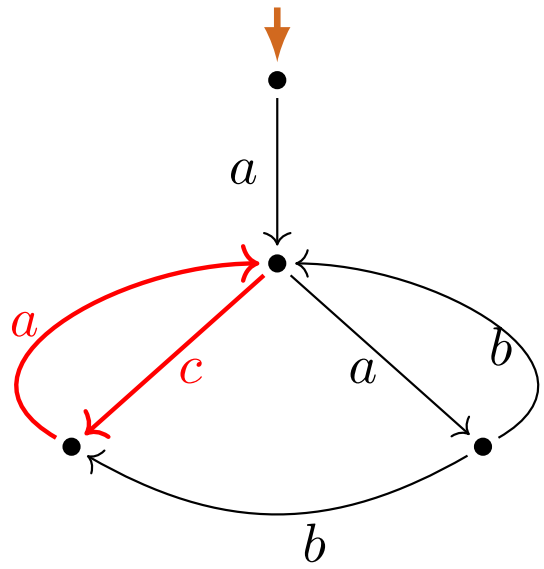
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

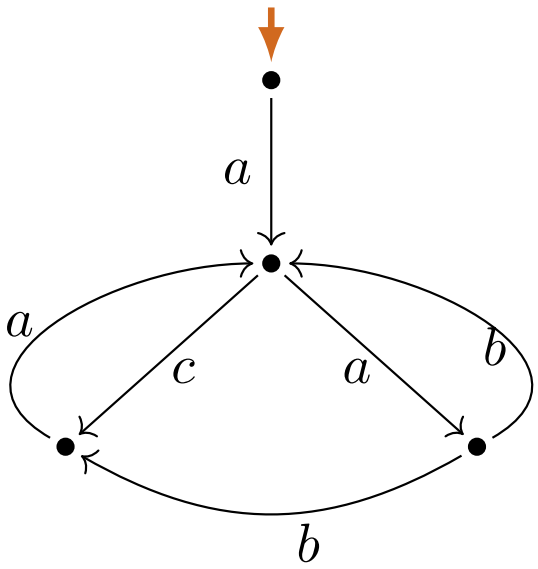
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

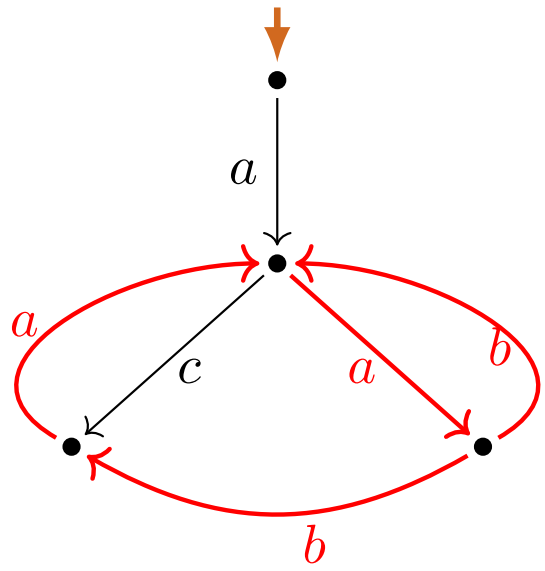
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

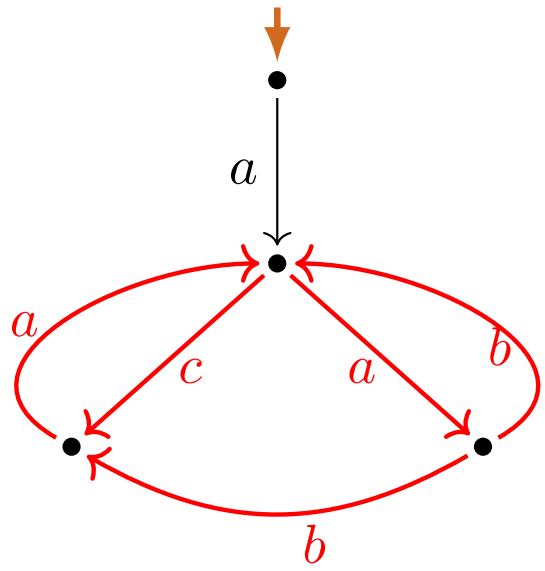
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

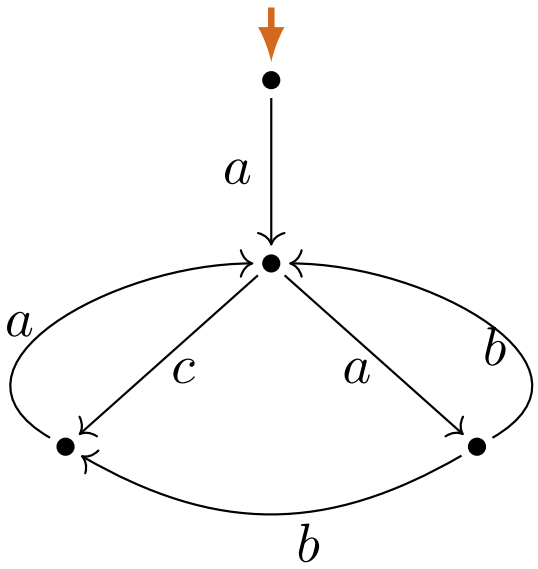
G_4



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

P-expressibility and $[\![\cdot]\!]_P$-expressibility

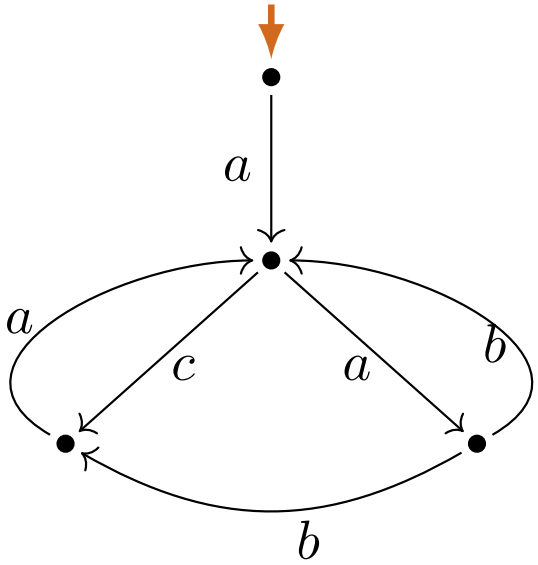
G_4



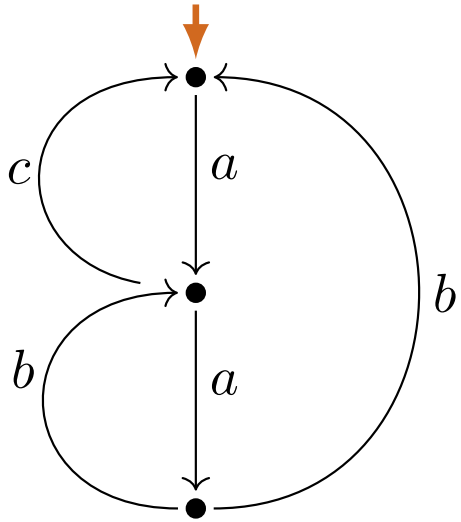
$$P(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^{f})$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility

G_4

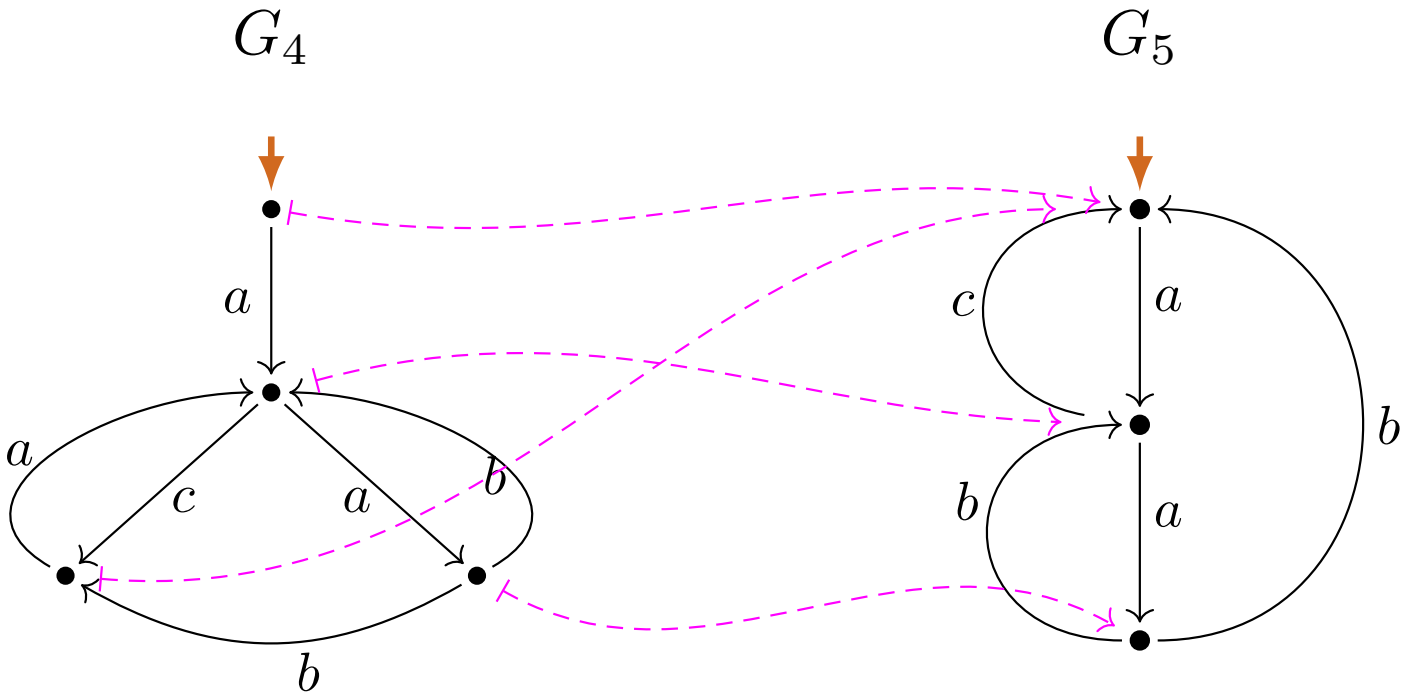


G_5



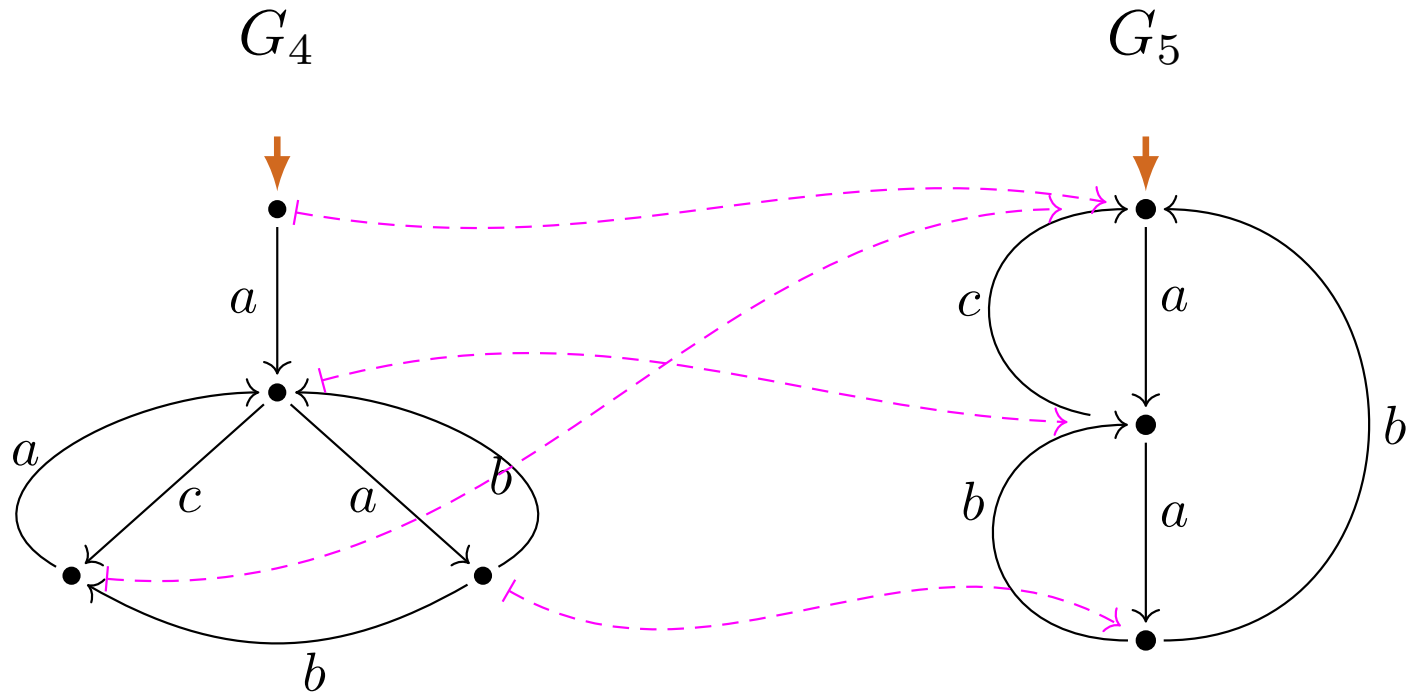
$$P\Big(\overbrace{(a\cdot(c\cdot a+a\cdot(b+b\cdot a))^*)\cdot 0}^{f}\Big)$$

P-expressibility and $[\![\cdot]\!]_P$-expressibility



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

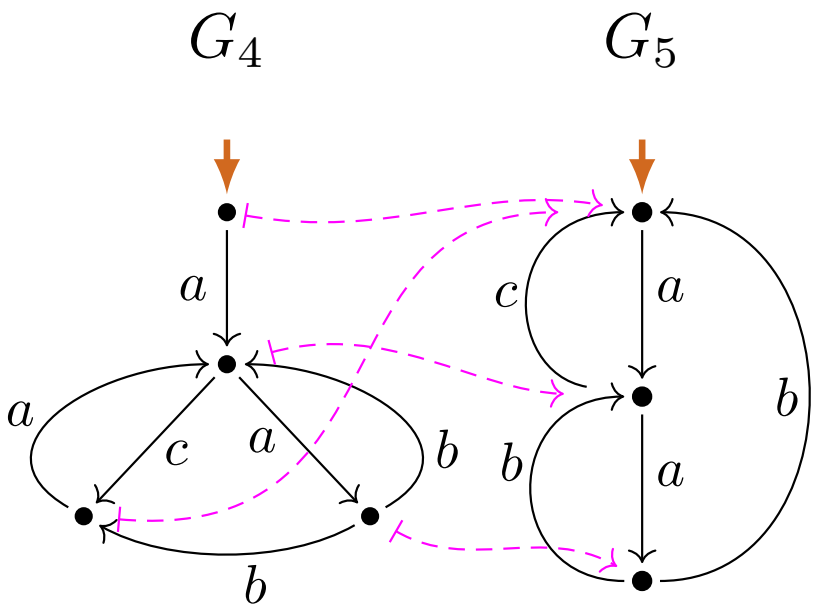
P-expressibility and $[\![\cdot]\!]_P$-expressibility



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

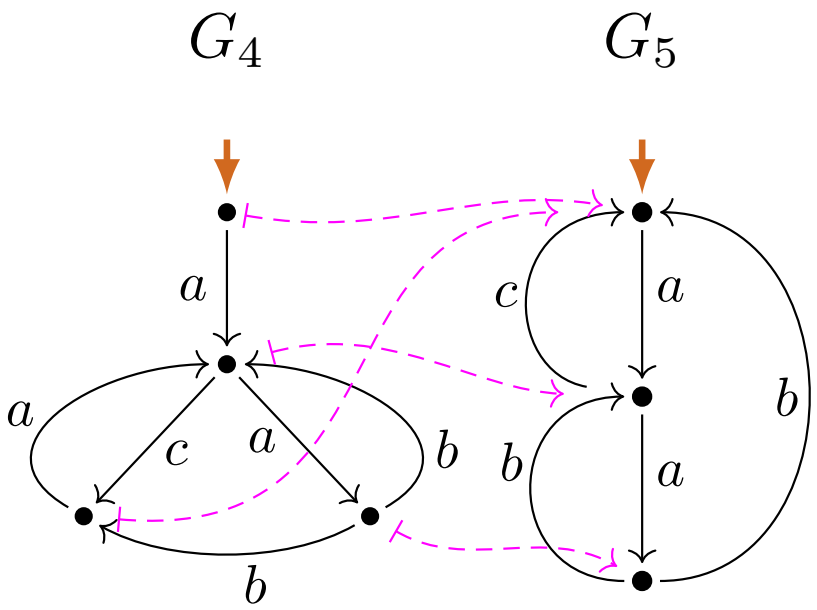
$G_4 \in [\![f]\!]_P$ $\qquad$ $G_5 \in [\![f]\!]_P$

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



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

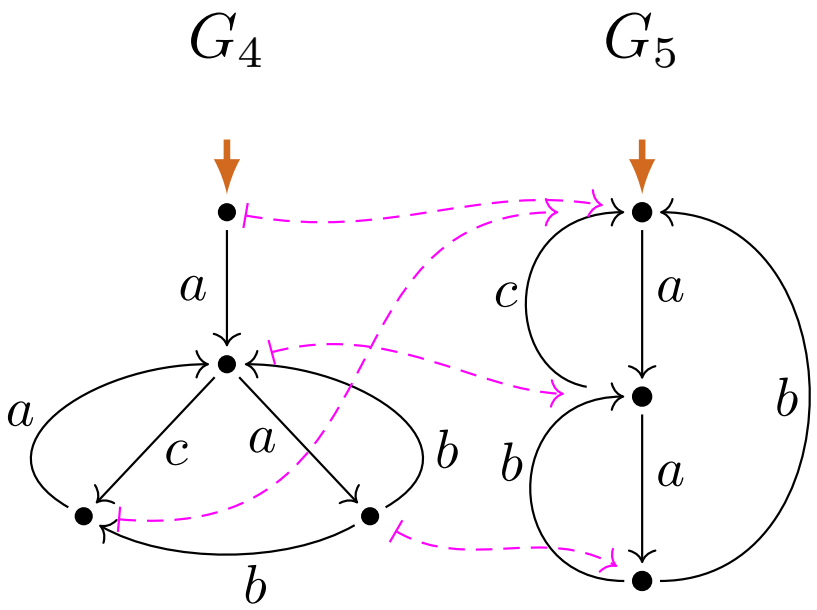
P-expressibility and $[\![\cdot]\!]_P$-expressibility/ (examples)



P-expressible

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

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

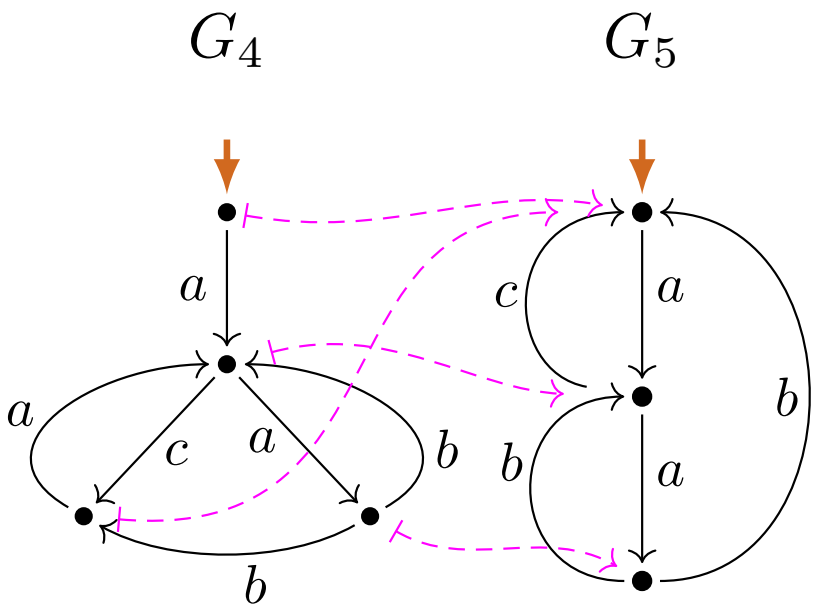


P-expressible

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

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

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

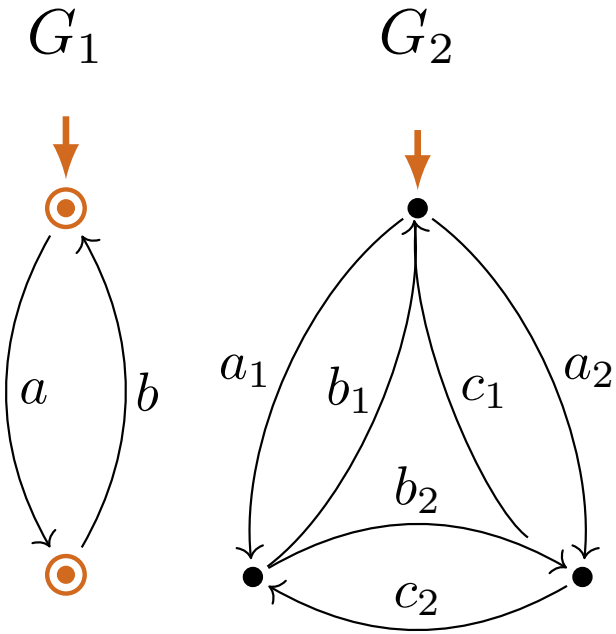


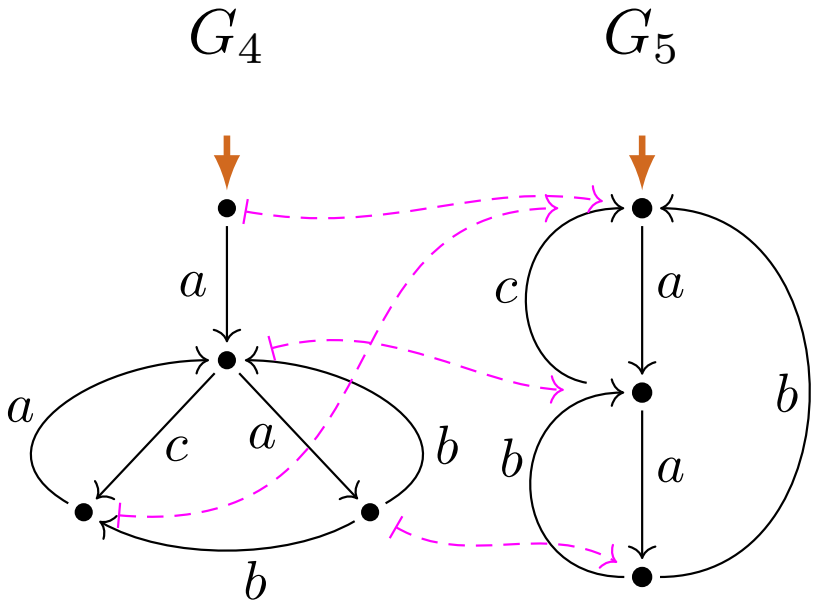
P-expressible
$[\![\cdot]\!]_P$-expressible

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

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

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





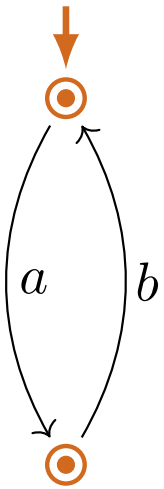
P-expressible
$[\![\cdot]\!]_P$-expressible

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

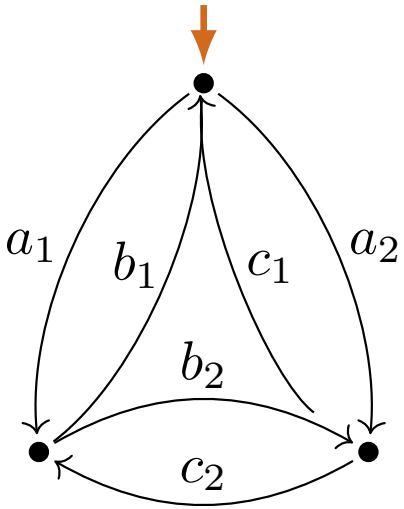
$$P((a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0)$$

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

G_1

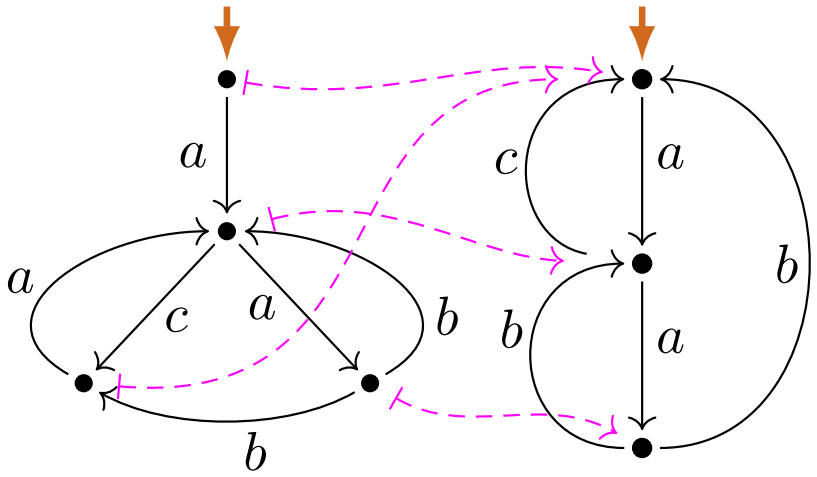


G_2



not P-expressible

G_4



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

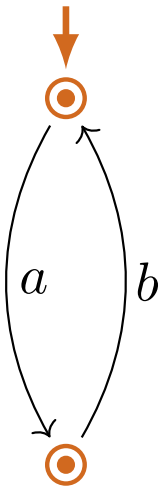
G_5

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

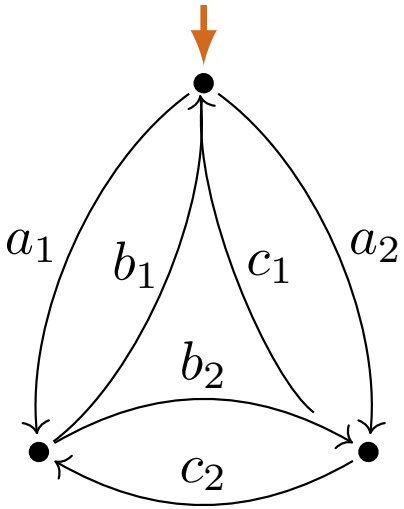
$$P((a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0)$$

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

G_1

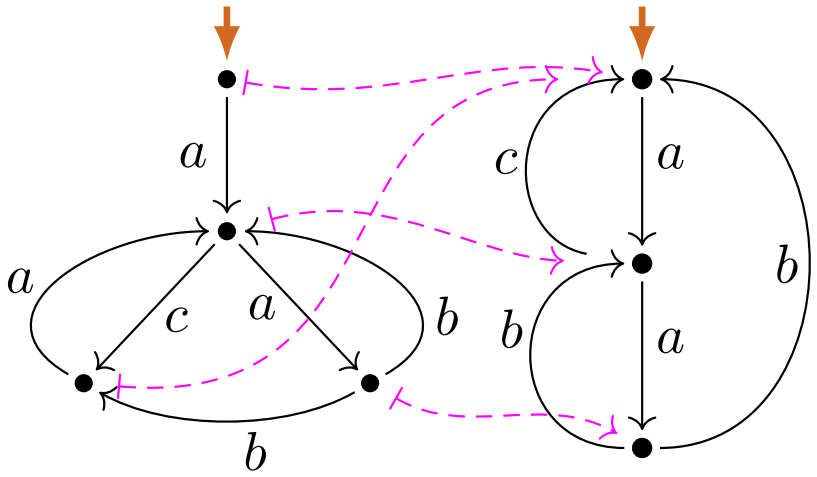


G_2



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

G_4



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

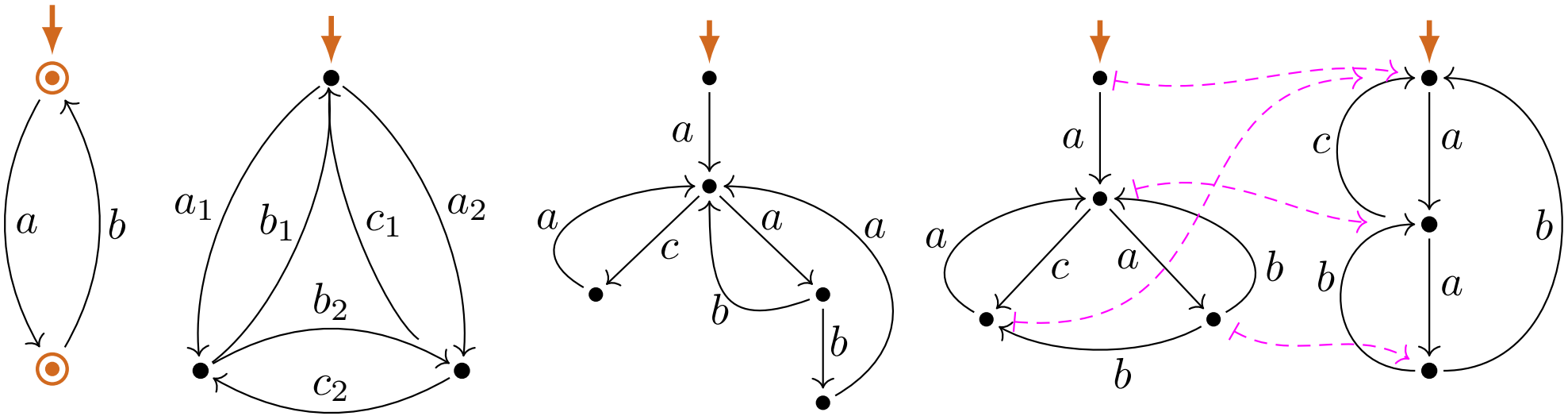
G_5

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

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

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

G_1 G_2 G_3 G_4 G_5



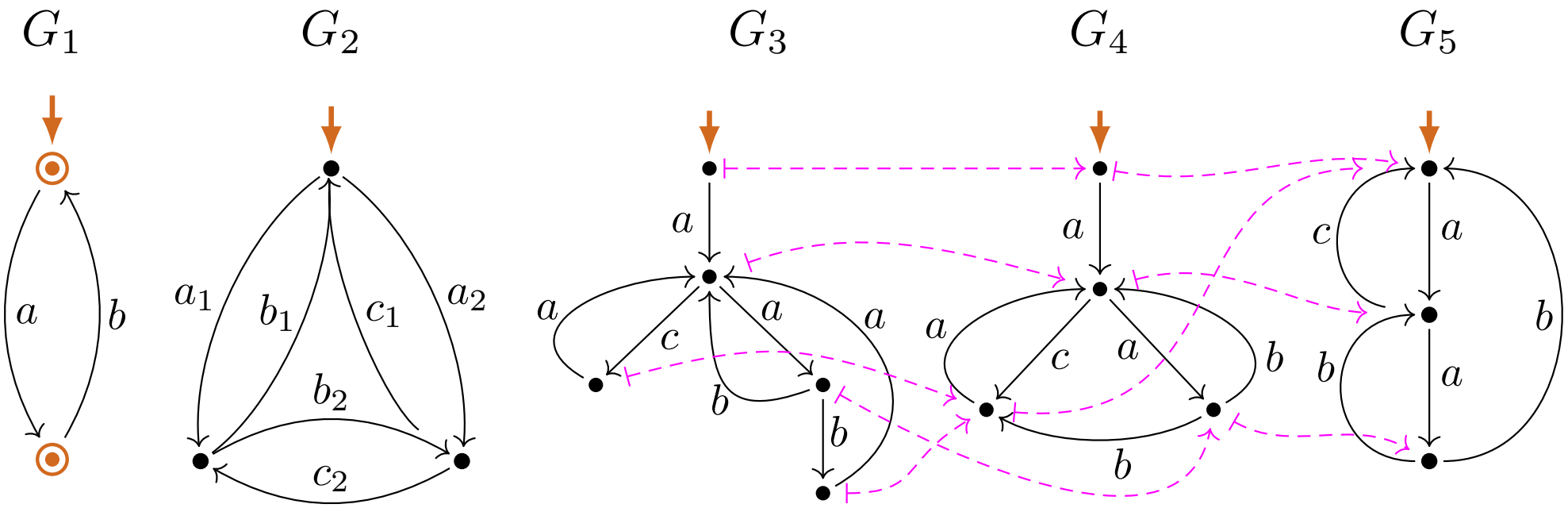
not P-expressible
not $[\![\cdot]\!]_P$-expressible

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

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

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

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



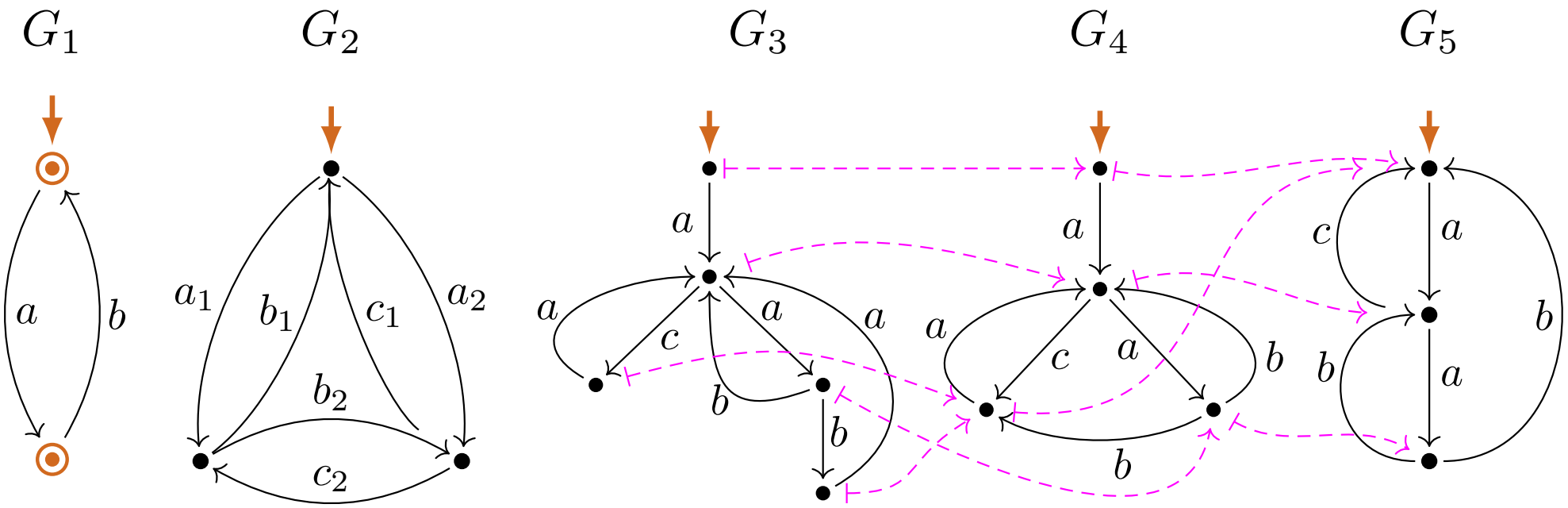
not P-expressible
not $[\![\cdot]\!]_P$-expressible

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

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

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

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



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

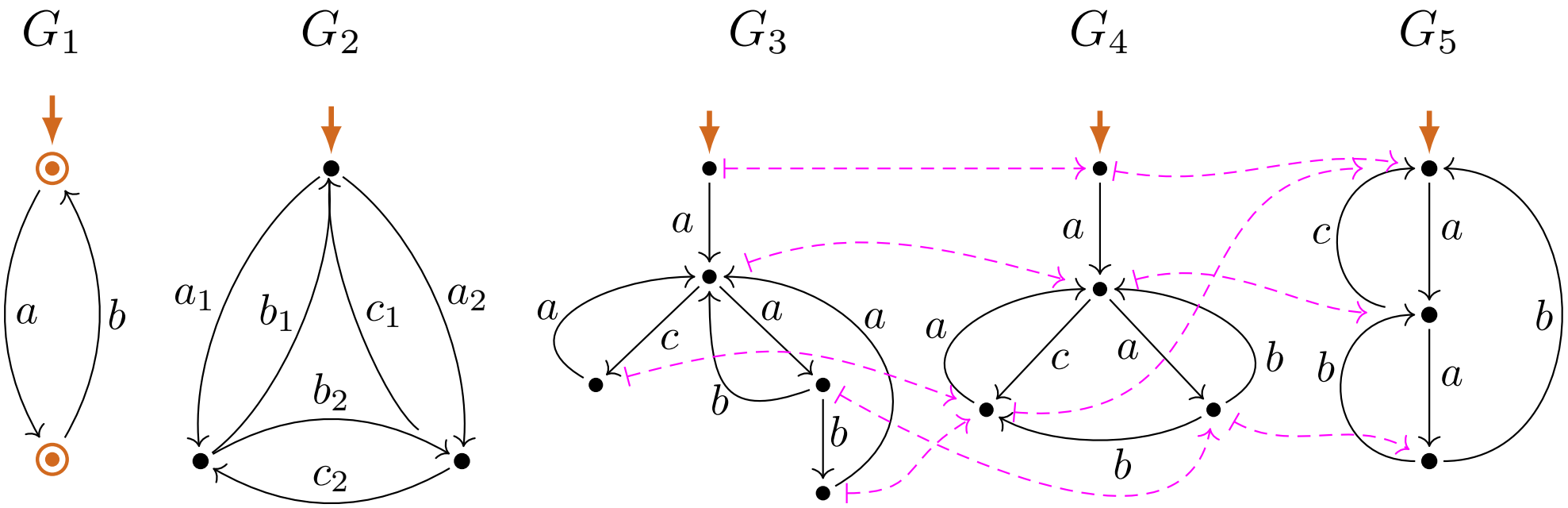
$[\![\cdot]\!]_P$-expressible

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

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

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

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



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

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

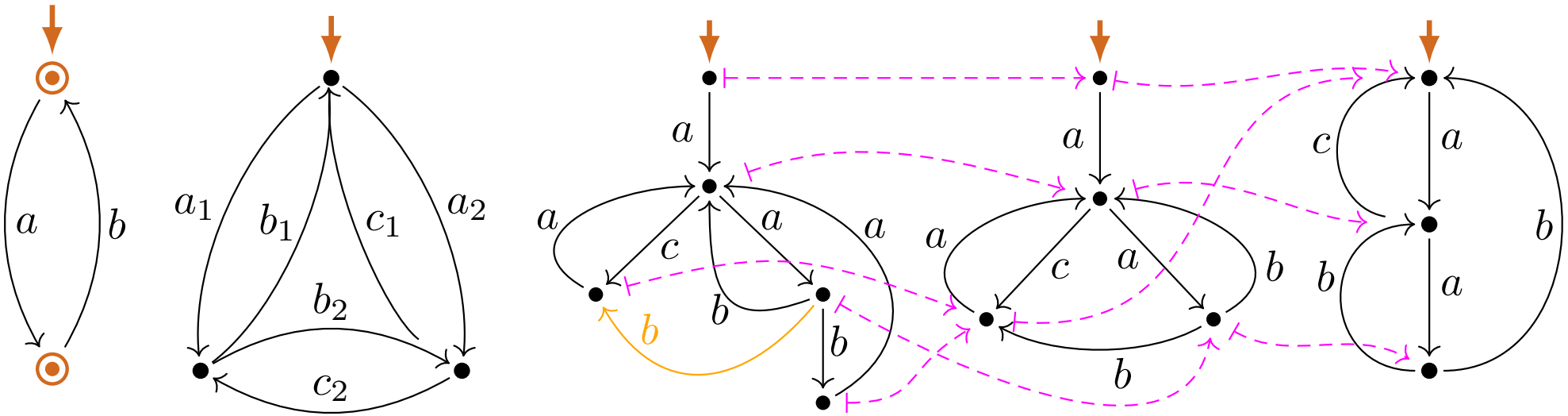
P-expressible
$[\![\cdot]\!]_P$-expressible

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

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

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

G_1 G_2 G_3^+ G_3 G_4 G_5



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

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

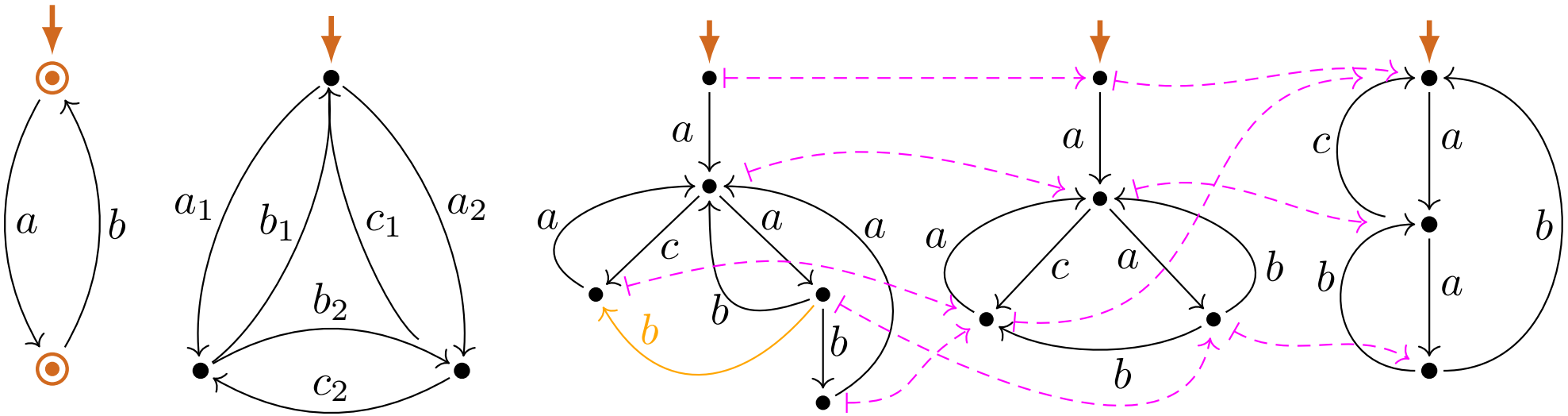
P-expressible
$[\![\cdot]\!]_P$-expressible

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

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

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

G_1 G_2 G_3^+ G_3 G_4 G_5



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

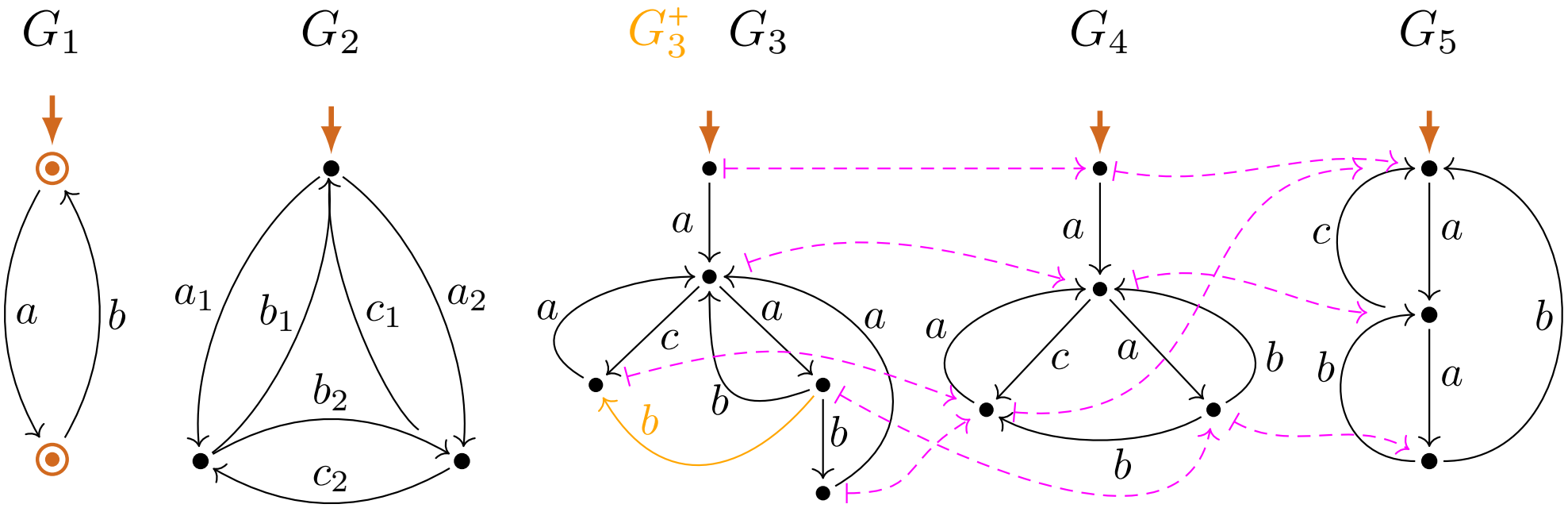
P-expressible ?
$[\![\cdot]\!]_P$-expressible

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

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

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

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



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

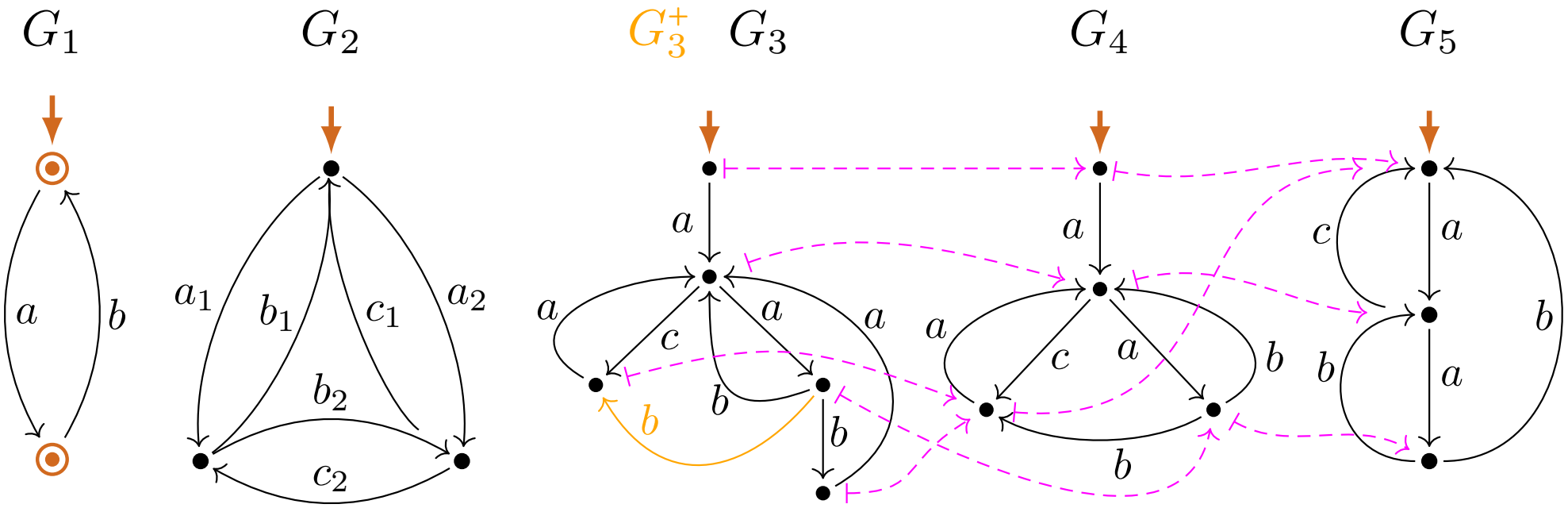
P-expressible ?
$[\![\cdot]\!]_P$-expressible

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

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

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

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



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

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

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

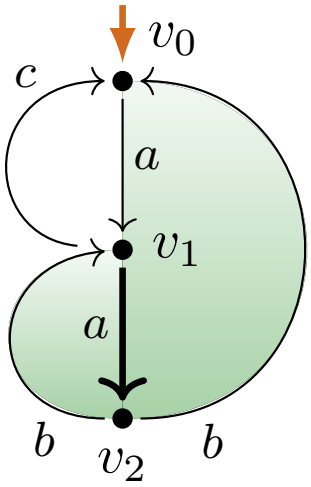
P-expressible ?
$[\![\cdot]\!]_P$-expressible

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

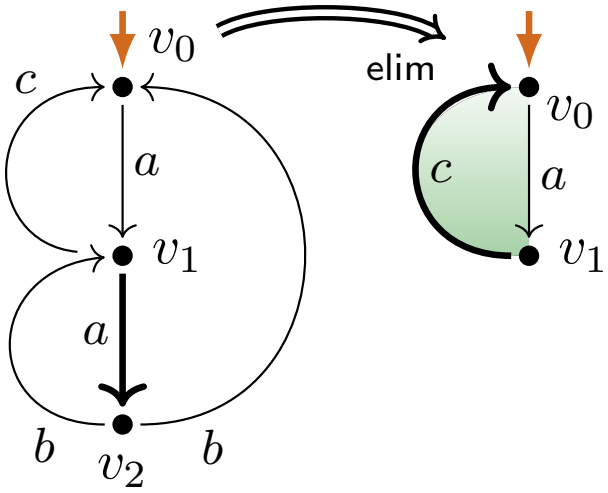
LLEE and 1-LLEE

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\underline{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\underline{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\underline{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

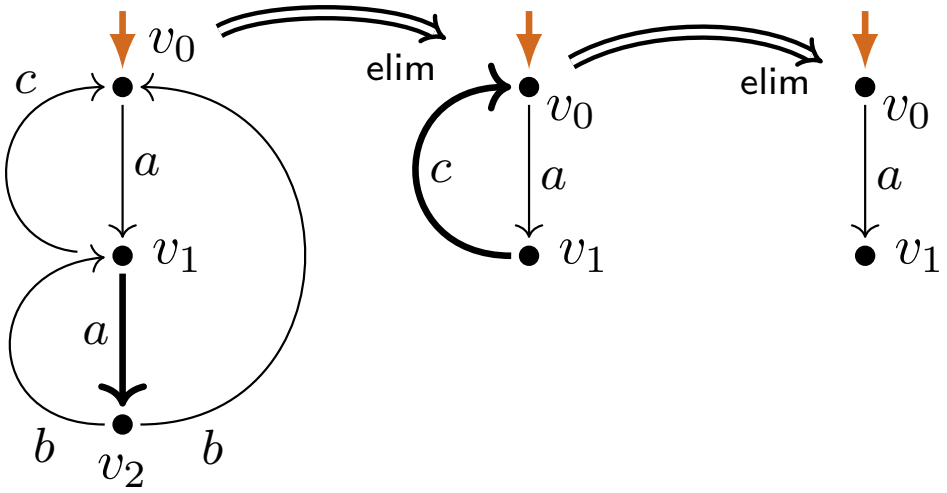
Loop existence and elimination



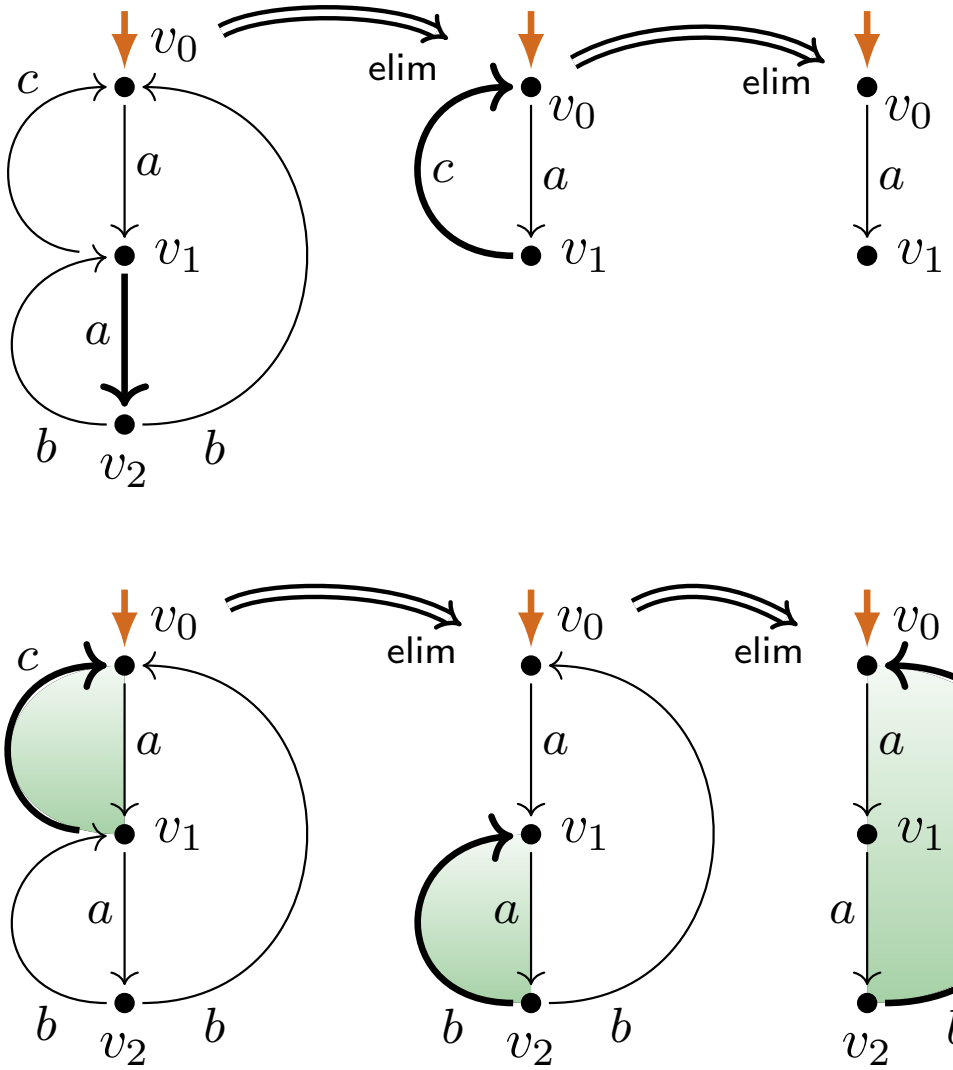
Loop existence and elimination



Loop existence and elimination



Loop existence and elimination



LEE

Definition

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

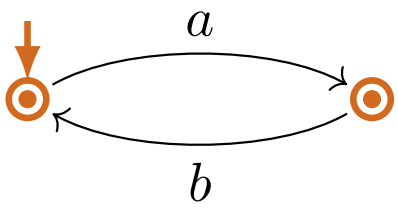
$$\exists G_0 \left(G \longrightarrow^{*}_{\mathsf{elim}} G_0 \not\longrightarrow_{\mathsf{elim}} \right.$$
$$\left. \wedge\ G_0 \text{ permits no infinite path} \right).$$

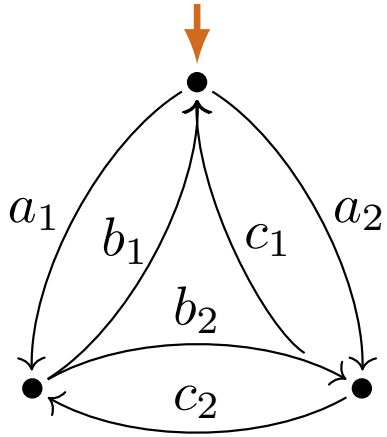
LEE

Definition

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

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



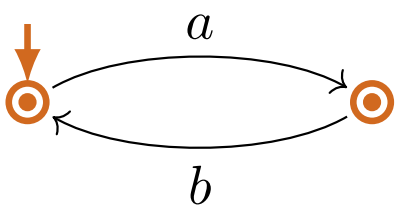


LEE

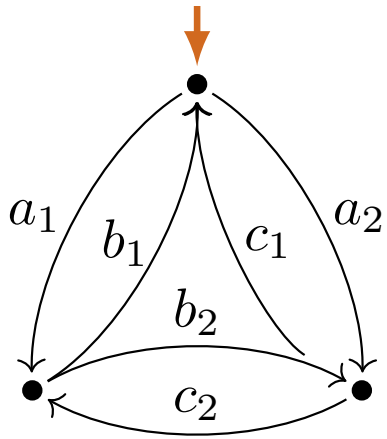
Definition

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

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

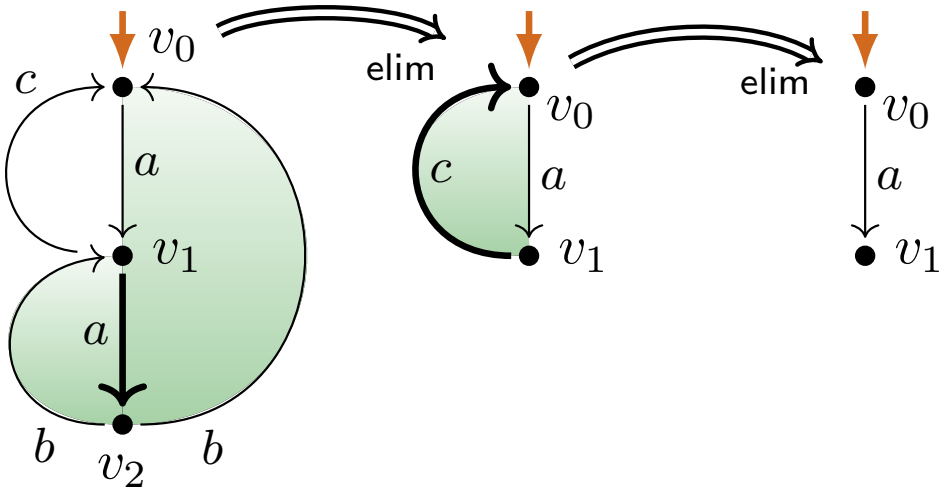


not LEE

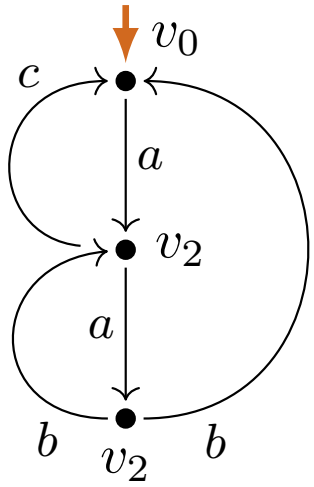


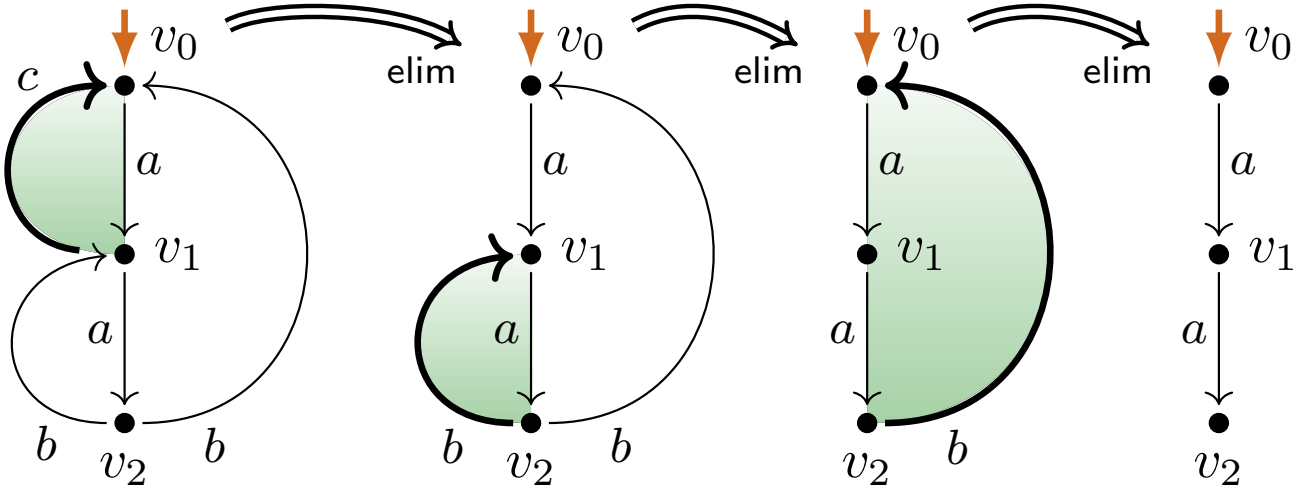
not LEE

LEE

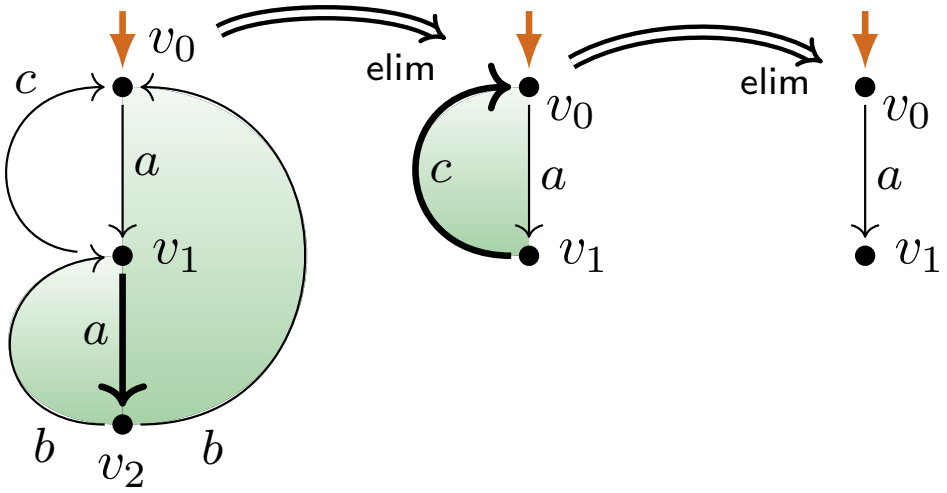

v_0
c
a
v_1
a
b
v_2
b
elim
v_0
c
a
v_1
elim
v_0
a
v_1

LEE

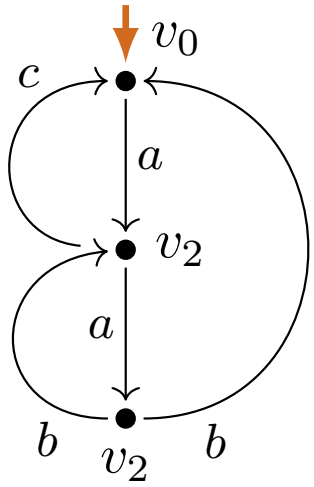

v_0
c
a
v_2
a
b
v_2
b

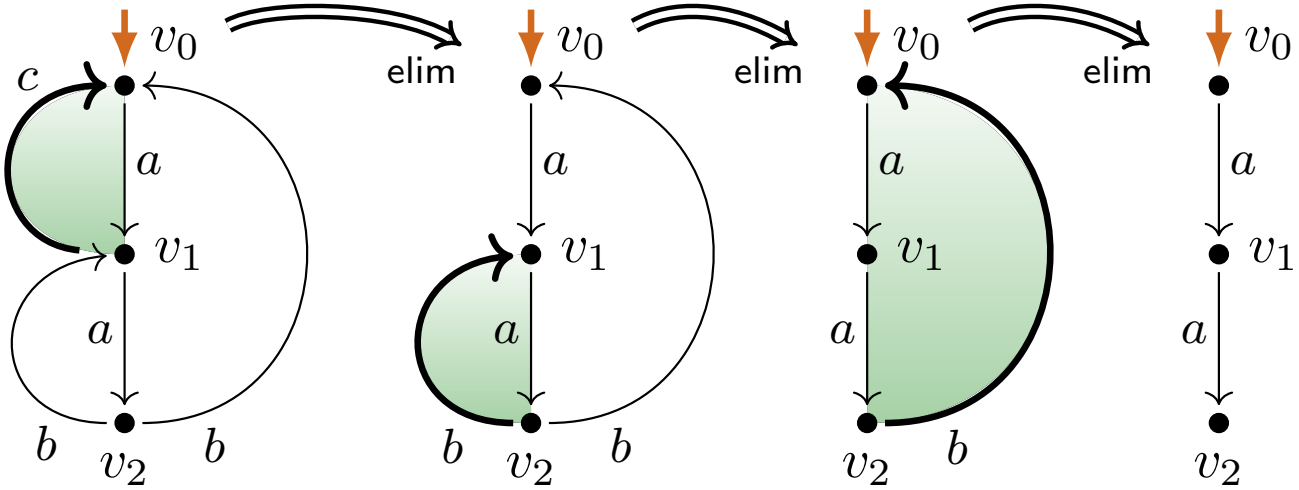

v_0
c
a
v_1
a
b
v_2
b
elim
v_0
a
v_1
a
b
v_2
b
elim
v_0
a
v_1
a
v_2
b
elim
v_0
a
v_1
a
v_2

Layered LEE

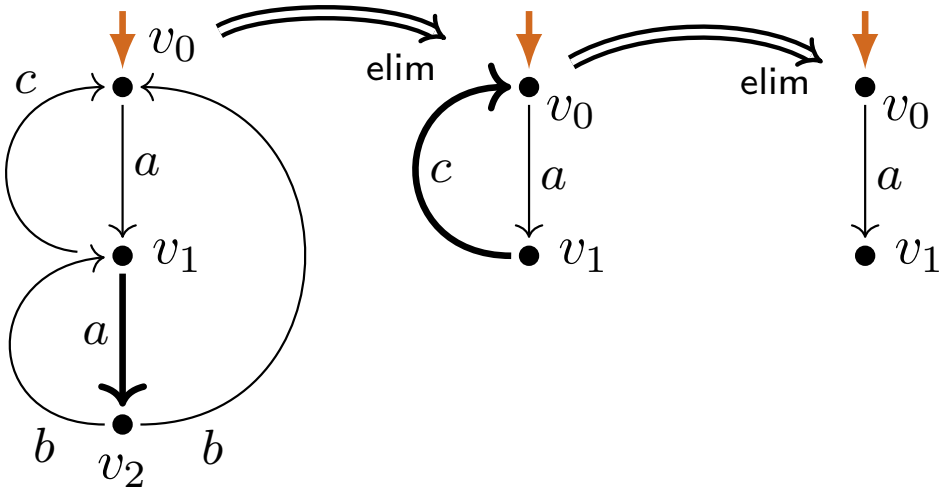


layered LEE
LLEE

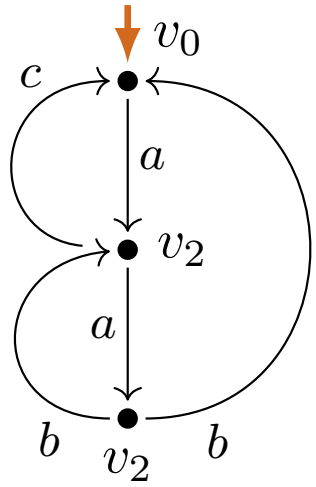


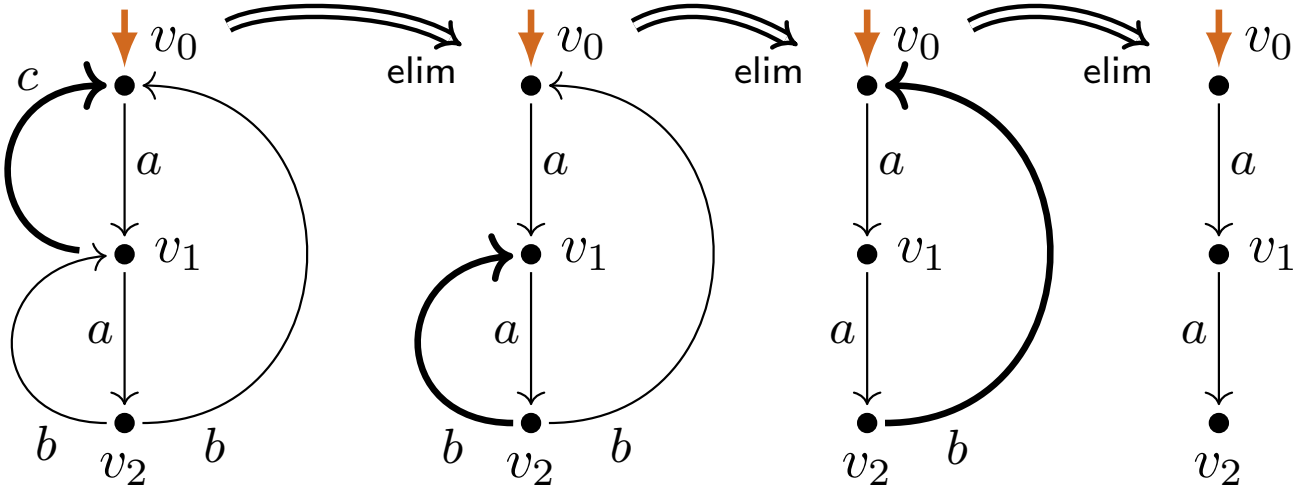


Layered LEE

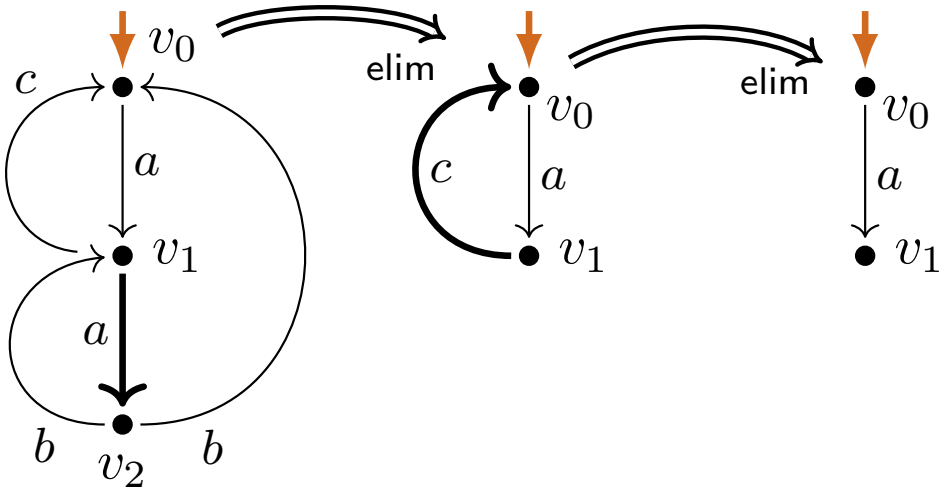


layered LEE
LLEE



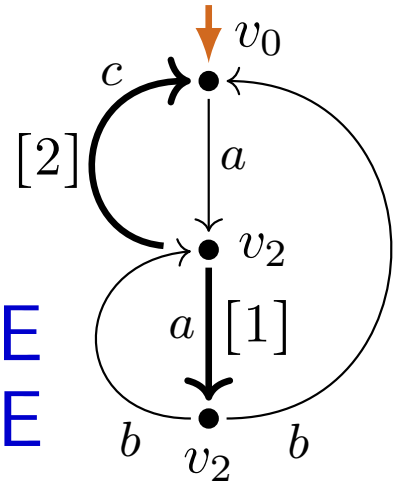


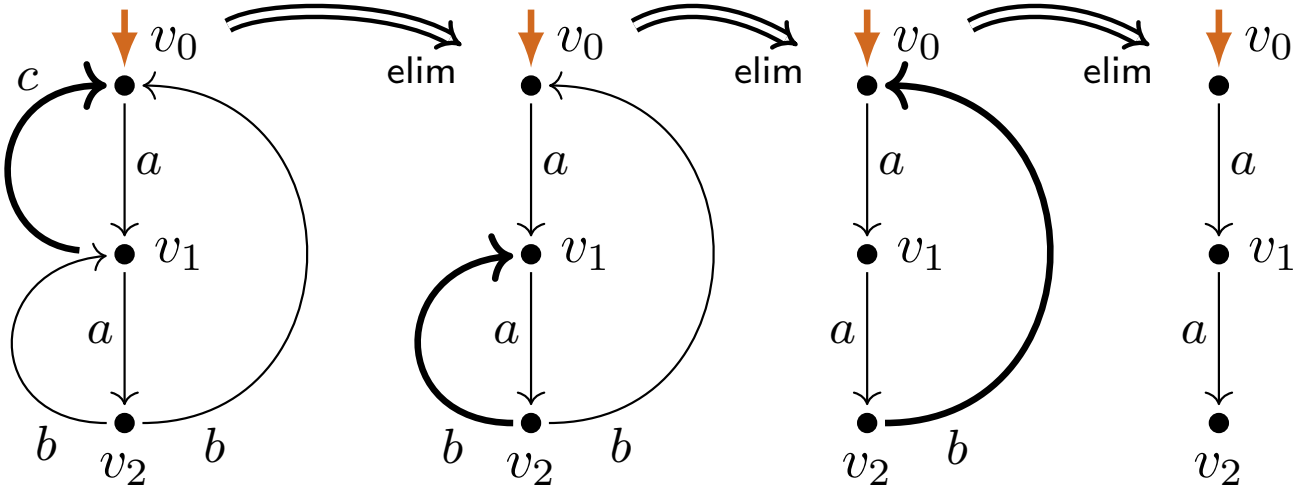
Layered LEE



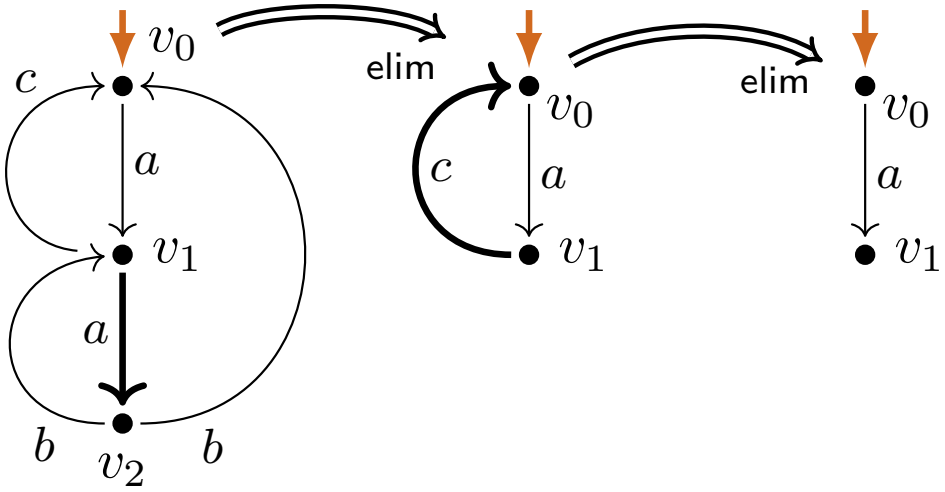
layered LEE

LLEE



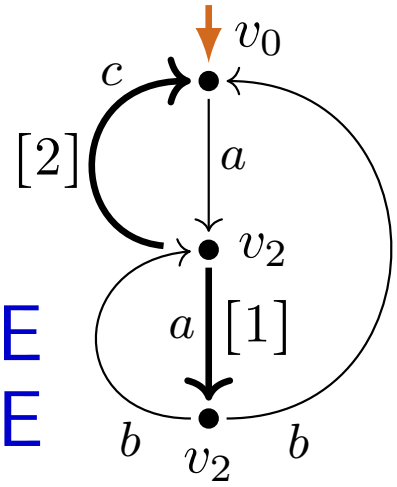


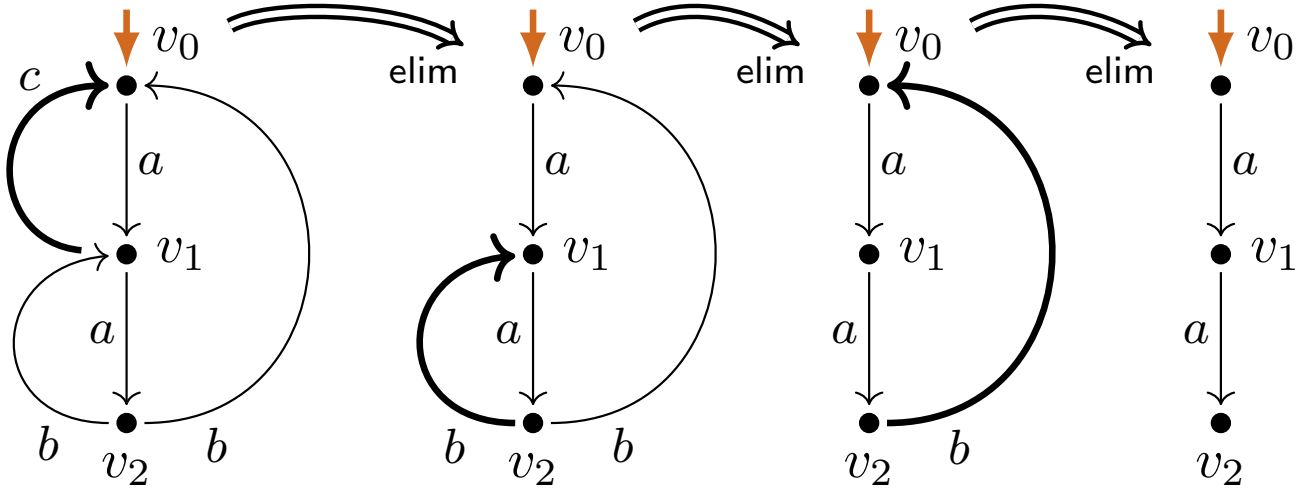
LLEE-witness and LLEE-graph

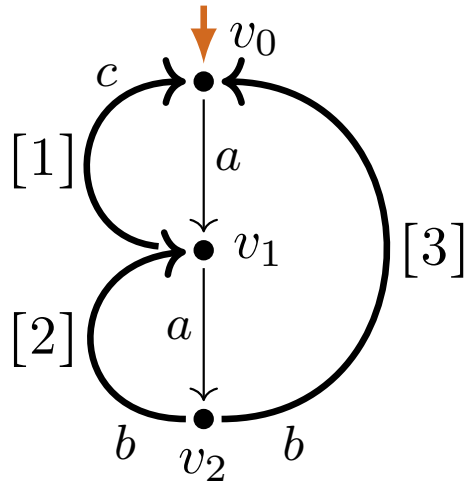


layered LEE

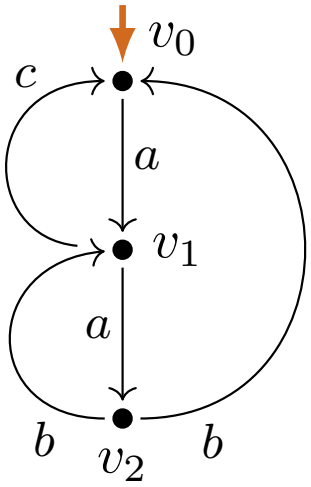
LLEE



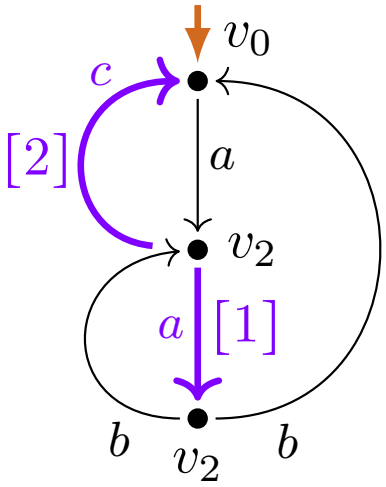




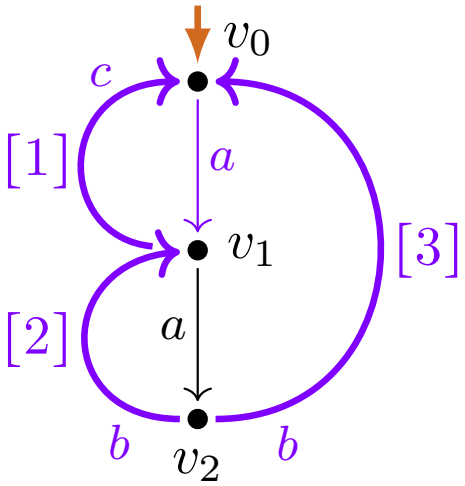
LLEE-witness and LLEE-graph



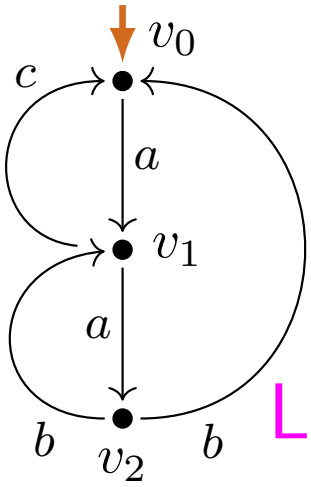
LLEE-witness



LLEE-witness

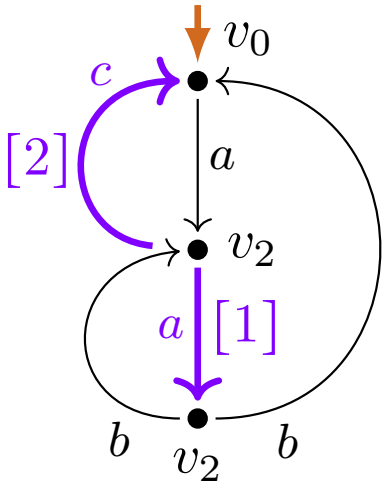


LLEE-witness and LLEE-graph

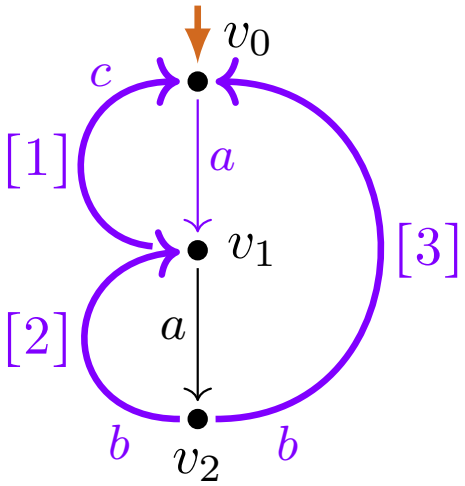


LLEE-graph

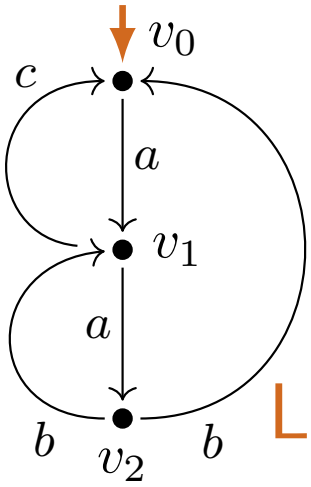
LLEE-witness



LLEE-witness

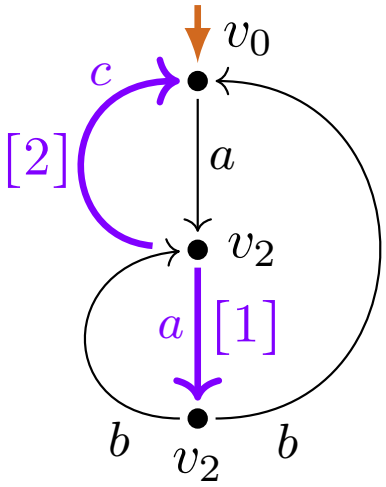


LLEE-witness and LLEE-graph

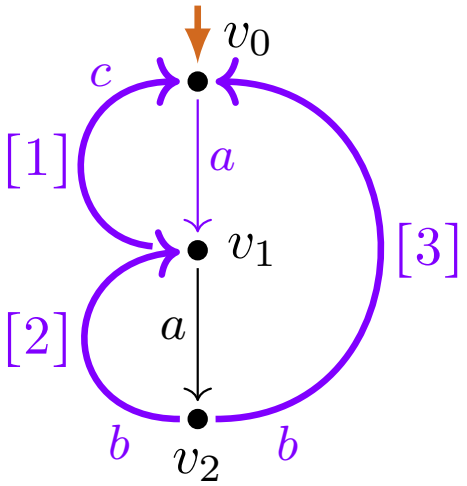


LLEE-graph

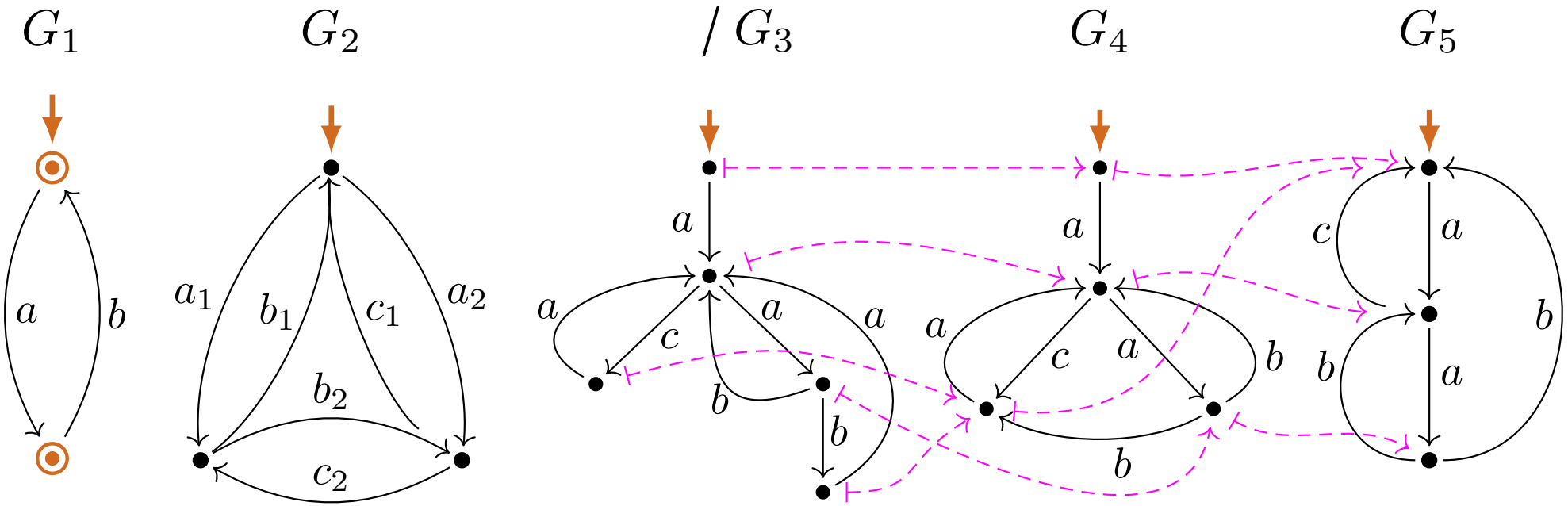
layered LLEE-witness



layered LLEE-witness



LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



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

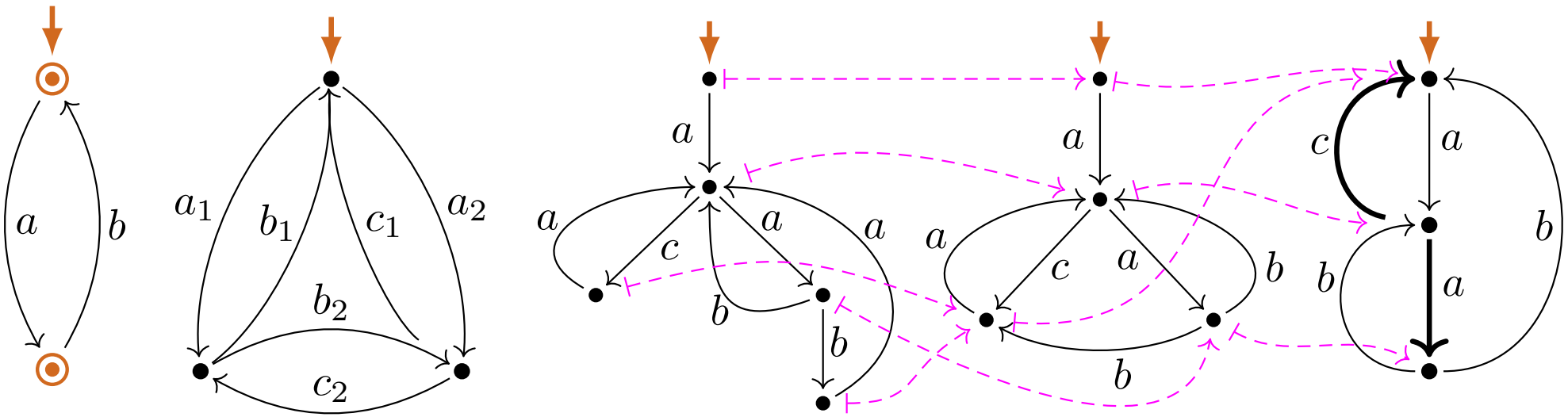
P-expressible ?
$[\![\cdot]\!]_P$-expressible

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

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

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)

G_1 G_2 / G_3 G_4 G_5



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

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

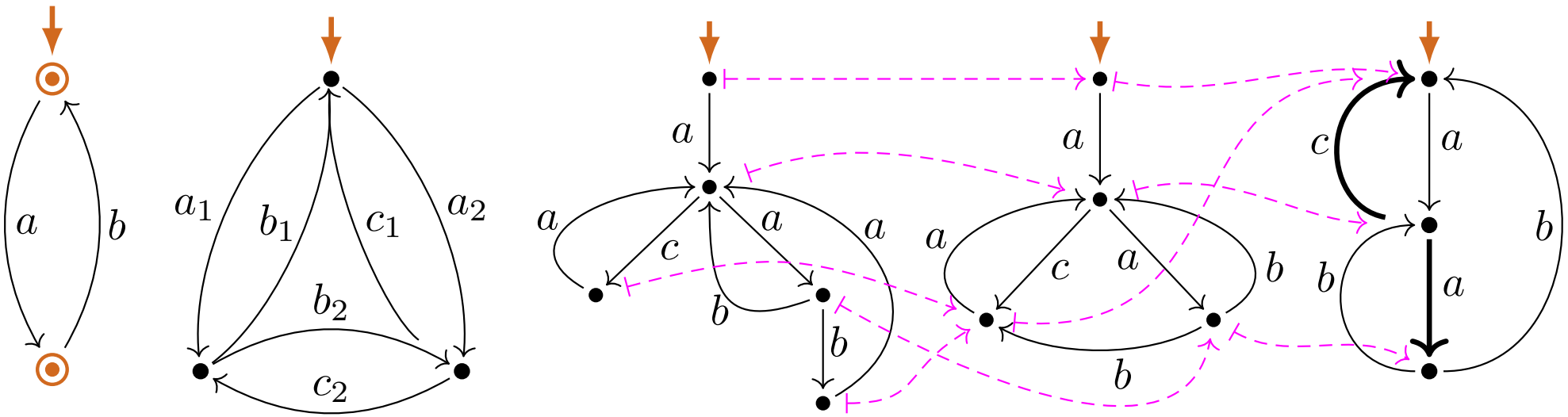
P-expressible
$[\![\cdot]\!]_P$-expressible

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

LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)

G_1 G_2 / G_3 G_4 G_5



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

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

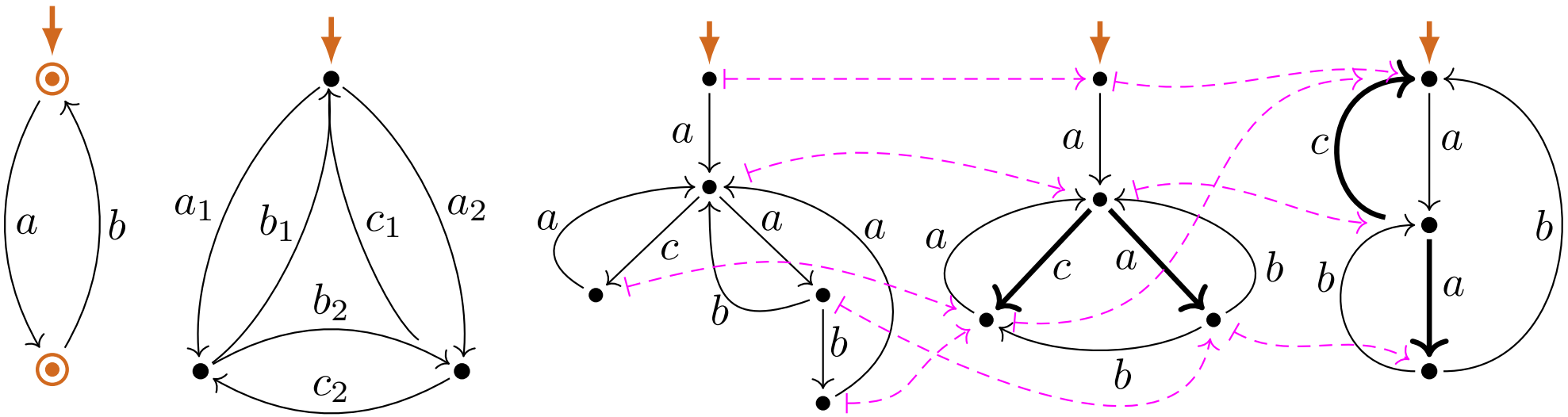
P-expressible
$[\![\cdot]\!]_P$-expressible

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)

G_1 G_2 / G_3 G_4 G_5



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

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

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

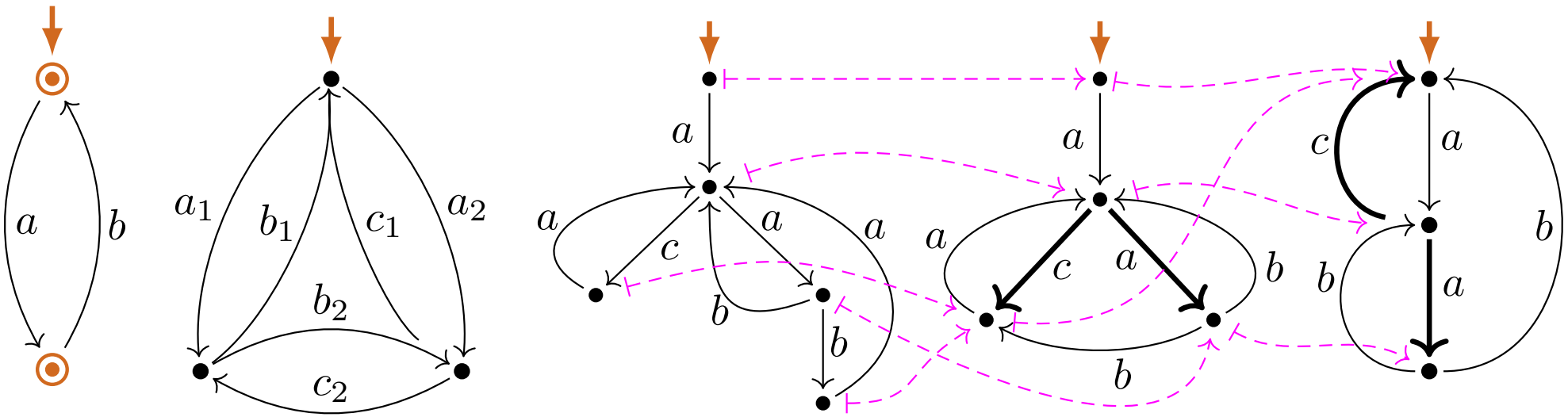
LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)

G_1 G_2 / G_3 G_4 G_5



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

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

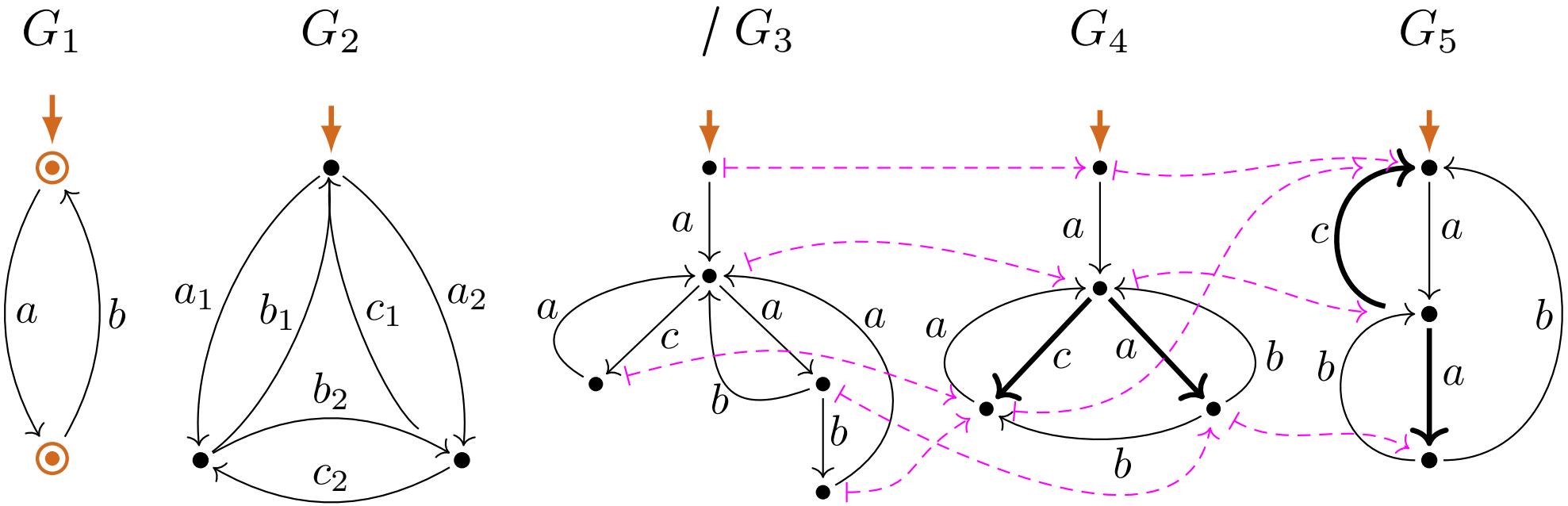
P-expressible
$[\![\cdot]\!]_P$-expressible

LLEE
1-LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



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

not LLEE

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

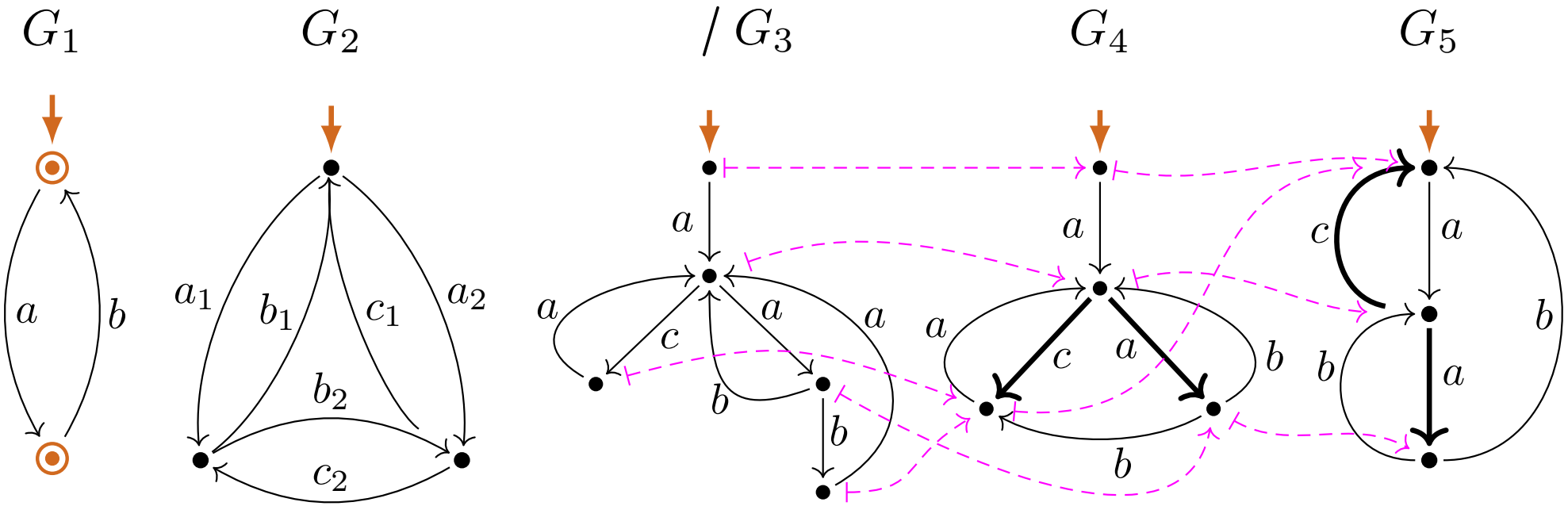
P-expressible
$[\![\cdot]\!]_P$-expressible

LLEE
1-LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



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

not LLEE
not 1-LLEE

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

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

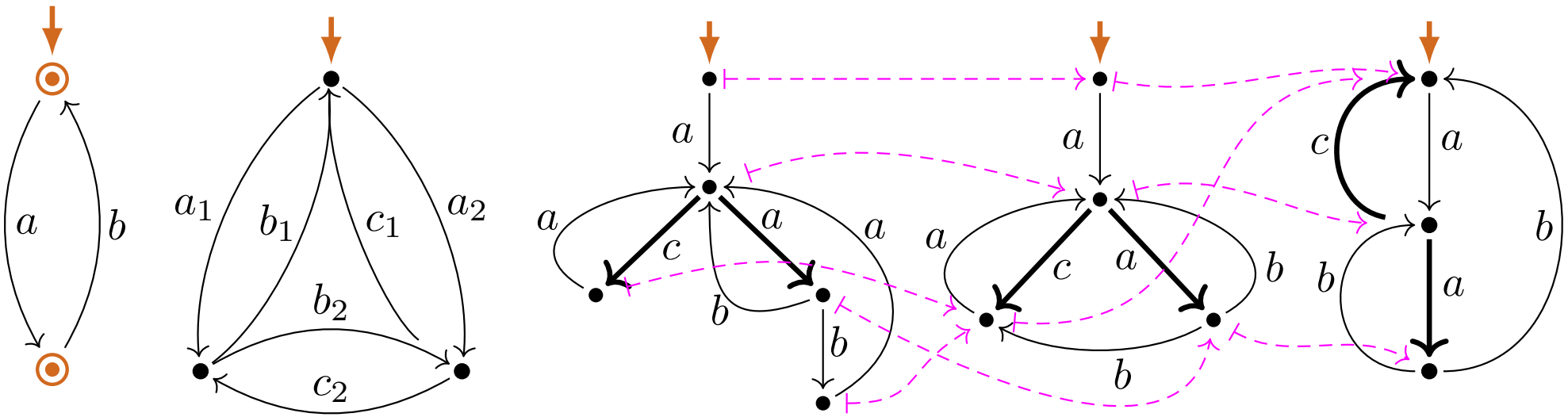
LLEE
1-LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)

G_1 G_2 / G_3 G_4 G_5



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

not LLEE
not 1-LLEE

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

LLEE

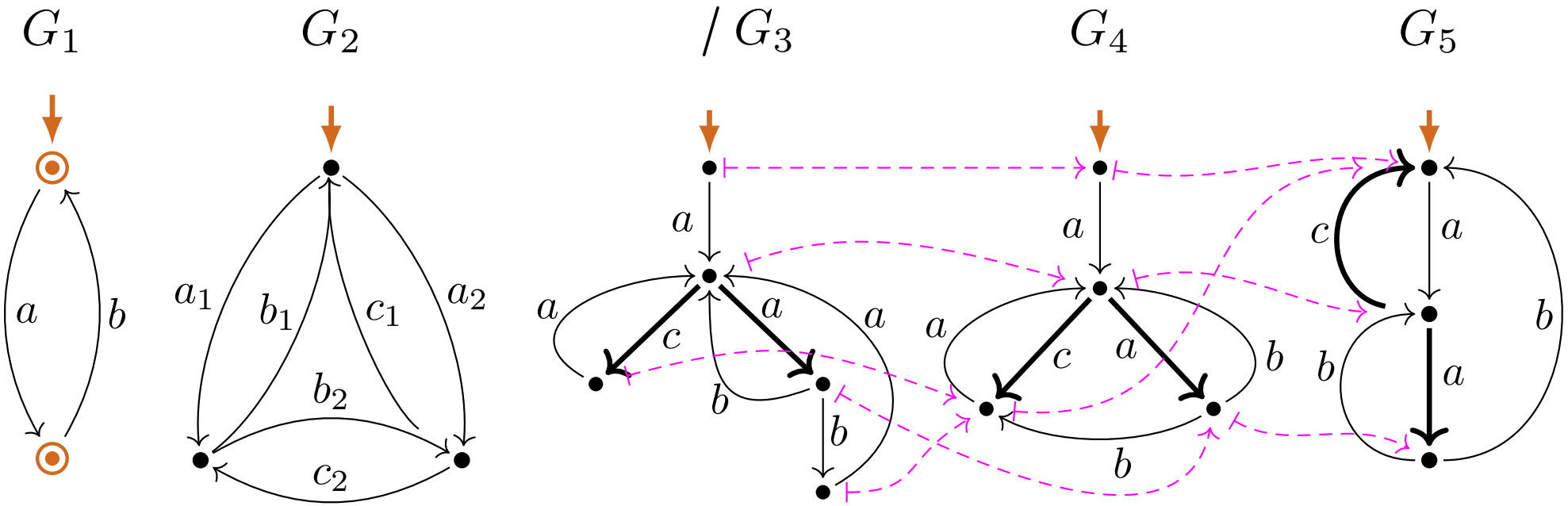
P-expressible
$[\![\cdot]\!]_P$-expressible

LLEE
1-LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



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

not LLEE
not 1-LLEE

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

LLEE
1-LLEE

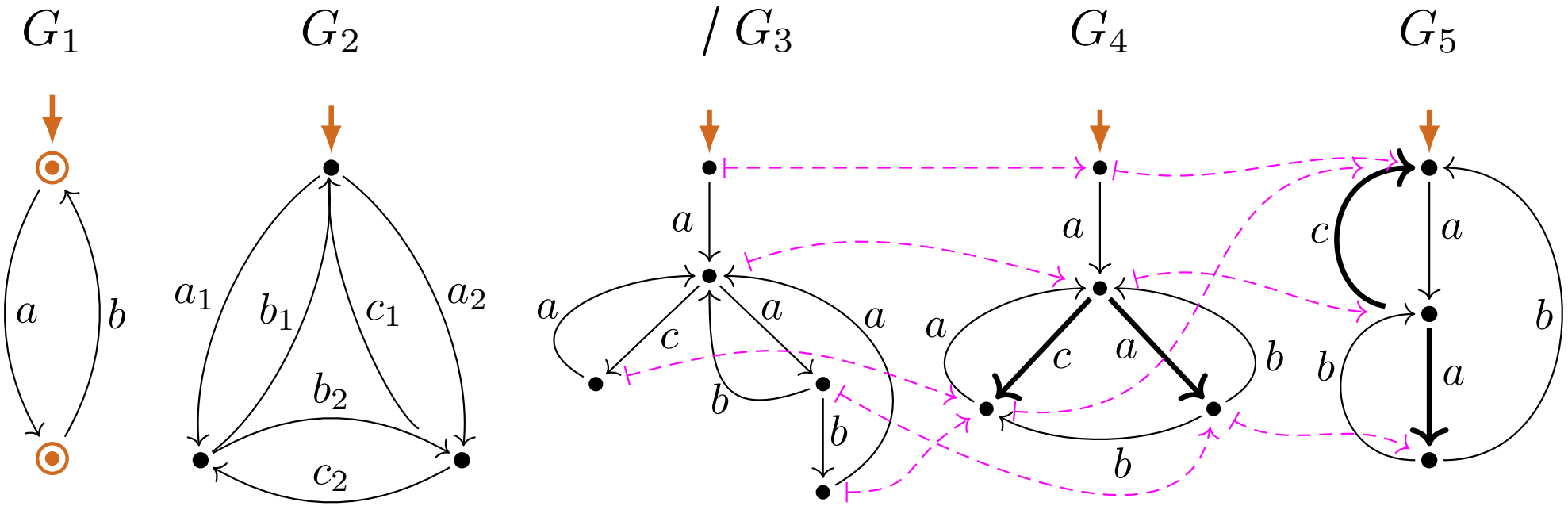
P-expressible
$[\![\cdot]\!]_P$-expressible

LLEE
1-LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



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

not LLEE
not 1-LLEE

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

LLEE
1-LLEE

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

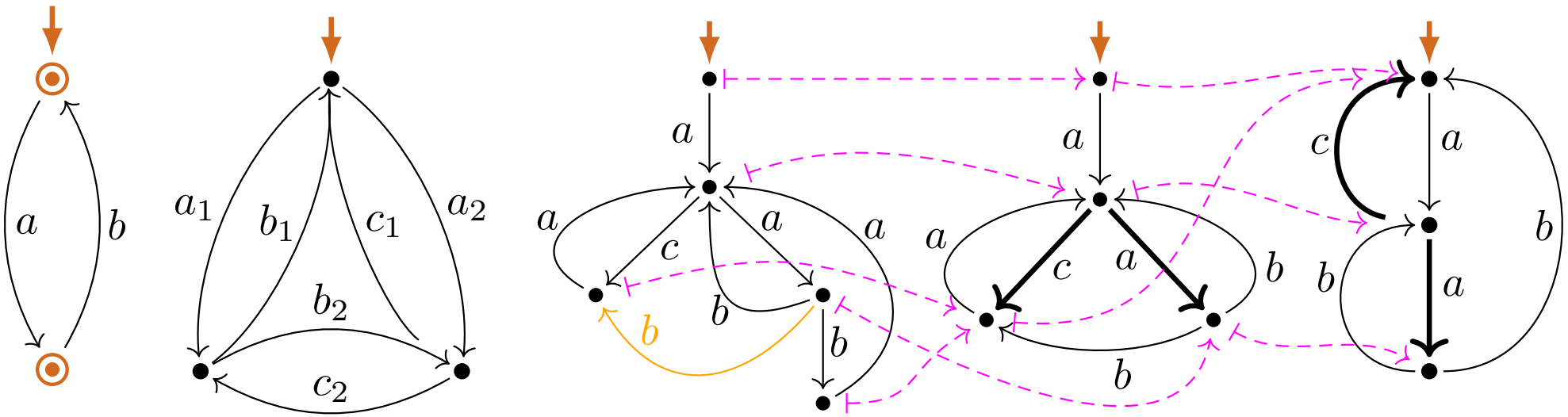
LLEE
1-LLEE

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

LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)

G_1 G_2 G_3^+ / G_3 G_4 G_5



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

not LLEE
not 1-LLEE

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

LLEE
1-LLEE

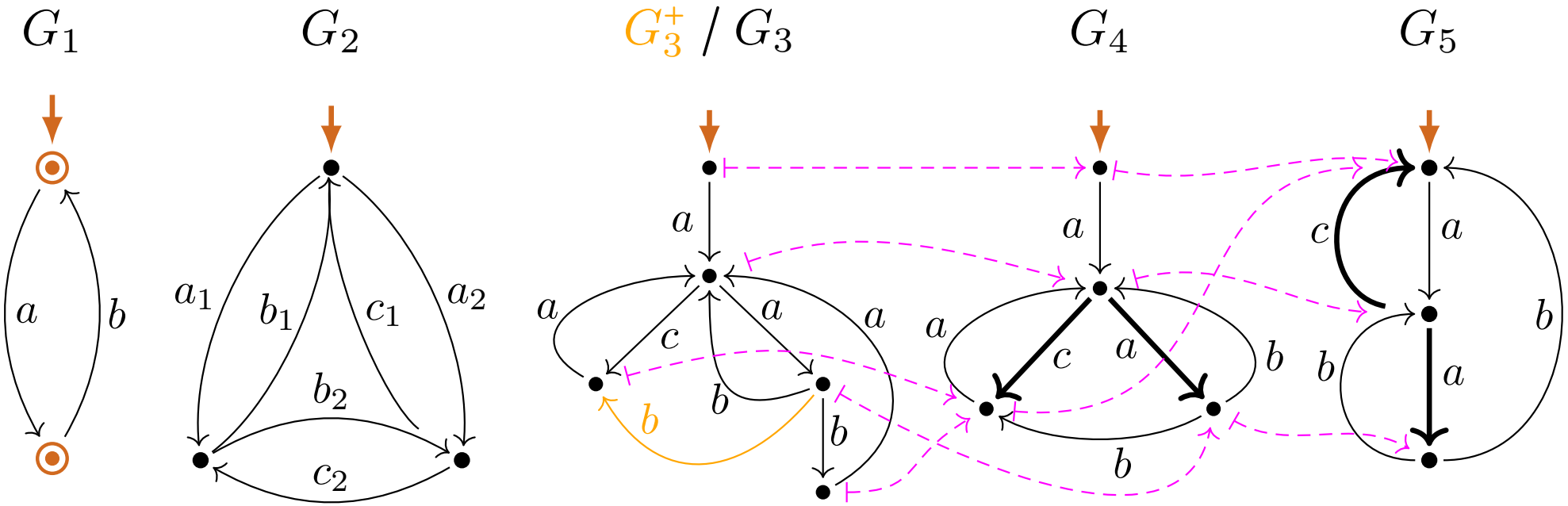
P-expressible
$[\![\cdot]\!]_P$-expressible

LLEE
1-LLEE

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

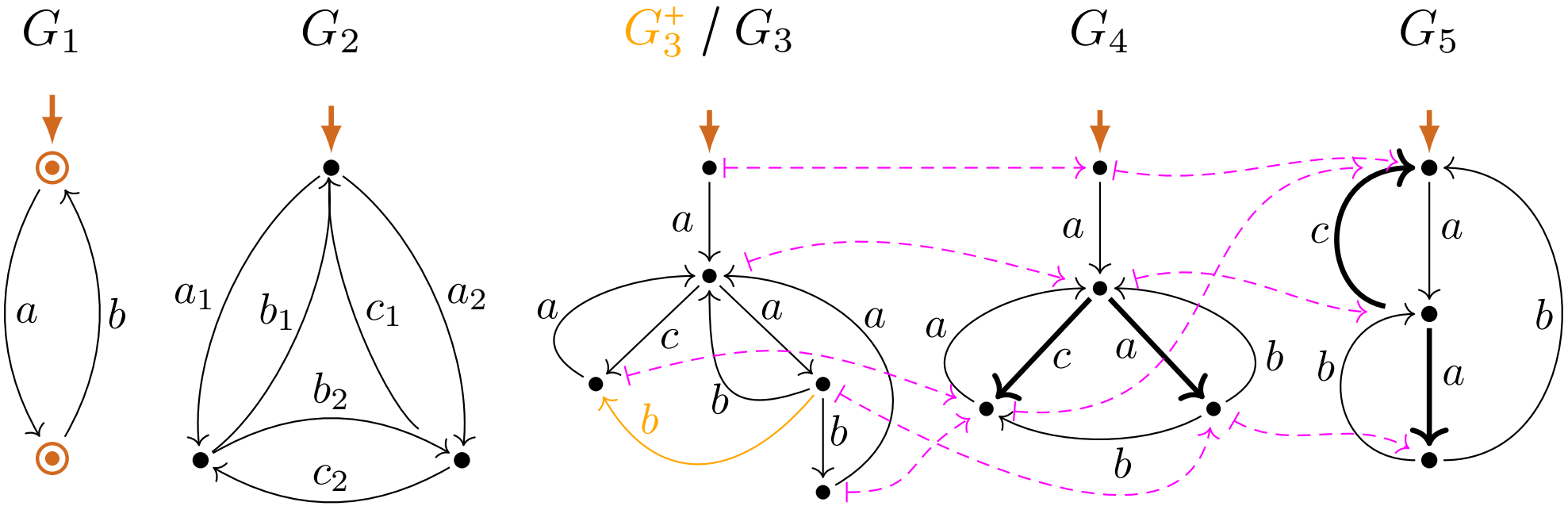
LLEE
1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



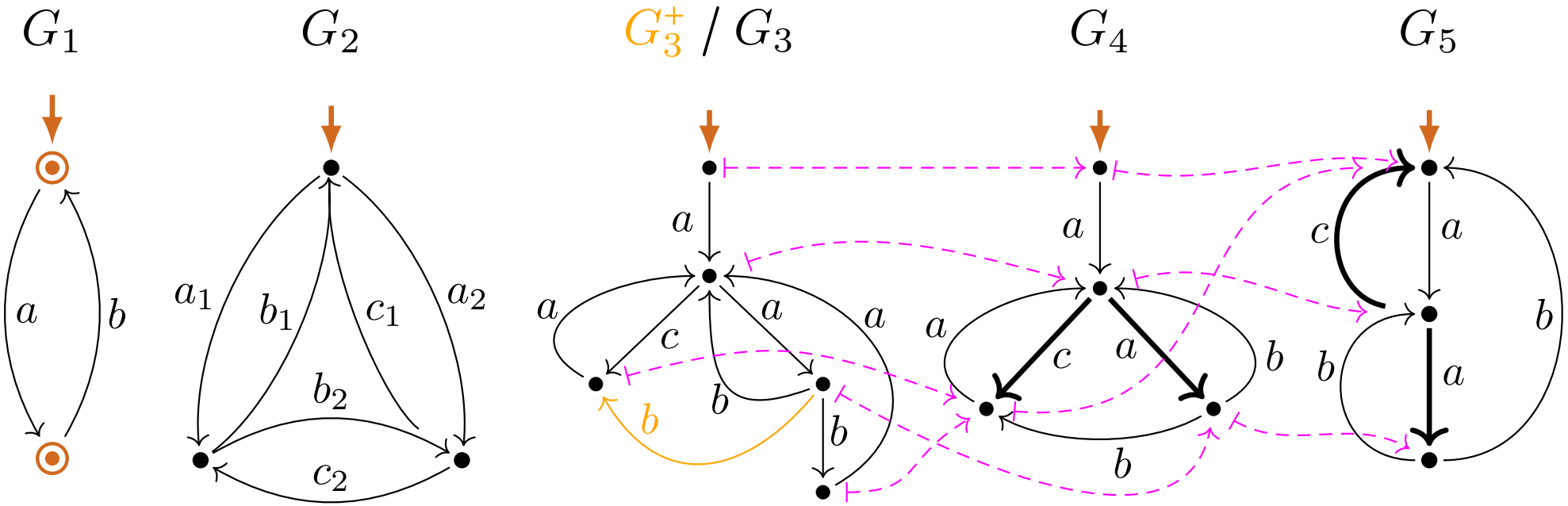
G_1	G_2	G_3^+ / G_3	G_4	G_5
not P-expressible		P-expressible **?**	P-expressible	P-expressible **?**
not $[\![\cdot]\!]_P$-expressible		$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible
not LLEE		**not** LLEE / LLEE	LLEE	LLEE
not 1-LLEE		1-LLEE	1-LLEE	1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



G_1	G_2	G_3^+ / G_3	G_4	G_5
not P-expressible		P-expressible **?**	P-expressible	P-expressible **?**
not $[\![\cdot]\!]_P$-expressible		$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible
not LLEE		**not** LLEE / LLEE	LLEE	LLEE
not 1-LLEE		**not** 1-LLEE/ 1-LLEE	1-LLEE	1-LLEE

LLEE versus P-/$[\![\cdot]\!]_P$-expressibility (examples)



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

not LLEE
not 1-LLEE

$P(\cdot)$-expressible **?**
$[\![\cdot]\!]_P$-expressible

not LLEE / LLEE
not 1-LLEE/ 1-LLEE

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

LLEE
1-LLEE

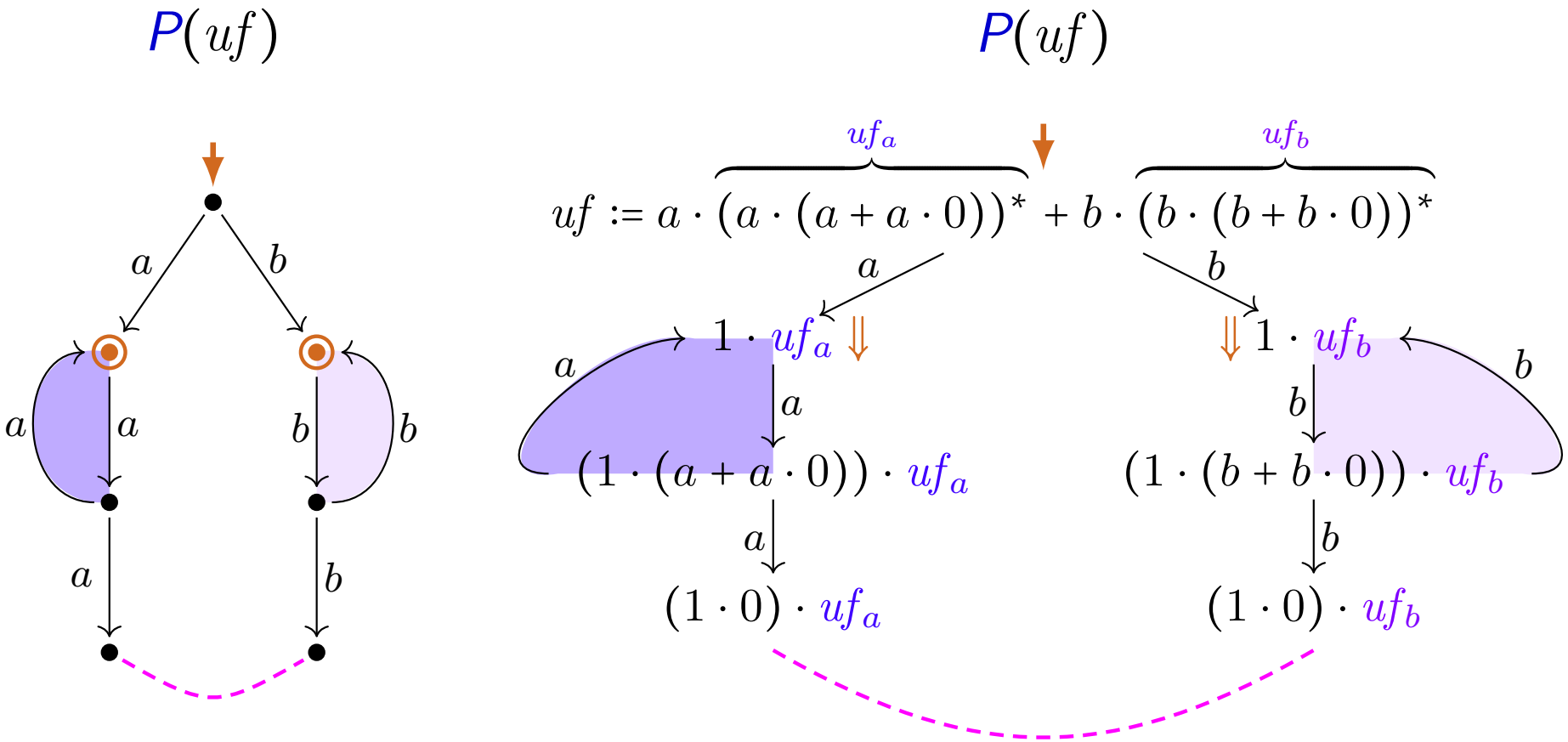
P-expressible **?**
$[\![\cdot]\!]_P$-expressible

LLEE
1-LLEE

TERMGRAPH '24

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

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



Process interpretation P (formally)

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

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

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

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

Responsible rule

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

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

Problem: if e_1' is not normed, $e_1' \cdot e_2$ has redundant subexpression e_2

Compact process interpretation $P^\bullet$

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

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

Compact process interpretation $P^\bullet$

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

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

Compact process interpretation $P^\bullet$

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

$$\pi(e) := \begin{cases} e_1 \cdot e_2 & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is normed,} \\ \pi(e_1) & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is not normed,} \\ e & \text{otherwise.} \end{cases}$$

$$\frac{e_1 \xrightarrow{a} ed}{e_1 \cdot e_2 \xrightarrow{a} \pi(ed \cdot e_2)}$$

Compact process interpretation $P^\bullet$

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

$$\pi(e) := \begin{cases} e_1 \cdot e_2 & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is normed,} \\ \pi(e_1) & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is not normed,} \\ e & \text{otherwise.} \end{cases}$$

$$\frac{e_1 \xrightarrow{a} ed}{e_1 \cdot e_2 \xrightarrow{a} \pi(ed \cdot e_2)} \qquad \frac{e \xrightarrow{a} ed}{e^* \xrightarrow{a} \pi(ed \cdot e^*)}$$

Compact process interpretation $P^\bullet$

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

$$\pi(e) := \begin{cases} e_1 \cdot e_2 & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is normed,} \\ \pi(e_1) & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is not normed,} \\ e & \text{otherwise.} \end{cases}$$

$$\frac{e_1 \xrightarrow{a} ed}{e_1 \cdot e_2 \xrightarrow{a} \pi(ed \cdot e_2)} \qquad \frac{e \xrightarrow{a} ed}{e^* \xrightarrow{a} \pi(ed \cdot e^*)}$$

Definition

The compact process (graph) interpretation $P^\bullet(e)$ of a reg. expr's e:
$P^\bullet(e)$:= labeled transition graph generated by $\pi(e)$ by derivations in $\mathcal{T}^\bullet$.

Compact process interpretation $P^\bullet$

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

$$\pi(e) := \begin{cases} e_1 \cdot e_2 & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is normed,} \\ \pi(e_1) & \text{if } e = e_1 \cdot e_2 \text{ and } e_1 \text{ is not normed,} \\ e & \text{otherwise.} \end{cases}$$

$$\frac{e_1 \xrightarrow{a} ed}{e_1 \cdot e_2 \xrightarrow{a} \pi(ed \cdot e_2)} \qquad \frac{e \xrightarrow{a} ed}{e^* \xrightarrow{a} \pi(ed \cdot e^*)}$$

Definition

The compact process (graph) interpretation $P^\bullet(e)$ of a reg. expr's e:
$P^\bullet(e)$:= labeled transition graph generated by $\pi(e)$ by derivations in $\mathcal{T}^\bullet$.

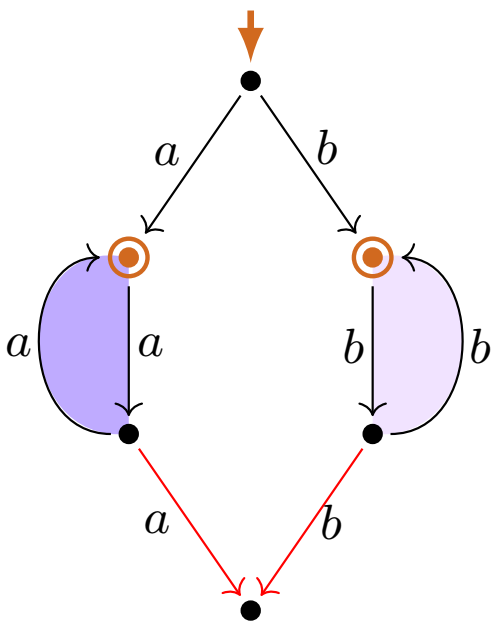
Lemma ($P^\bullet$ increases sharing; $P^\bullet$, P have same bisimulation semantics)

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

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

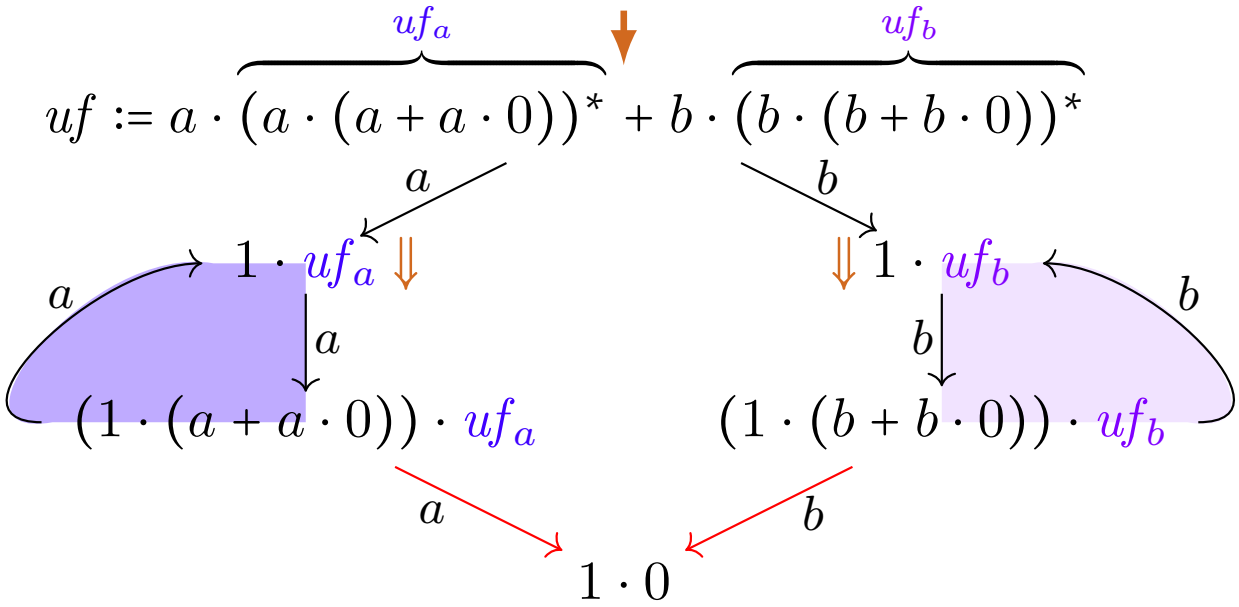
Image of $P^\bullet$ under bisimulation collapse . . .

$P^\bullet(uf)$



$P^\bullet(uf)$

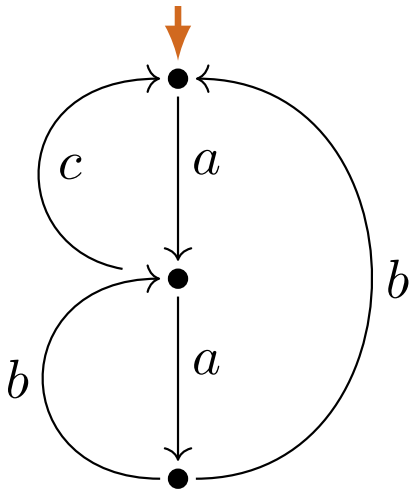
$$uf := \overbrace{a \cdot (a \cdot (a + a \cdot 0))^*}^{uf_a} + \overbrace{b \cdot (b \cdot (b + b \cdot 0))^*}^{uf_b}$$



Refined extraction expression (example)

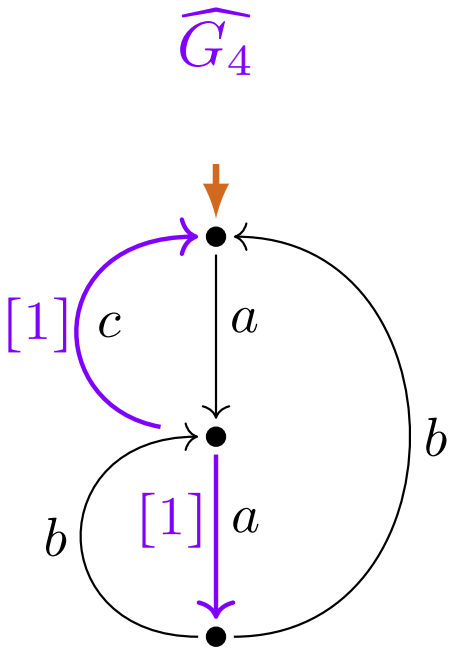
Define readback steps to precisely reverse $P^\bullet$-defined transitions!

G_4



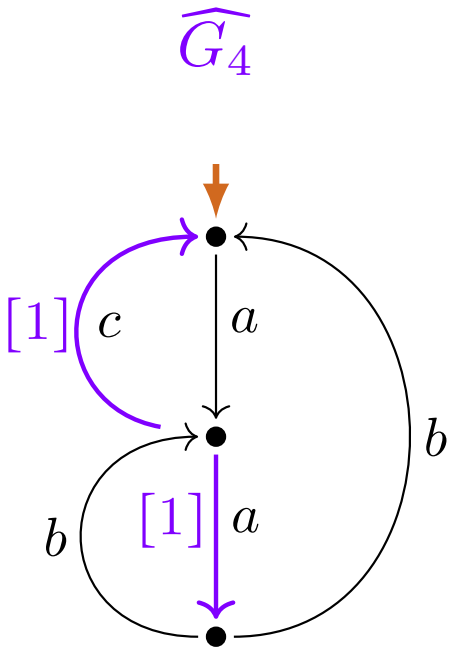
Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!



Refined extraction expression (example)

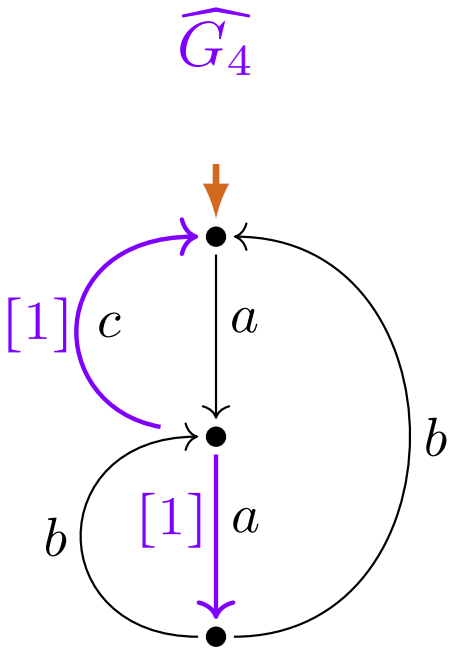
Define readback steps to precisely reverse $P^\bullet$-defined transitions!



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

Refined extraction expression (example)

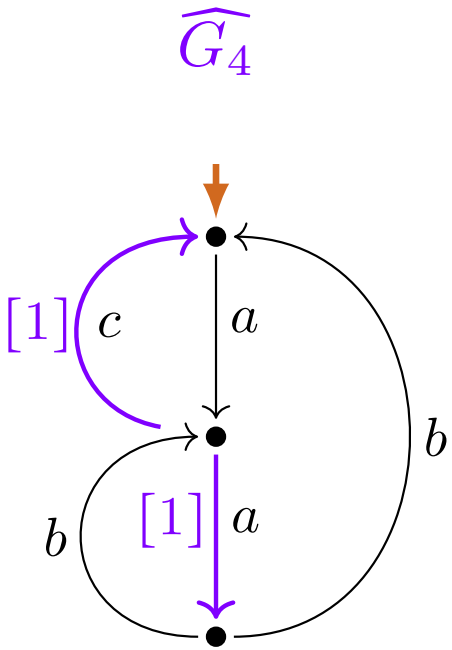
Define readback steps to precisely reverse $P^\bullet$-defined transitions!



$$(1 \cdot (\quad \overset{a}{\vdots} \quad)^*) \cdot 0$$

Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!



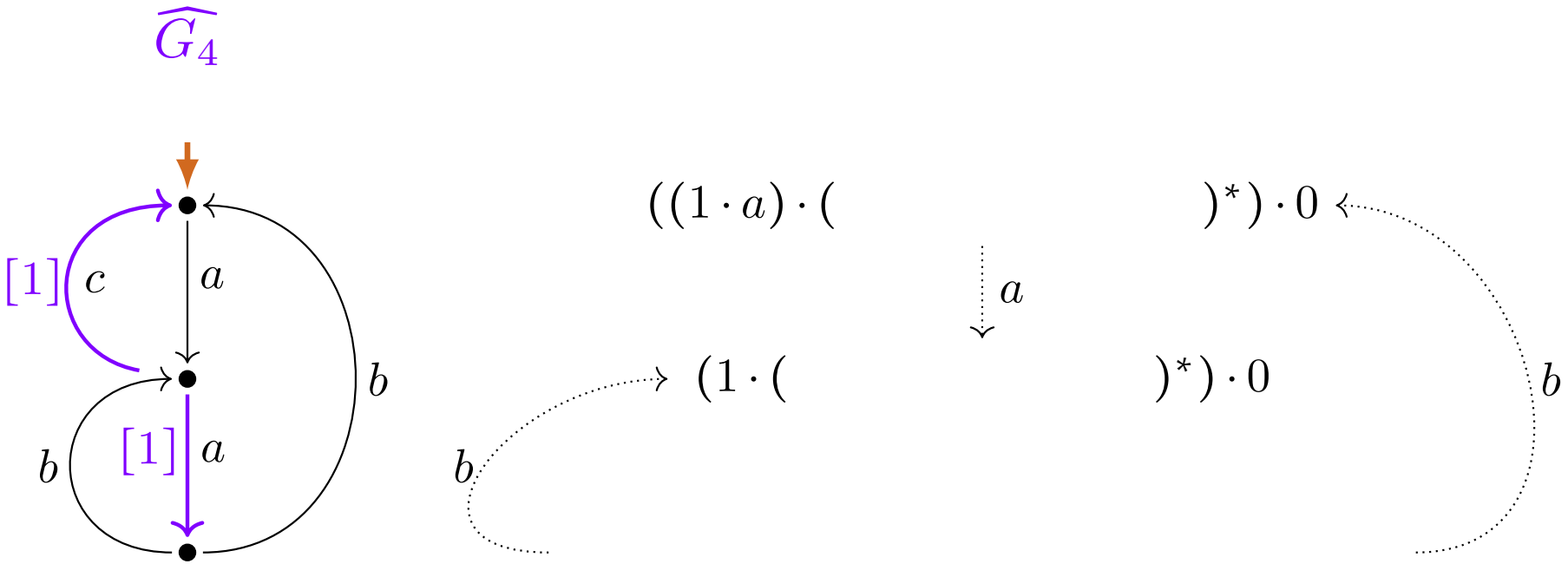
$$((1 \cdot a) \cdot (\qquad)^*) \cdot 0$$

$$\vdots\ a$$

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

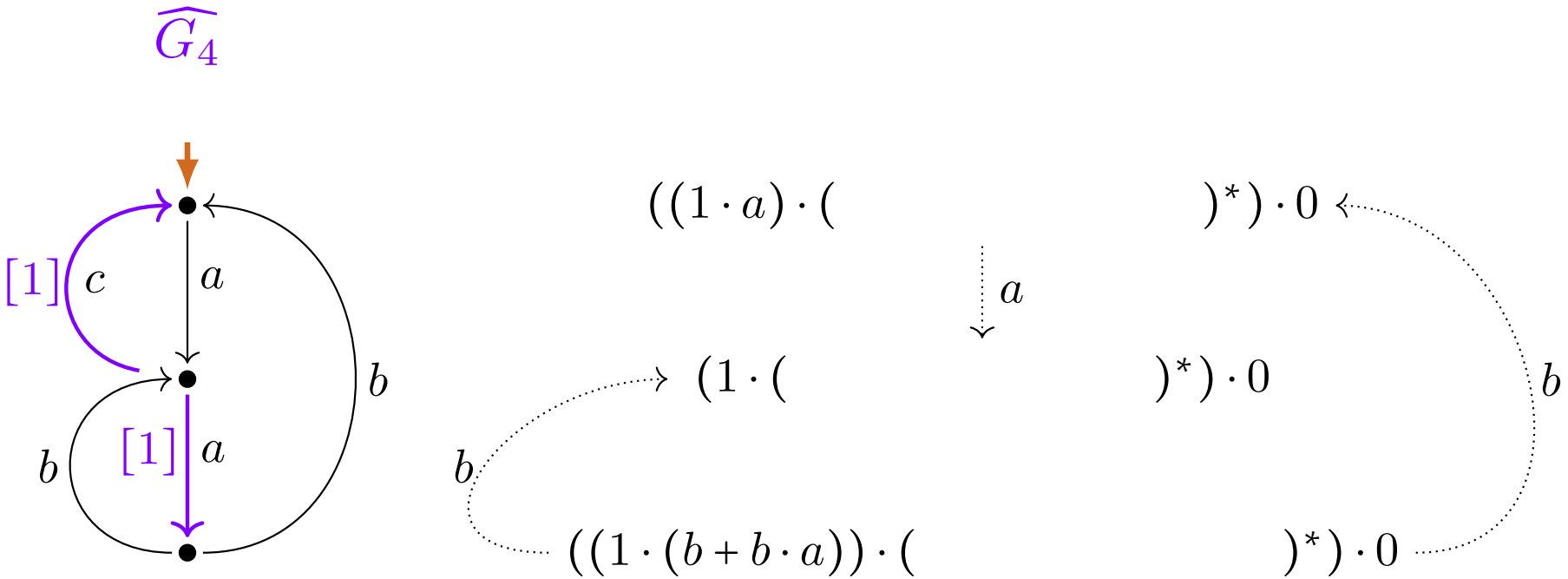
Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!



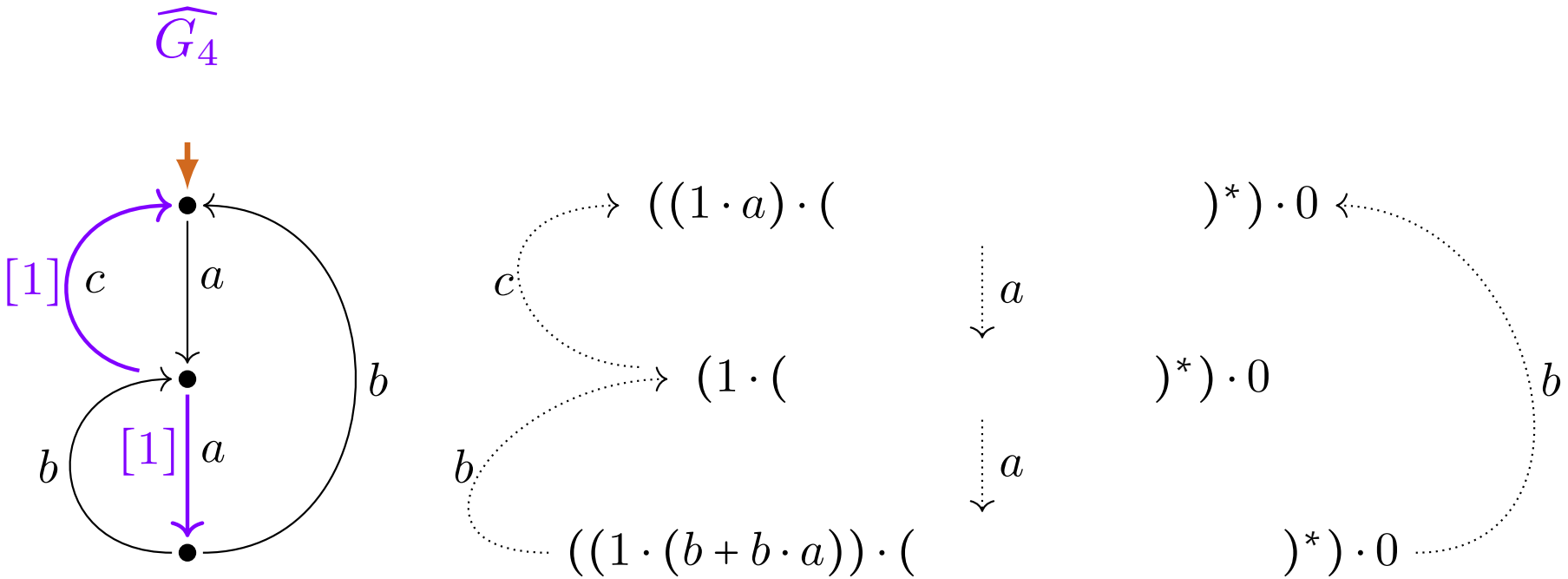
Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!



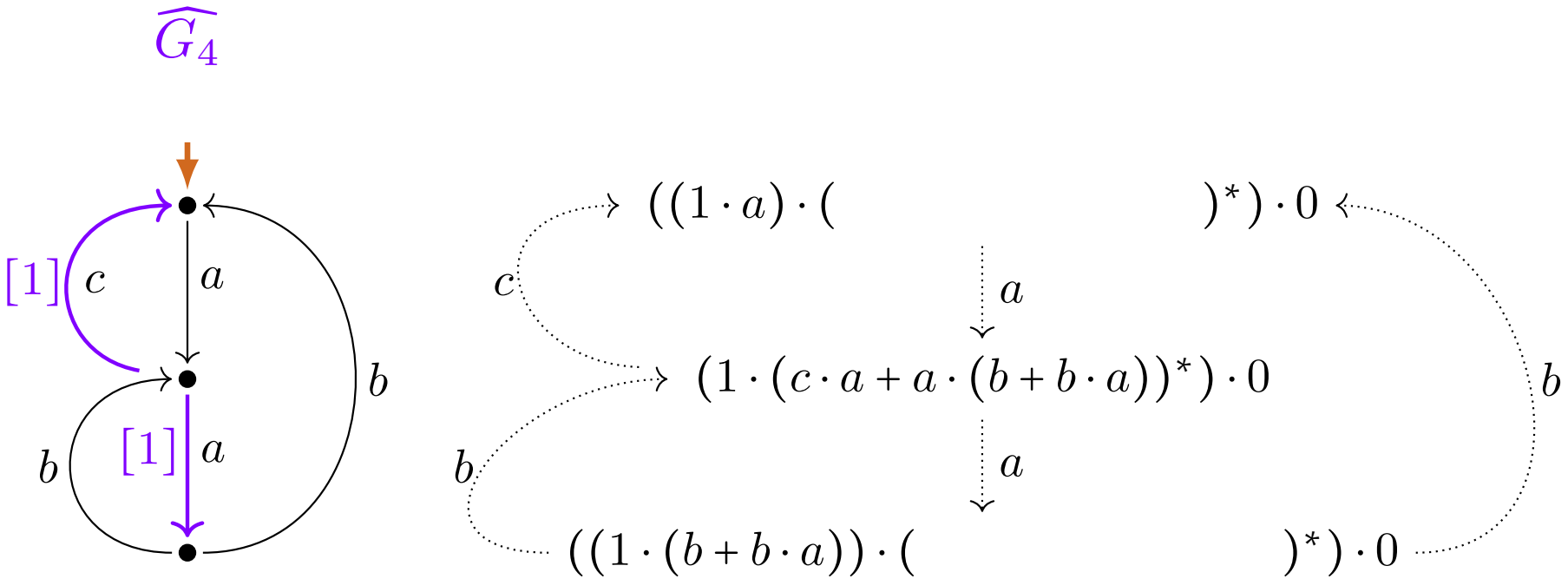
Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!



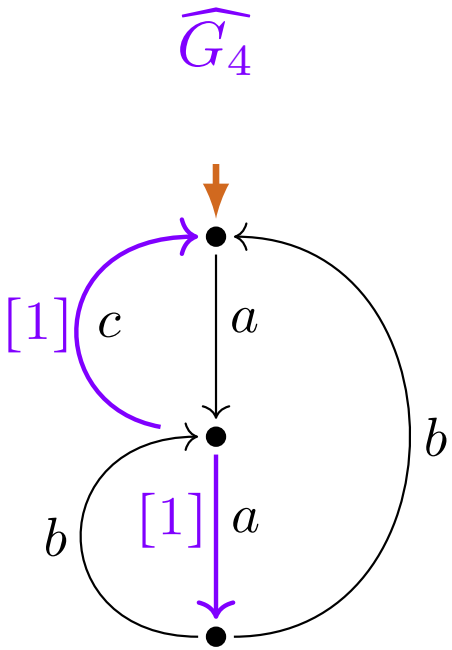
Refined extraction expression (example)

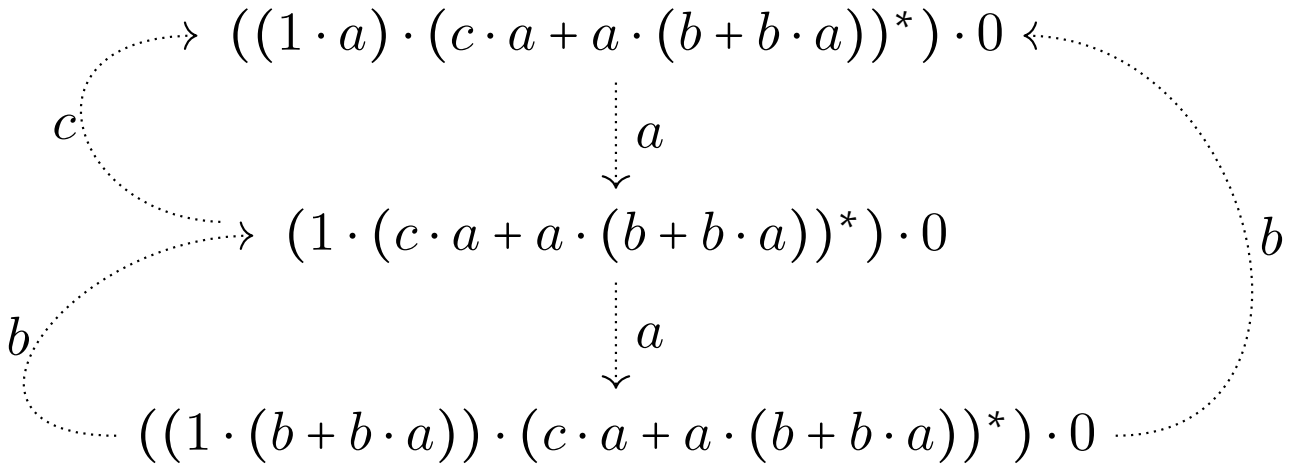
Define readback steps to precisely reverse $P^\bullet$-defined transitions!



Refined extraction expression (example)

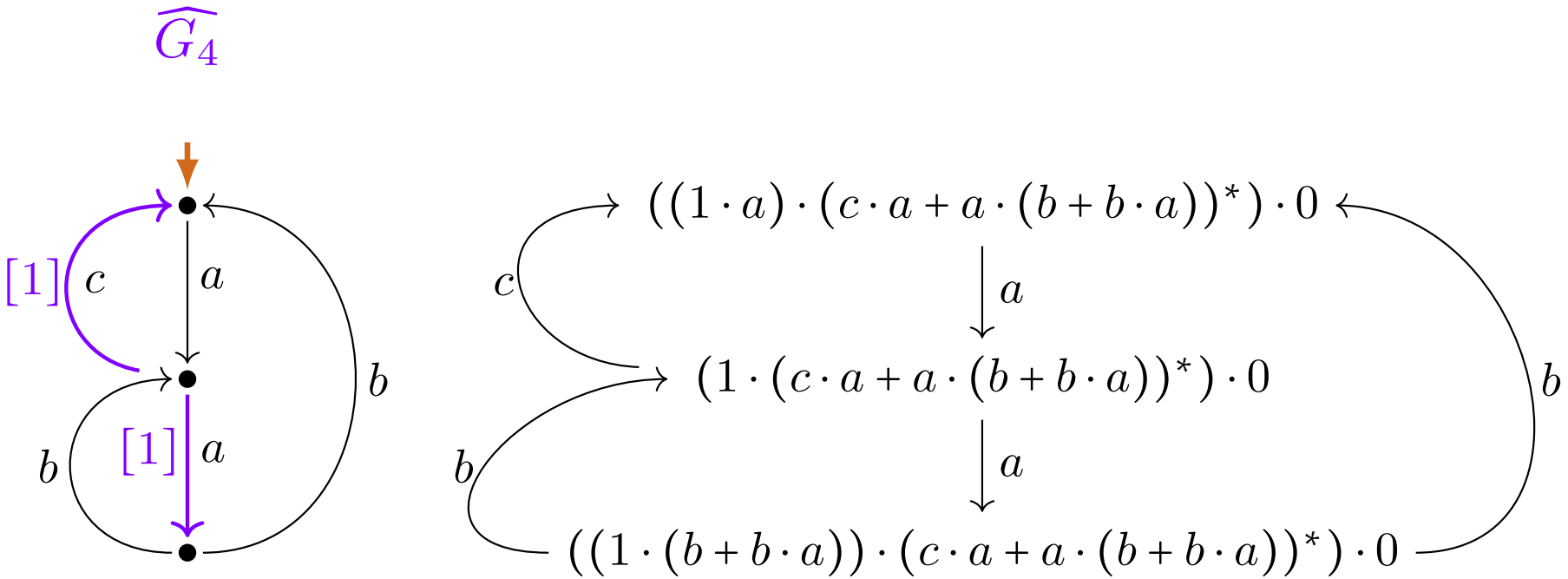
Define readback steps to precisely reverse $P^\bullet$-defined transitions!





Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!

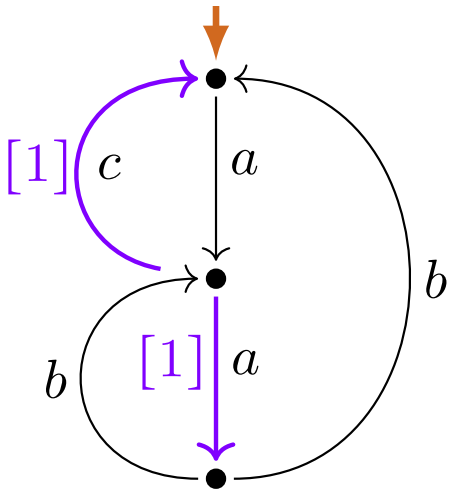


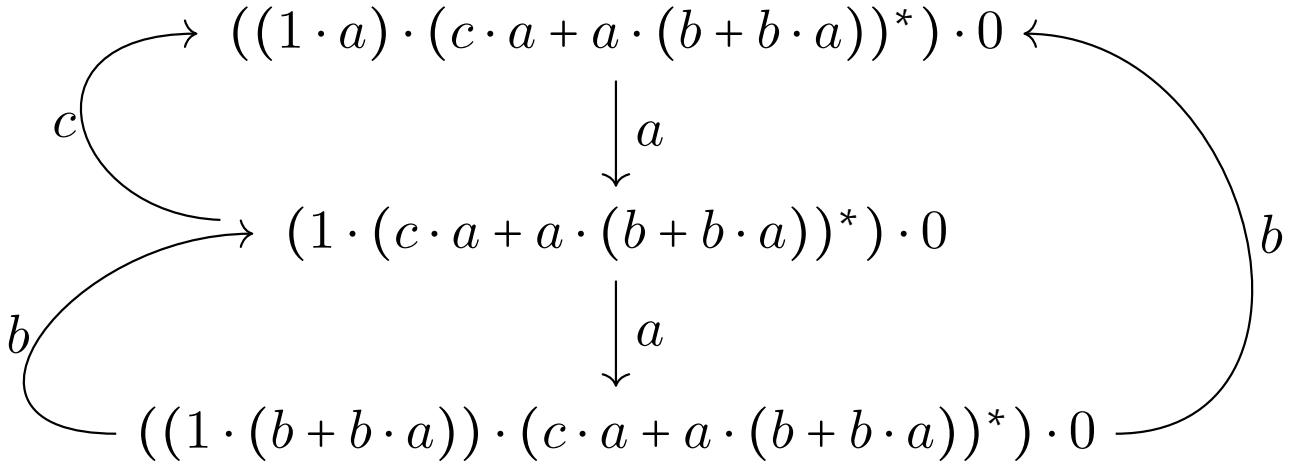
Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!

$\widehat{G_4}$

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

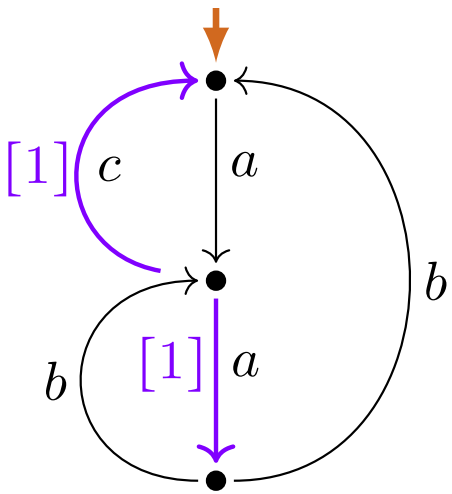




Refined extraction expression (example)

Define readback steps to precisely reverse $P^\bullet$-defined transitions!

$\widehat{G_4}$



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

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

$$\downarrow a$$

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

$$\downarrow a$$

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

Refined extraction expression (example)

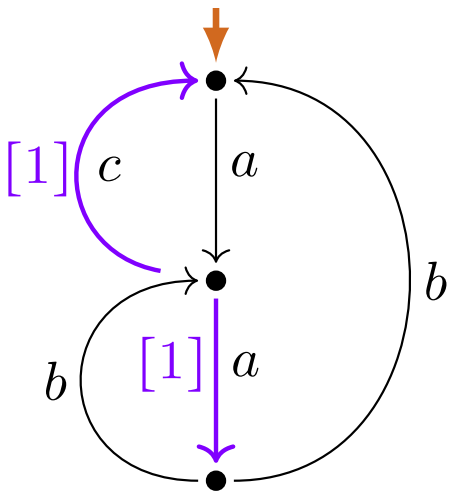
Lemma (*TERMGRAPH '24*)

Process graphs with LLEE are $P^\bullet$-expressible by $(*/\not{1})$ reg. expressions:

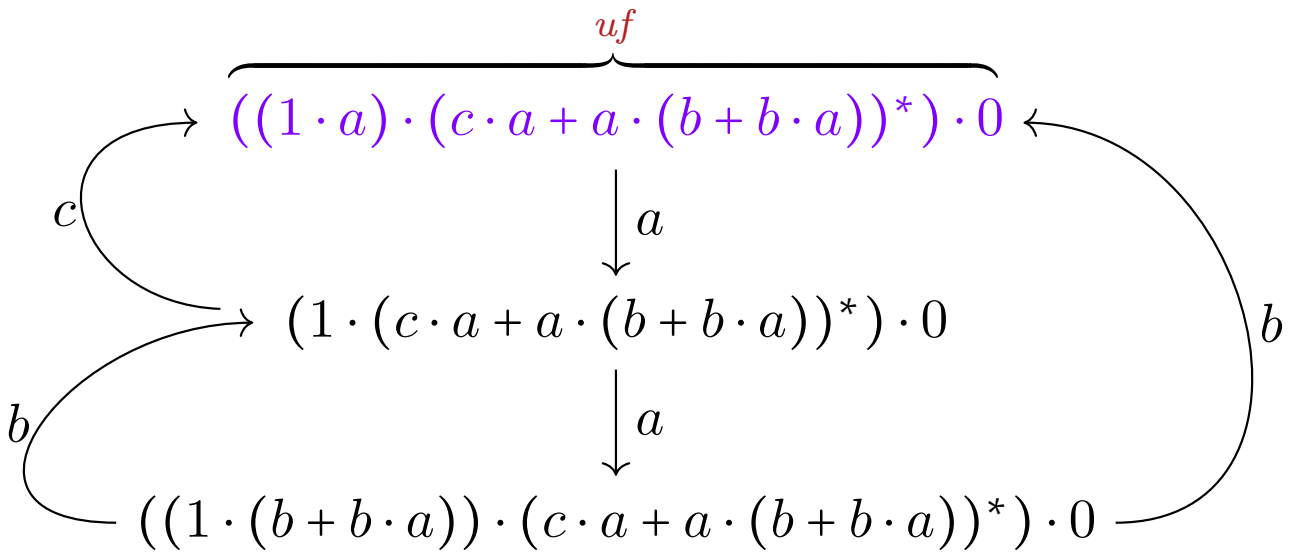
$$\mathsf{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\not{1})} \big(G \twoheadrightarrow P^\bullet(uf) \big).$$

Proof: Define readback steps to precisely reverse $P^\bullet$-defined transitions!

$\widehat{G_4}$



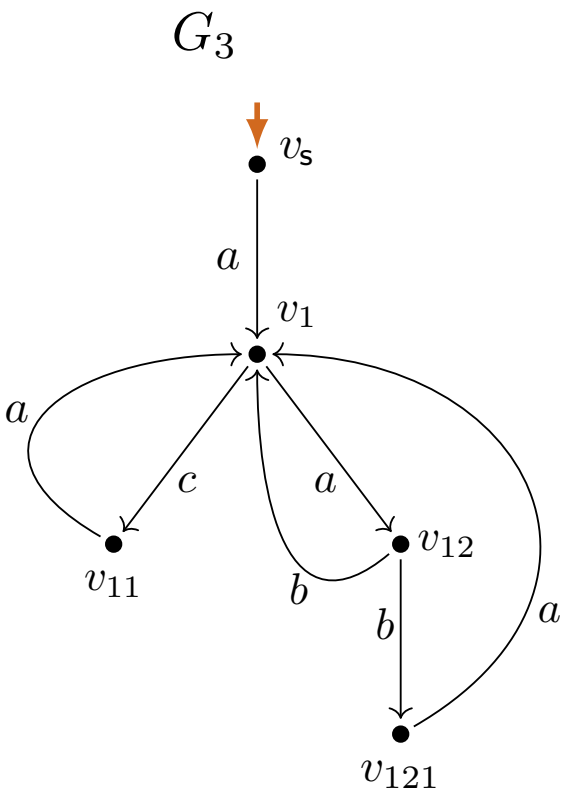
$$P^\bullet(uf) = P(uf) \simeq G_4$$



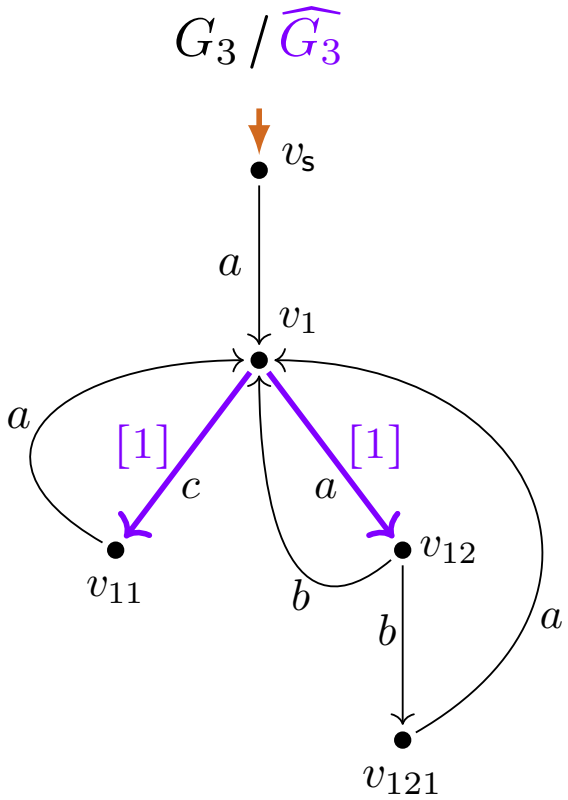
Characterising the image of $P^\bullet$ for $(*/\not{1})$ expressions

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

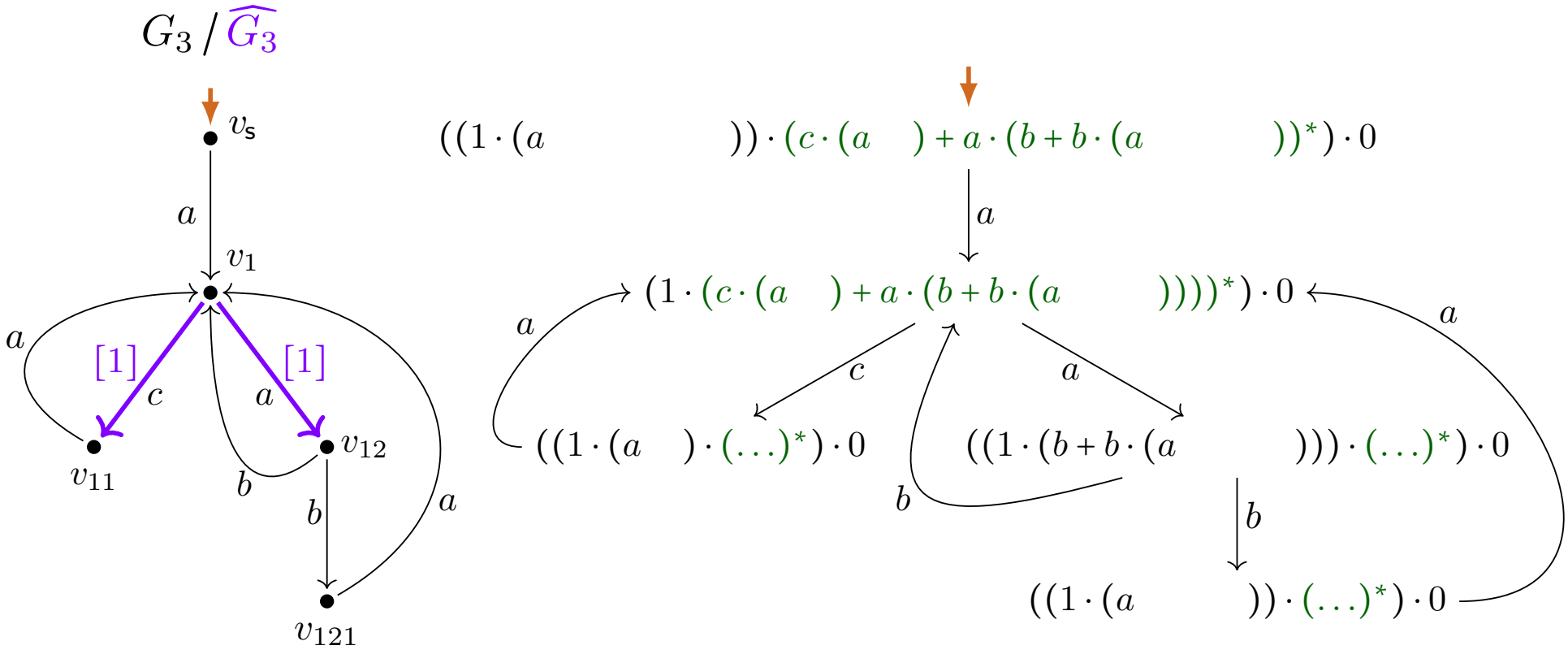
Refined extraction



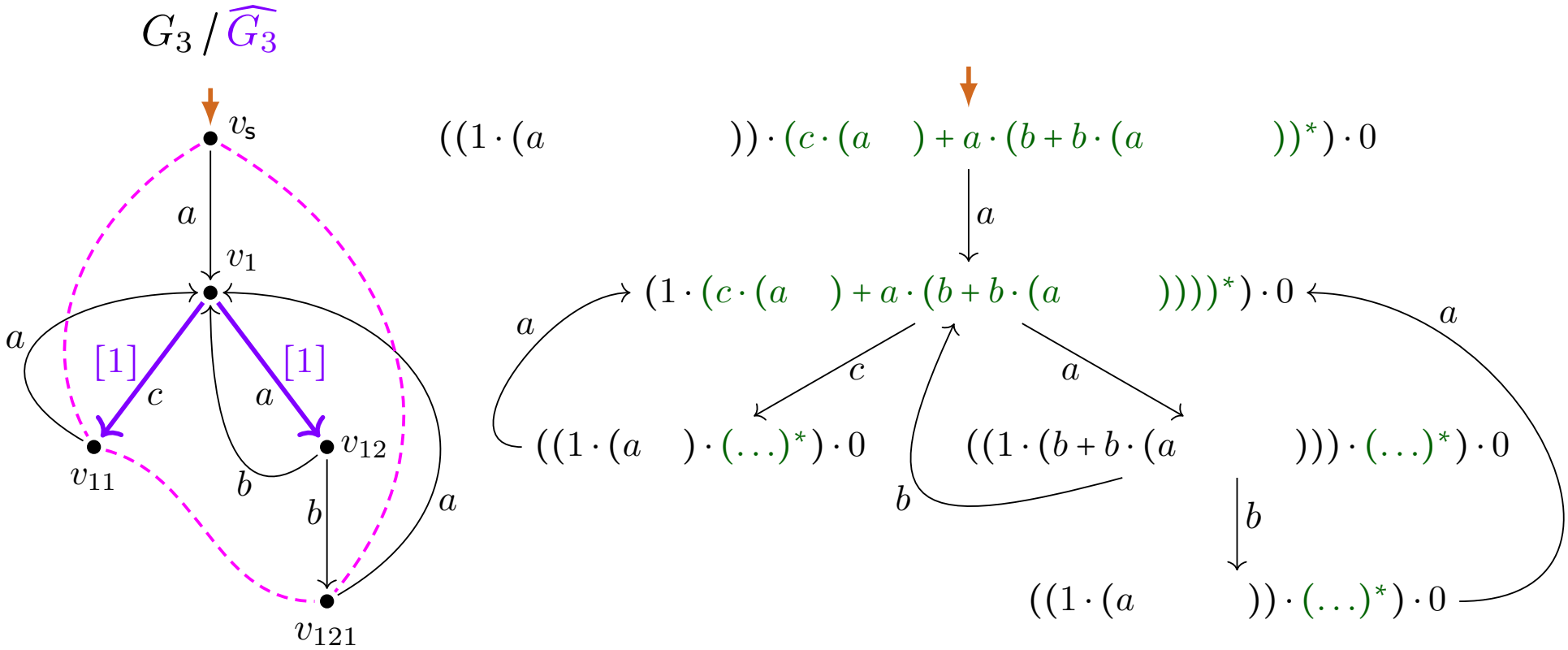
Refined extraction



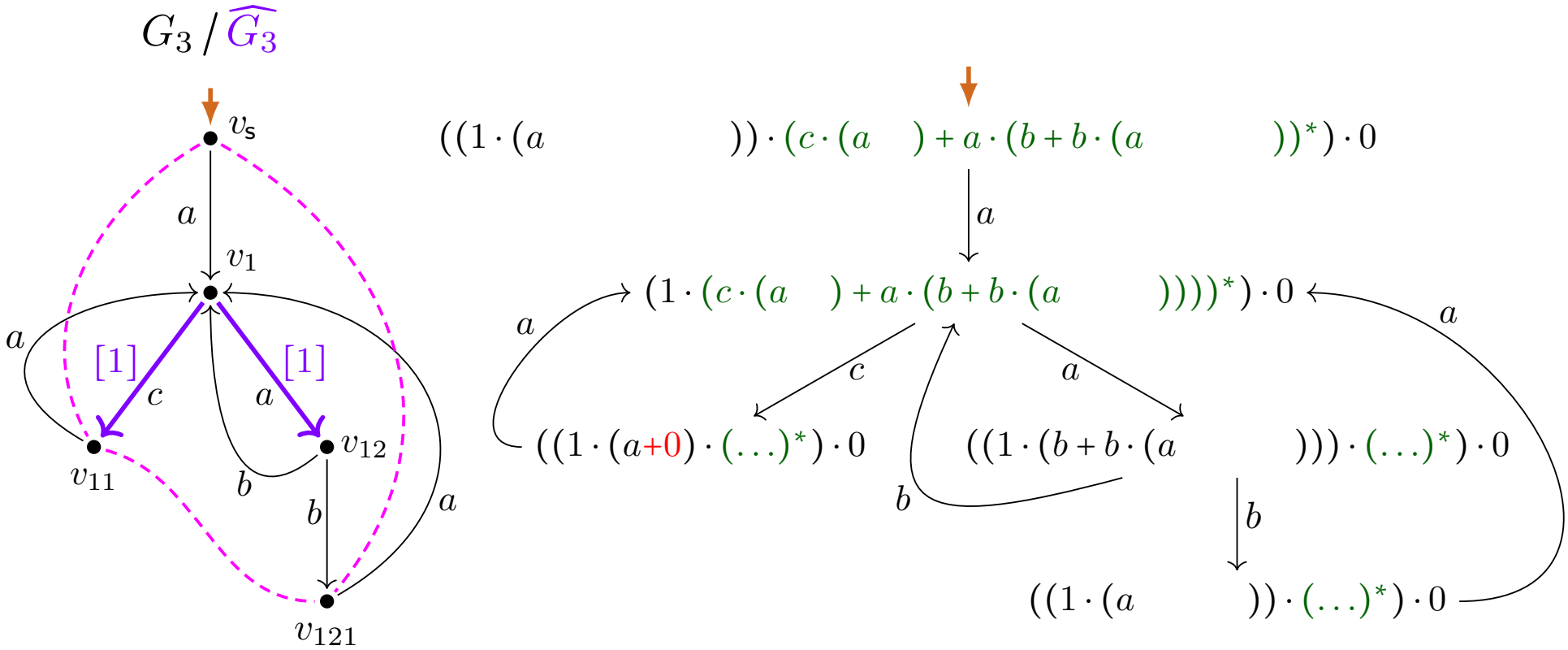
Refined extraction



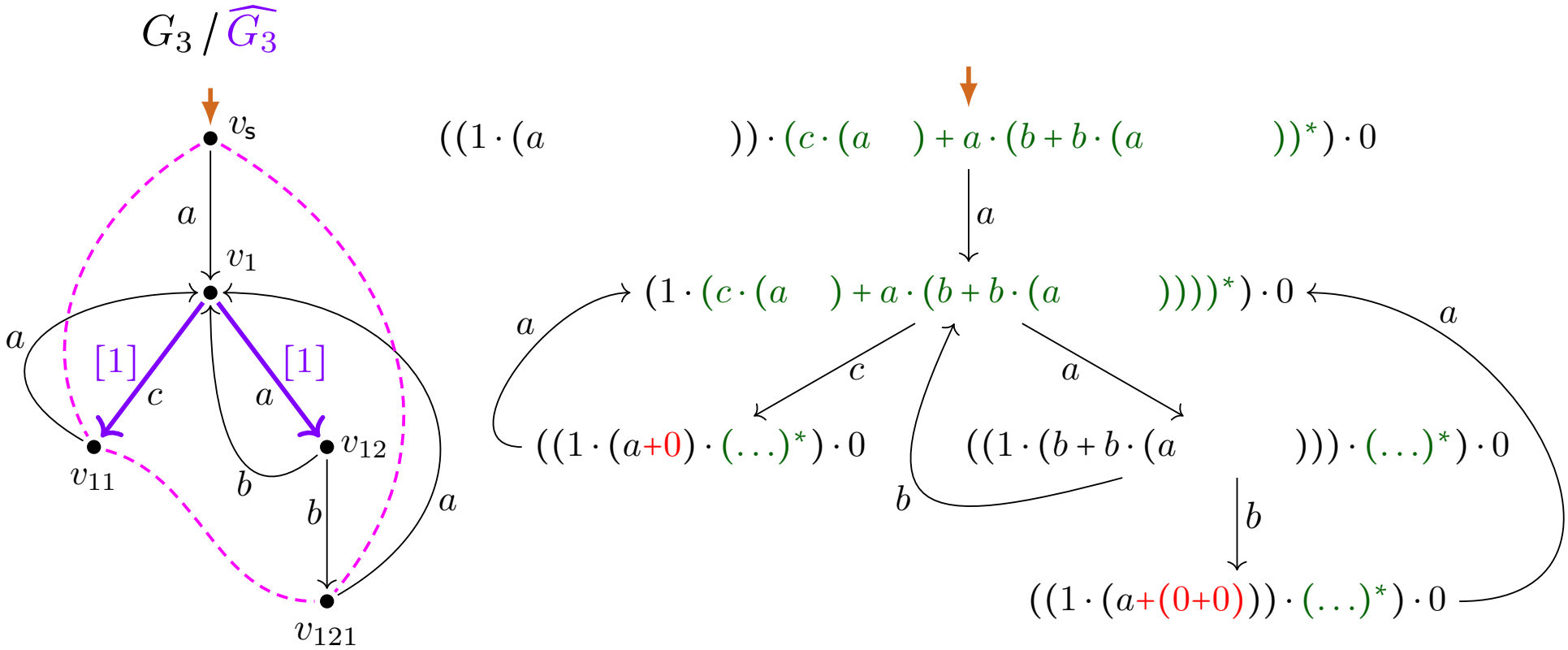
Refined extraction



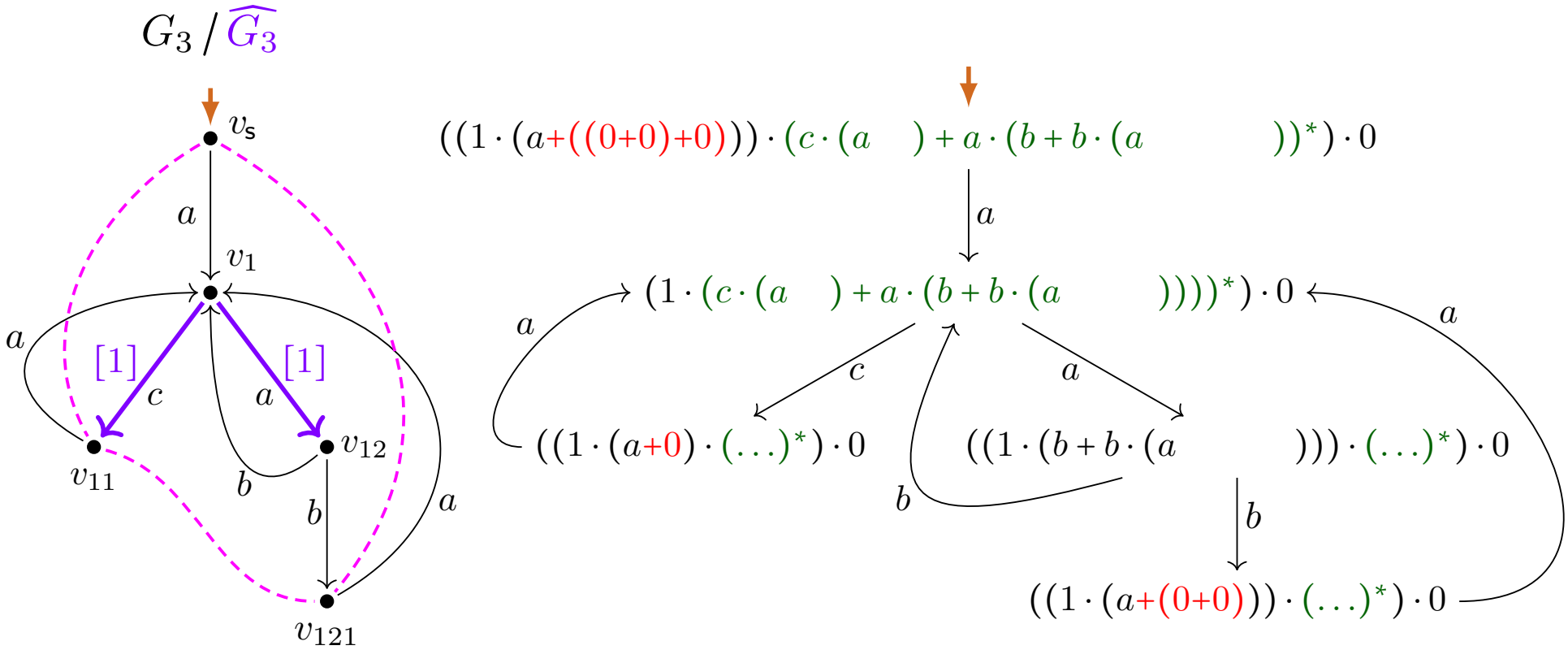
Refined extraction, adapted to prevent sharing



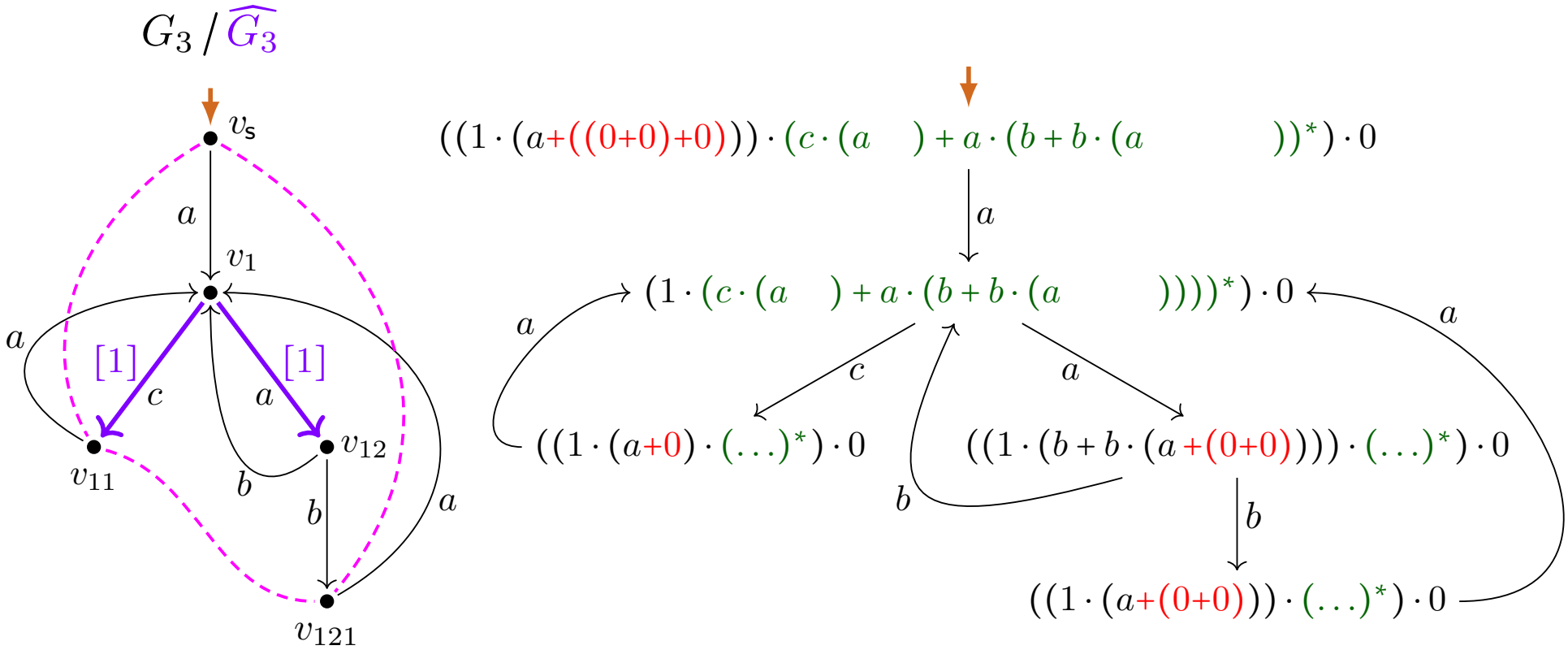
Refined extraction, adapted to prevent sharing



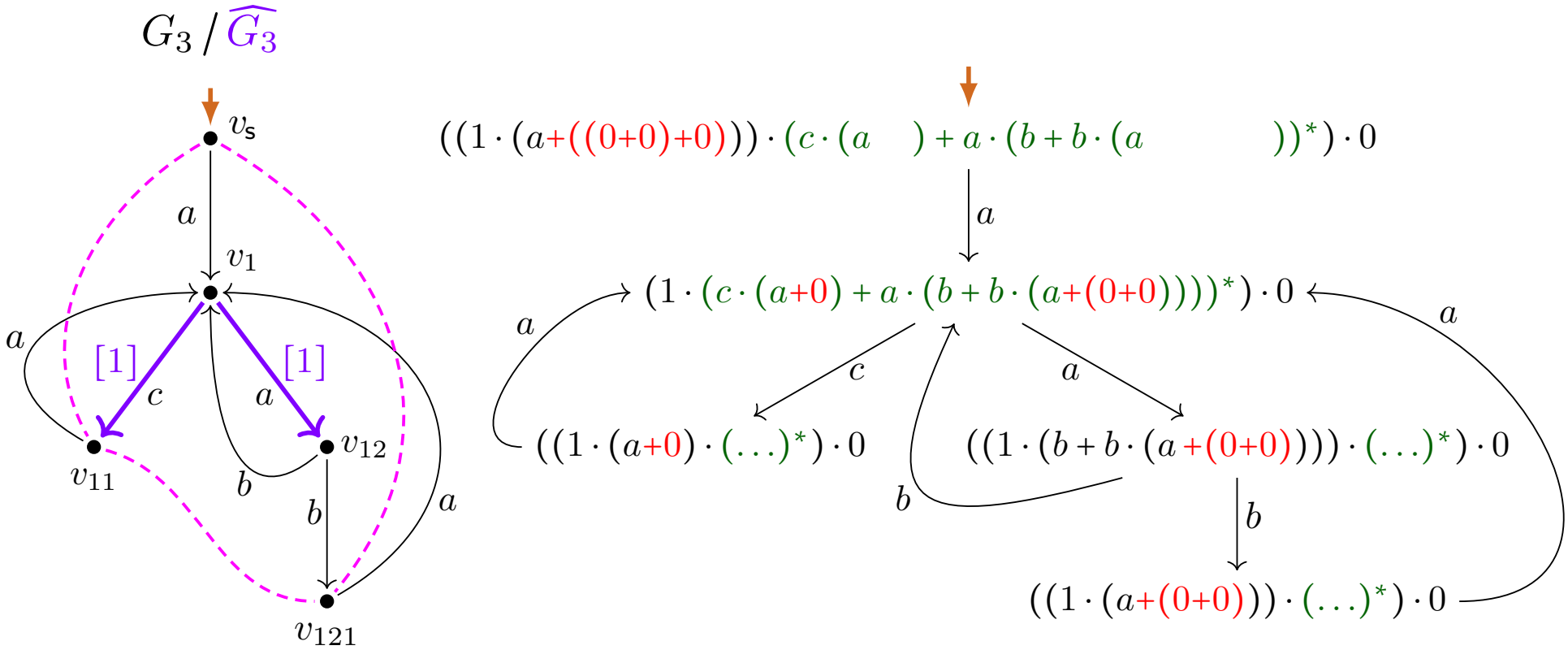
Refined extraction, adapted to prevent sharing



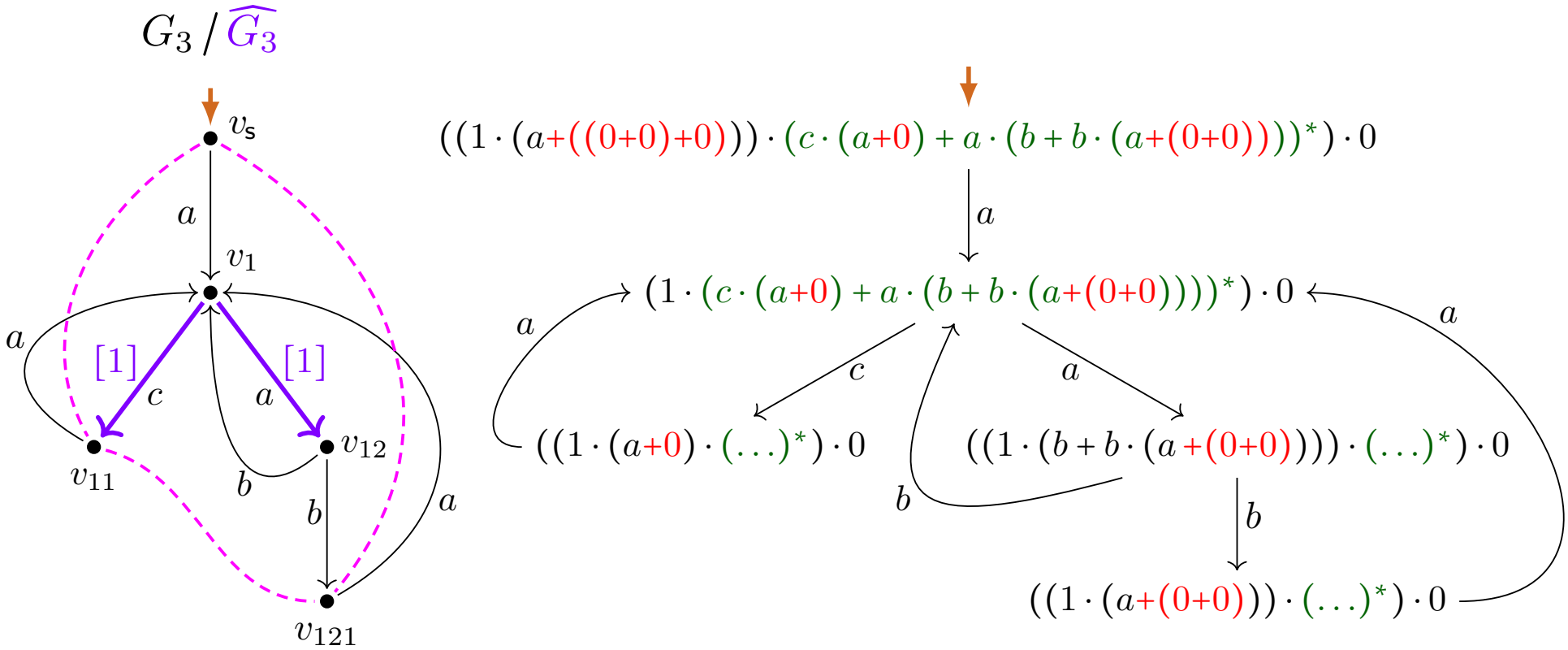
Refined extraction, adapted to prevent sharing



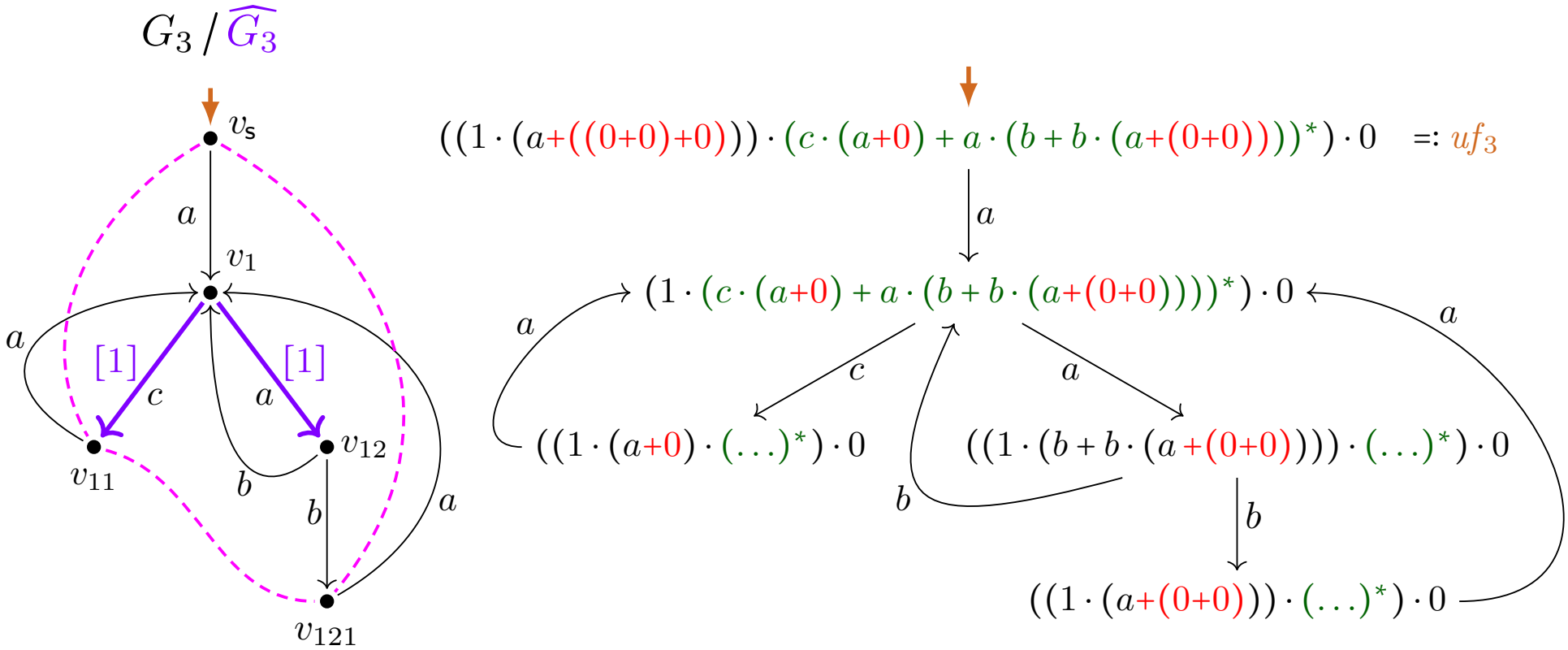
Refined extraction, adapted to prevent sharing



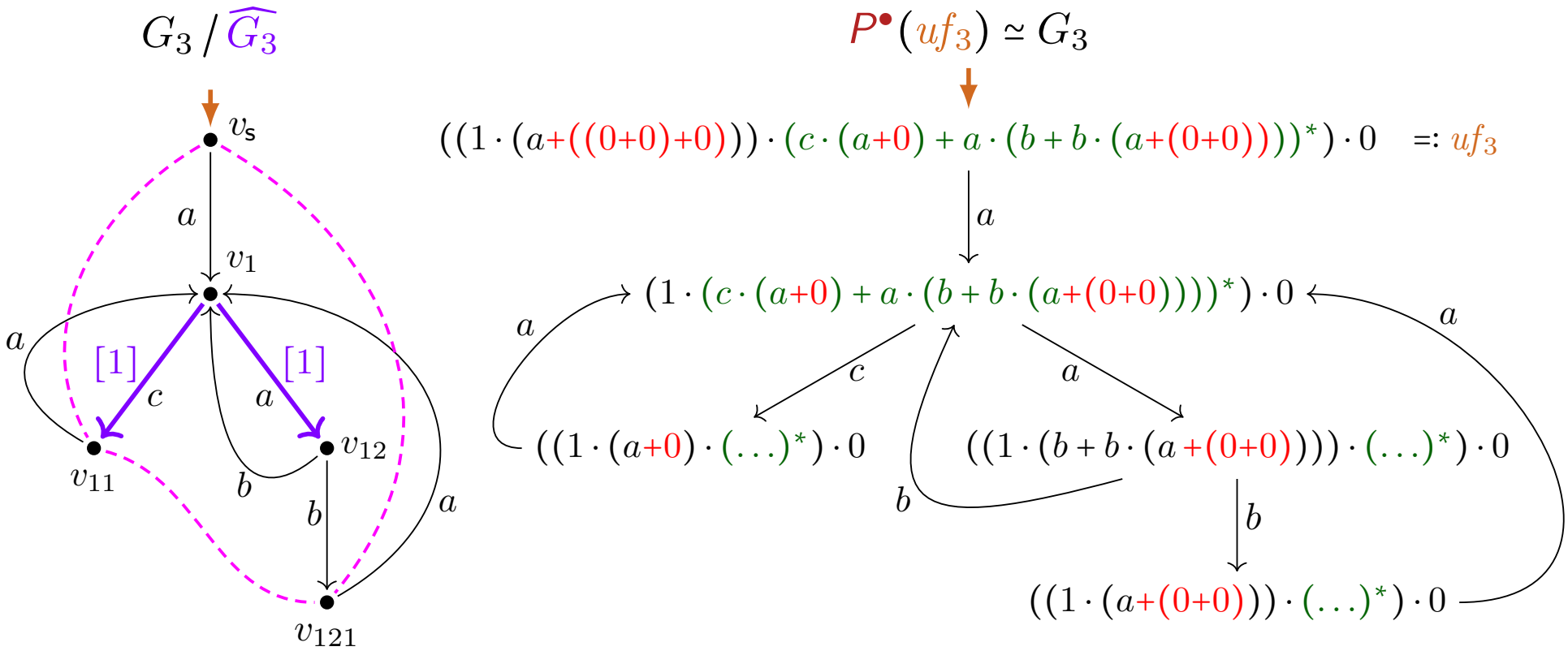
Refined extraction, adapted to prevent sharing



Refined extraction, adapted to prevent sharing



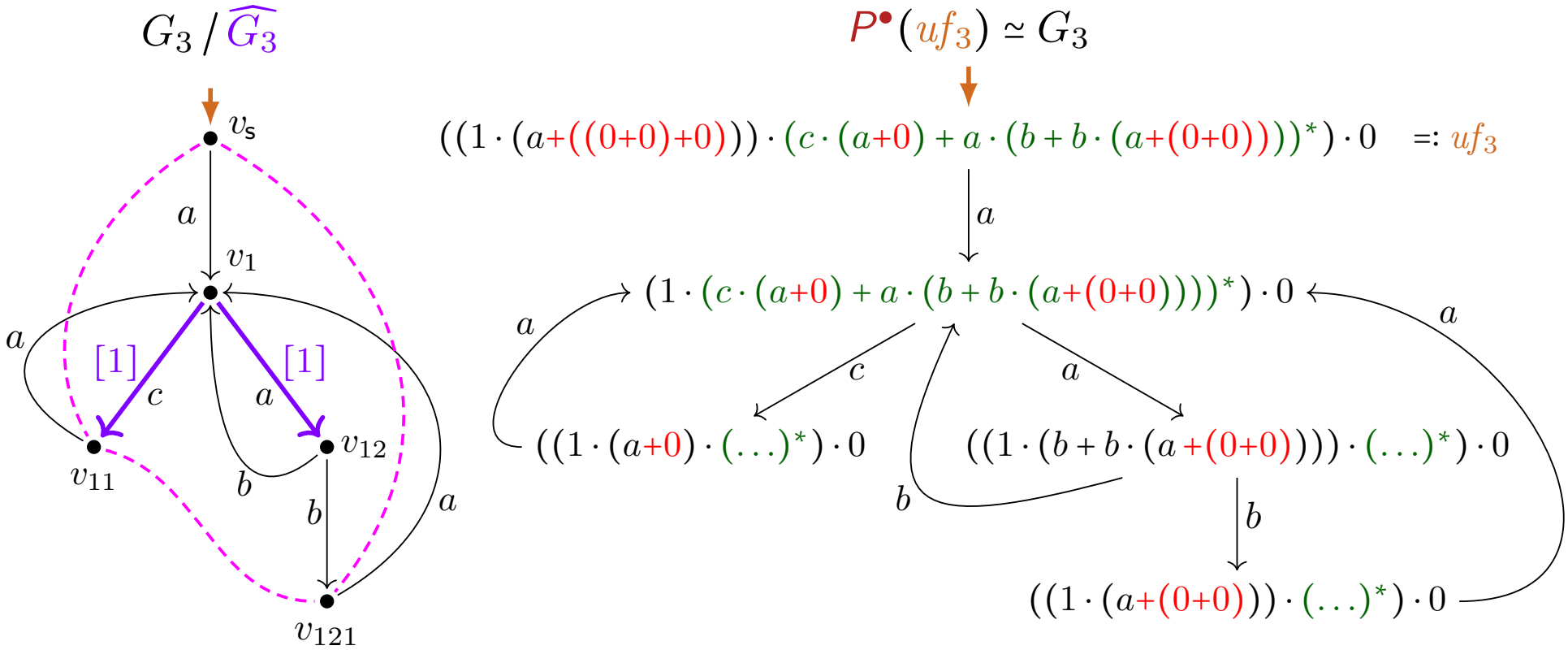
Refined extraction, adapted to prevent sharing



Refined extraction, adapted to prevent sharing

Lemma [Refined extraction, prevent sharing]

From every graph G with LLEE
a $(*/\perp)$ regular expression uf can be extracted such that $G \simeq P^\bullet(uf)$.



Characterizing the image of $P^\bullet$ for $(*/\dot{1})$ expressions

Lemma [Refined extraction, prevent sharing]

From every graph G with LLEE
a $(*/\dot{1})$ regular expression uf can be extracted such that $G \simeq P^\bullet(uf)$.

Characterizing the image of $P^\bullet$ for $(*/\pm)$ expressions

Lemma [Refined extraction, prevent sharing]

From every graph G with LLEE
a $(*/\pm)$ regular expression uf can be extracted such that $G \simeq P^\bullet(uf)$.

Statement

A process graph G is $P^\bullet$-expressible by an $(*/\pm)$ regular expression
$\iff$ G is finite, and G satisfies LLEE.

Characterizing the image of $P^\bullet$ for $(*/\pm)$ expressions

Lemma [Refined extraction, prevent sharing]

From every graph G with LLEE
a $(*/\pm)$ regular expression uf can be extracted such that $G \simeq P^\bullet(uf)$.

Statement

A process graph G is $P^\bullet$-expressible by an $(*/\pm)$ regular expression
$\iff$ G is finite, and G satisfies LLEE.

That is: $\mathrm{im}(P^\bullet)\,|_{RExp^{(*/\pm)}} = \{G \mid G \text{ is finite and satisfies LLEE}\}$.

Consequence

For every process graph G, TFAE:

(i) G is $[\![\cdot]\!]_P$-expressible by an $(*/\pm)$ regular expression.

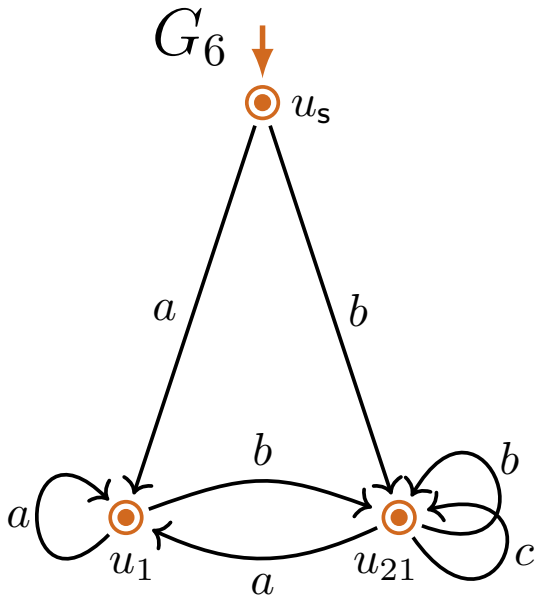
(ii) The bisimulation collapse of G is finite, and satisfies LLEE.

(iii) The bisimulation collapse of G
is $P^\bullet$-expressible by a $(*/\pm)$ regular expression.

Characterising the image of $P^\bullet$

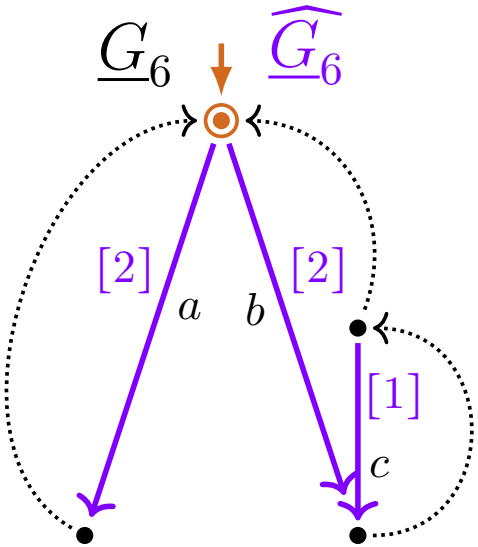
- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

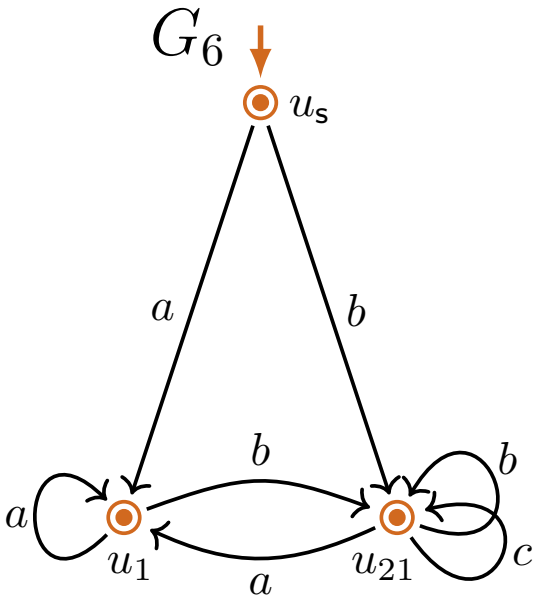
Refined extraction from 1-LLEE graph



$\neg\mathsf{LLEE}((\underline{G_6}])$

Refined extraction from 1-LLEE graph

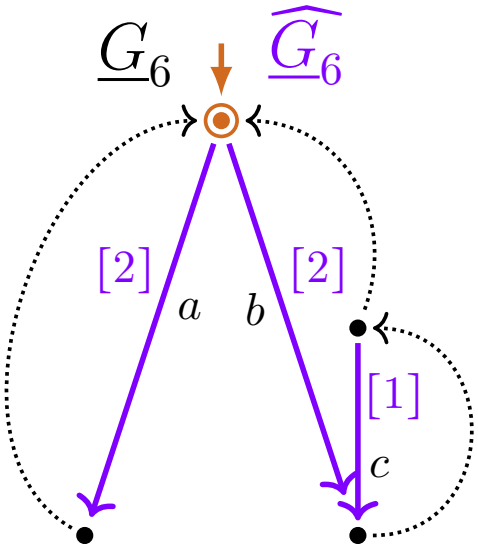


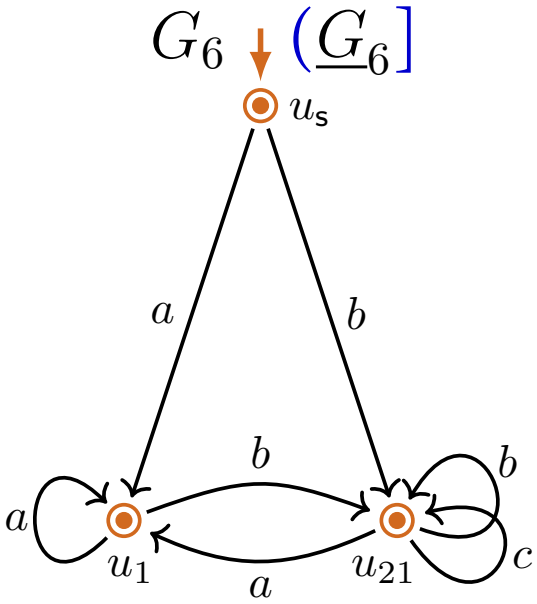


LLEE($\underline{G}_6$)

¬LLEE(($\underline{G}_6$])

Refined extraction from 1-LLEE graph

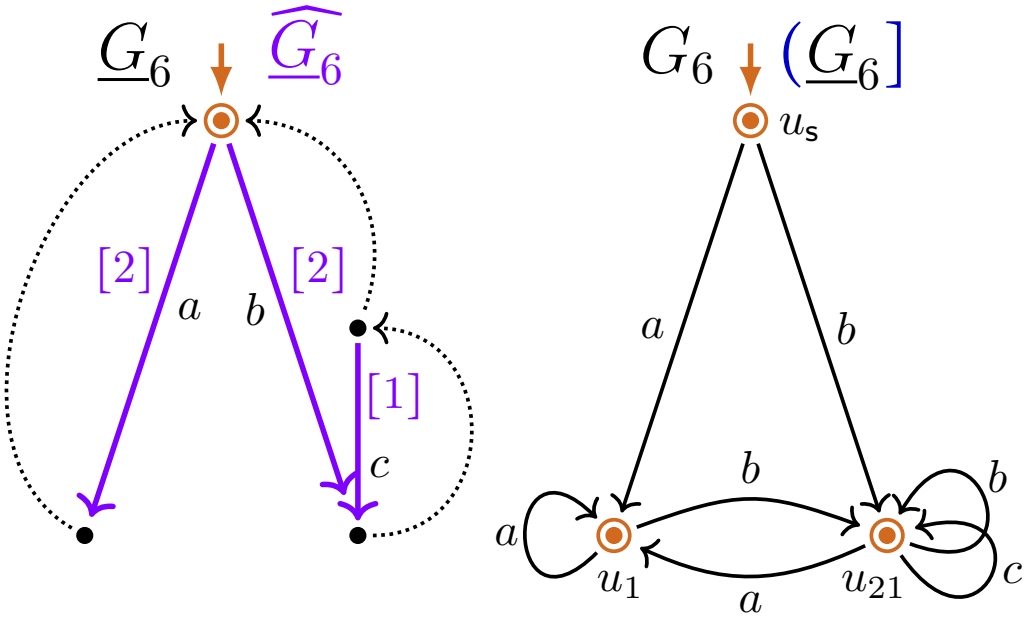




LLEE($\underline{G}_6$)

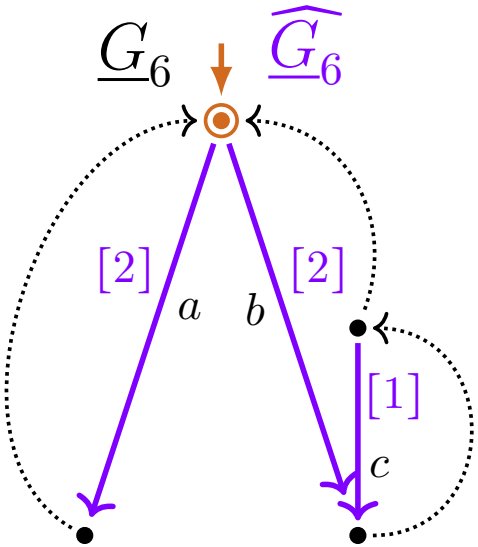
¬LLEE(($\underline{G}_6$])

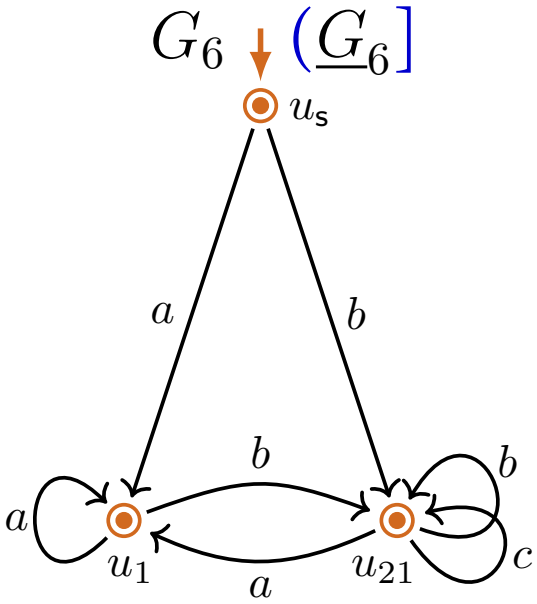
Refined extraction from 1-LLEE graph

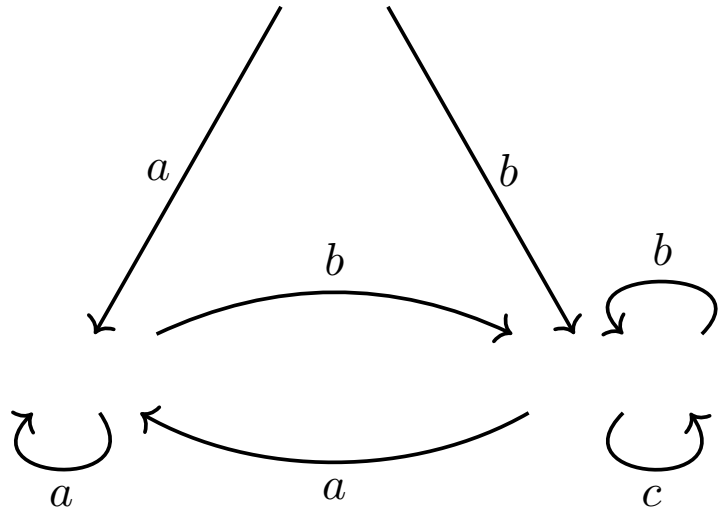


LLEE($\underline{G}_6$) 1-LLEE(($\underline{G}_6$]), ¬LLEE(($\underline{G}_6$])

Refined extraction from 1-LLEE graph

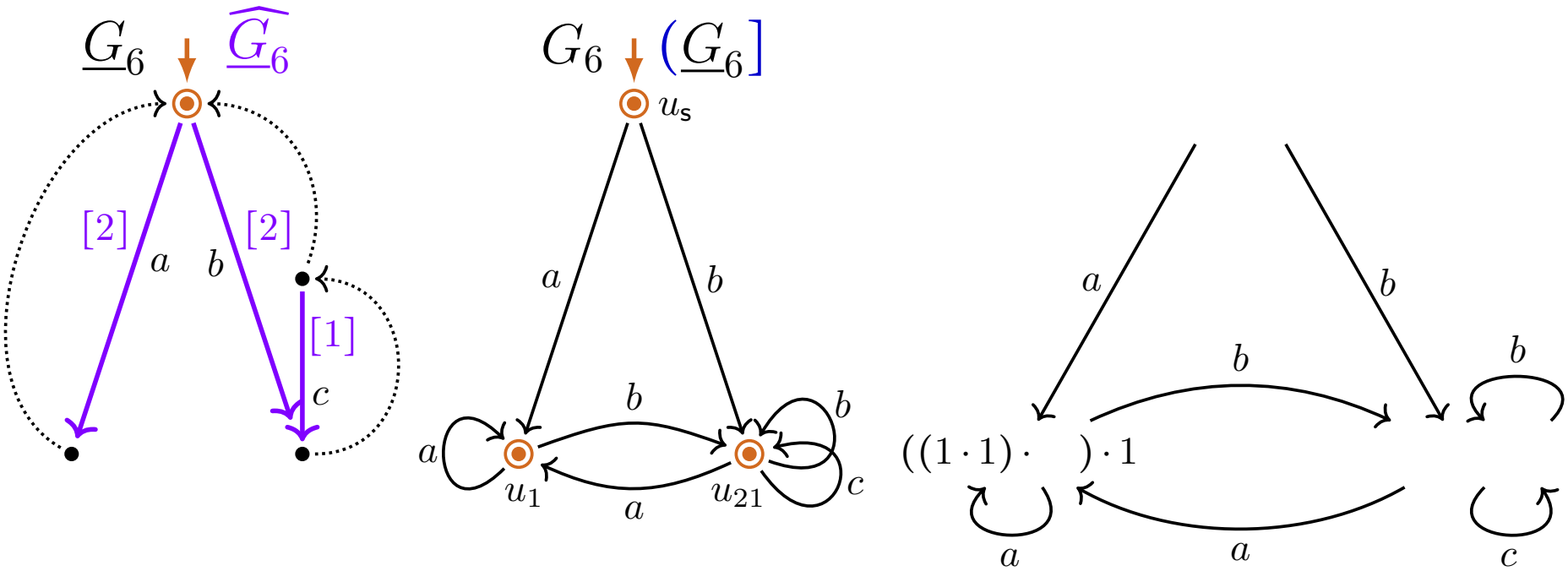






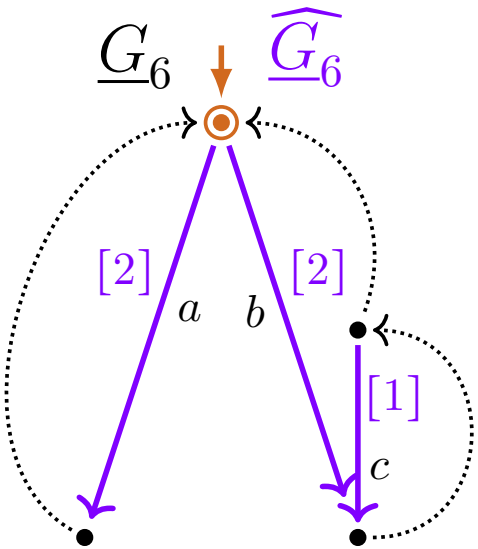
LLEE($\underline{G}_6$) 1-LLEE($(\underline{G}_6]$), ¬LLEE($(\underline{G}_6]$)

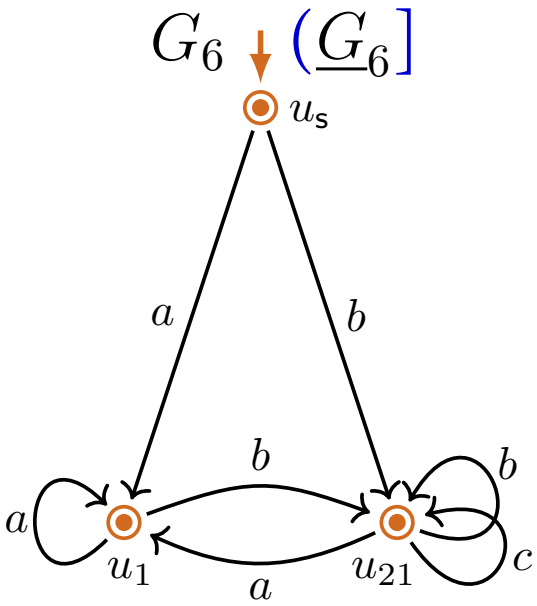
Refined extraction from 1-LLEE graph

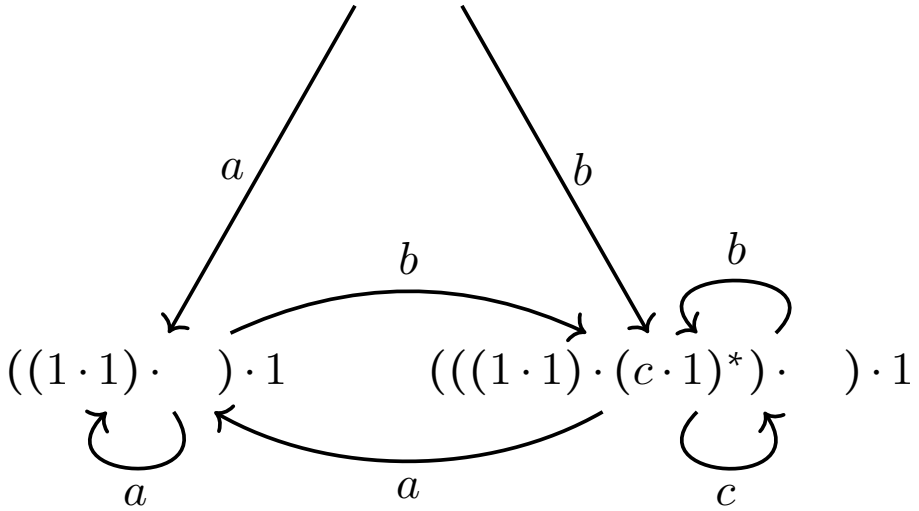


LLEE($\underline{G}_6$) 1-LLEE($(\underline{G}_6]$), ¬LLEE($(\underline{G}_6]$)

Refined extraction from 1-LLEE graph

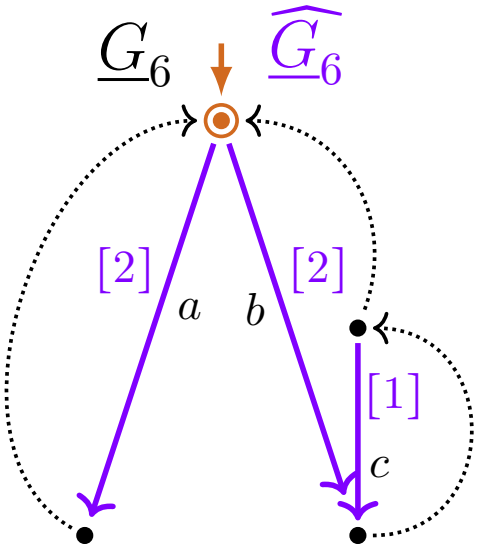


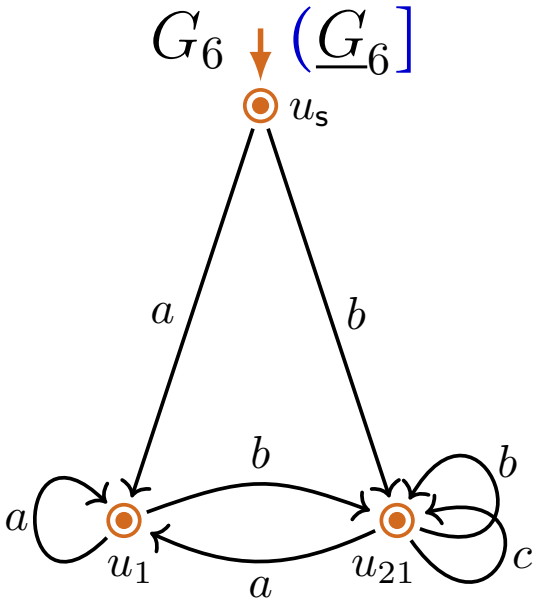


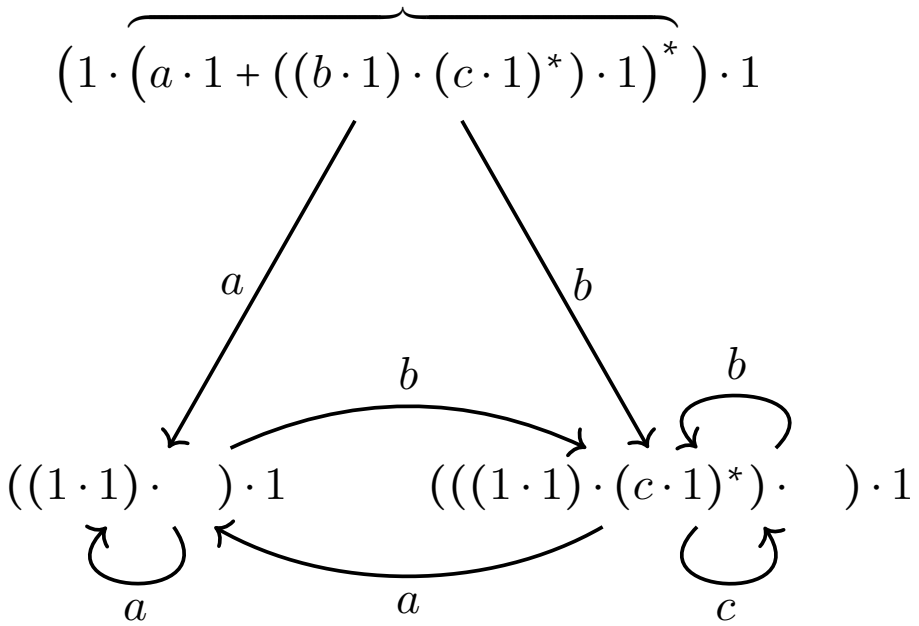


LLEE($\underline{G}_6$) 1-LLEE(($\underline{G}_6$]), ¬LLEE(($\underline{G}_6$])

Refined extraction from 1-LLEE graph

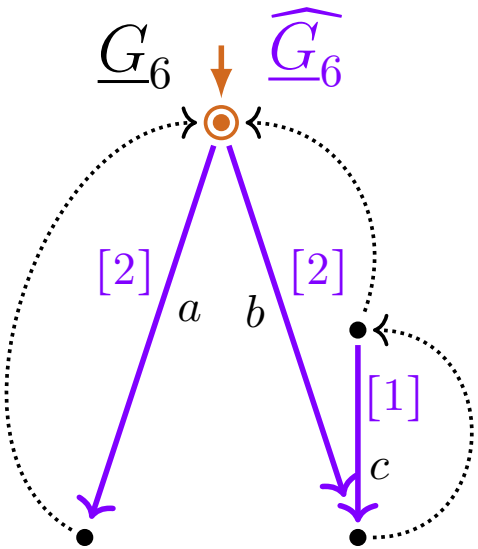


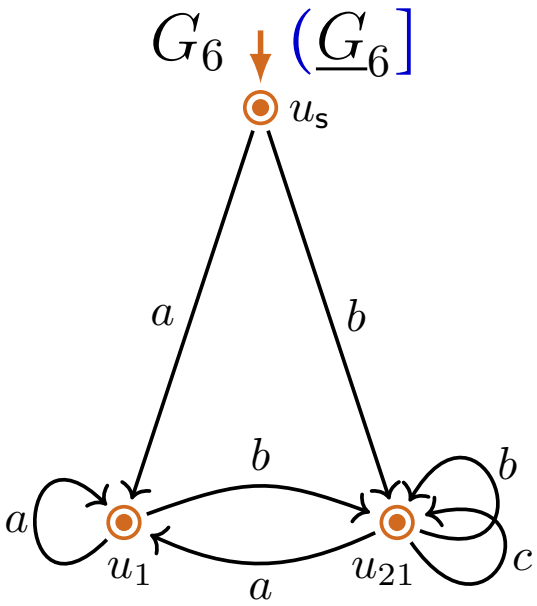


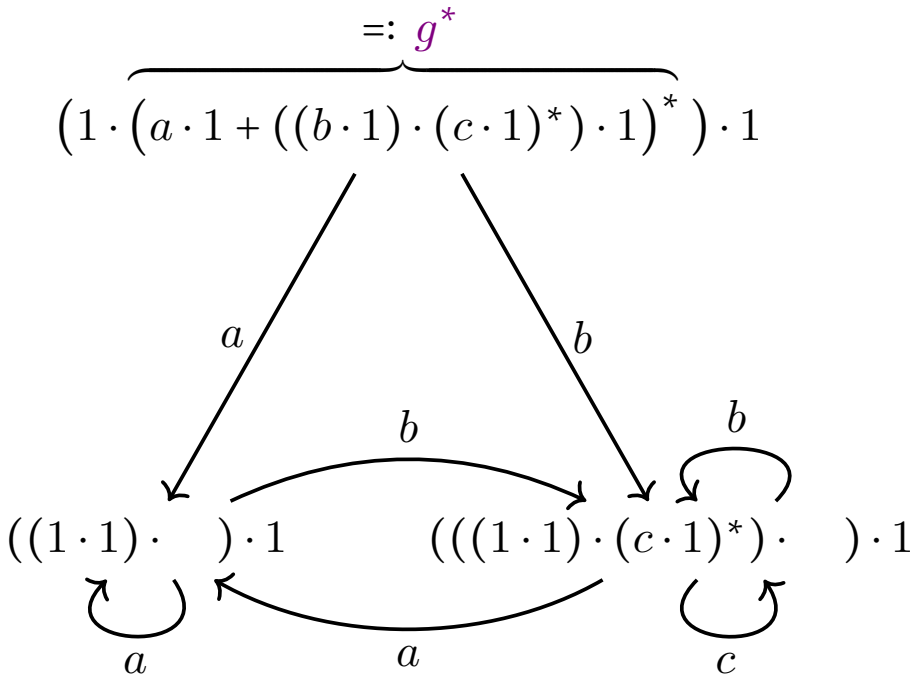


LLEE($\underline{G}_6$) 1-LLEE($(\underline{G}_6]$), ¬LLEE($(\underline{G}_6]$)

Refined extraction from 1-LLEE graph

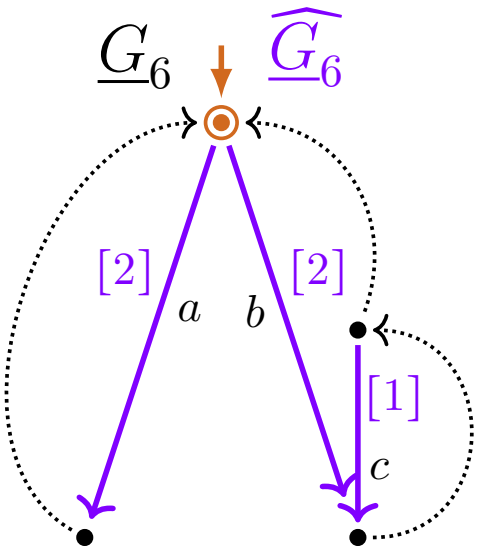


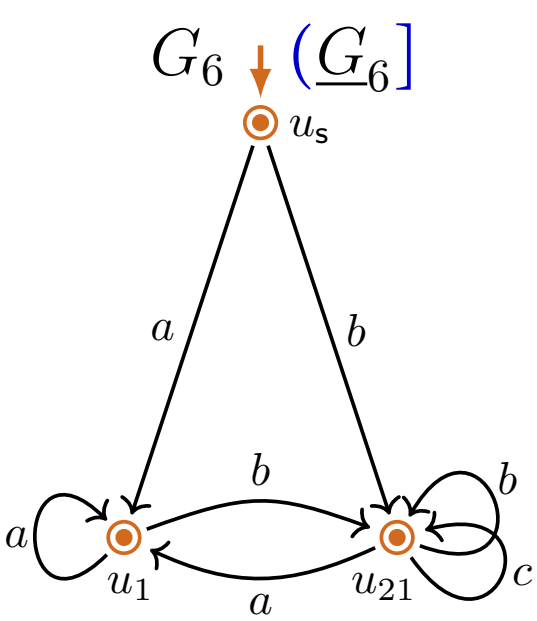


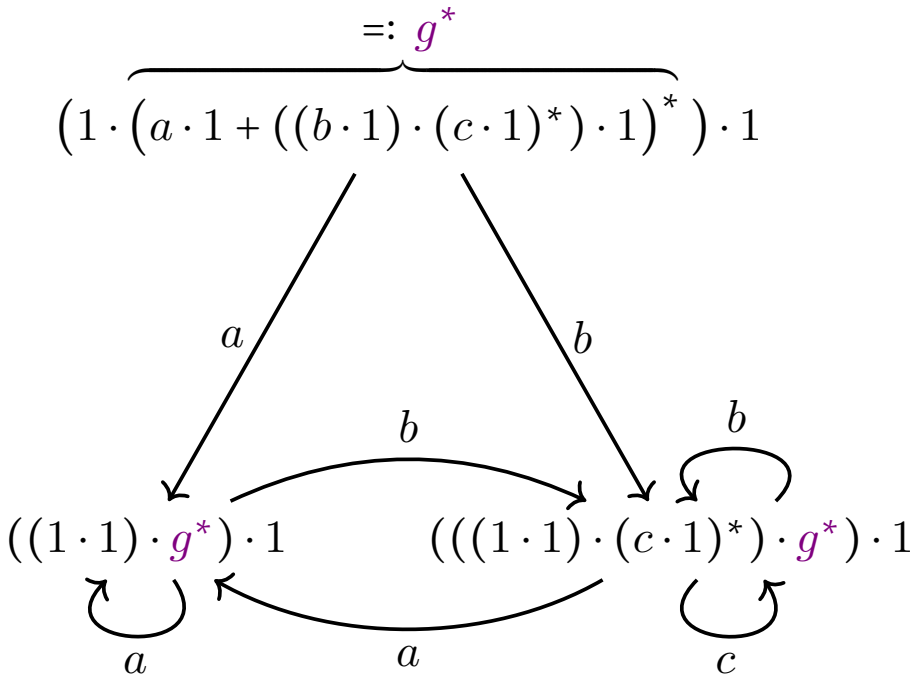


LLEE($\underline{G_6}$) 1-LLEE($(\underline{G_6}]$), ¬LLEE($(\underline{G_6}]$)

Refined extraction from 1-LLEE graph

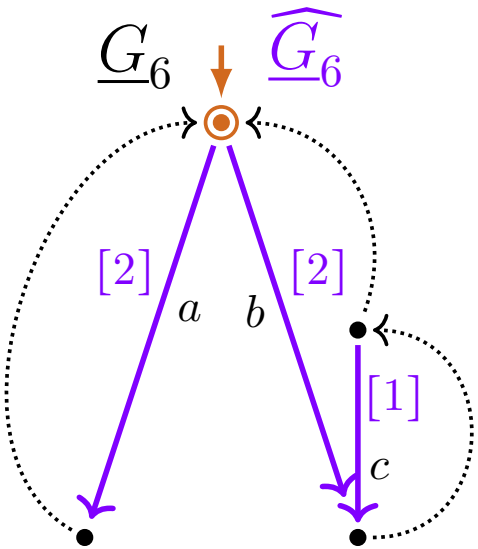


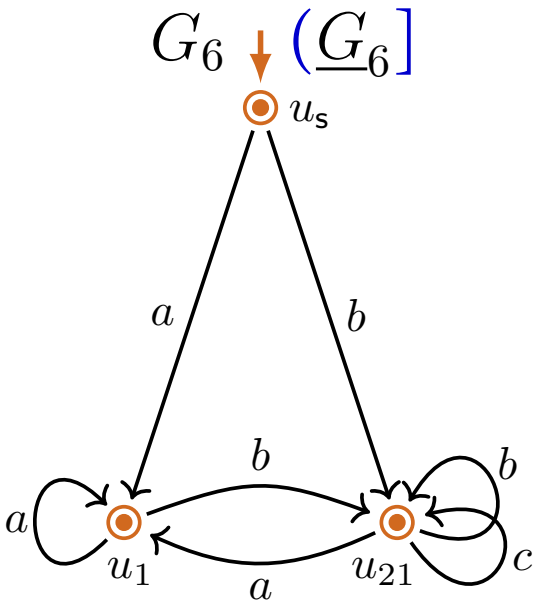


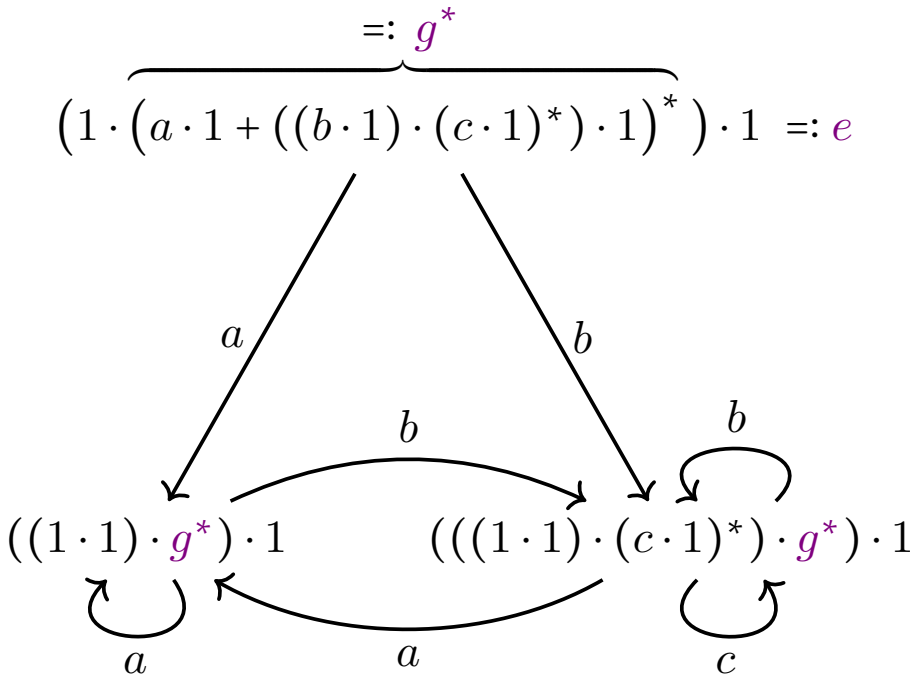


LLEE($\underline{G_6}$) 1-LLEE(($\underline{G_6}$]), ¬LLEE(($\underline{G_6}$])

Refined extraction from 1-LLEE graph

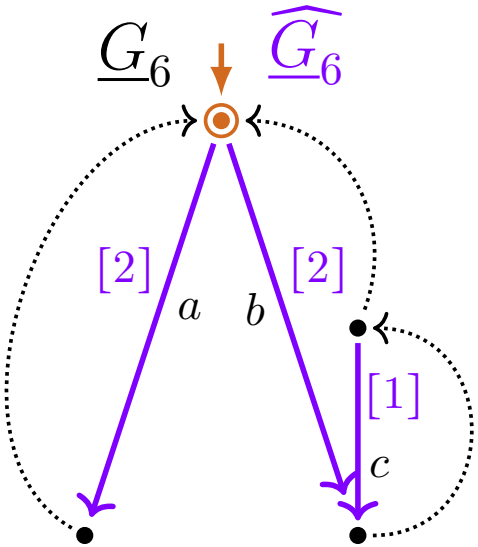


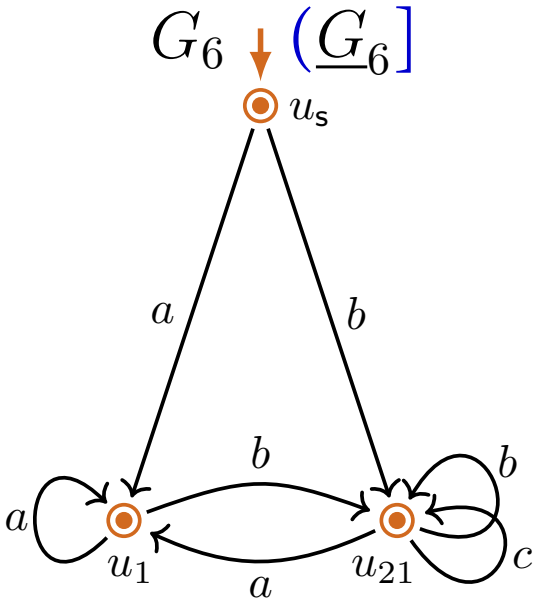


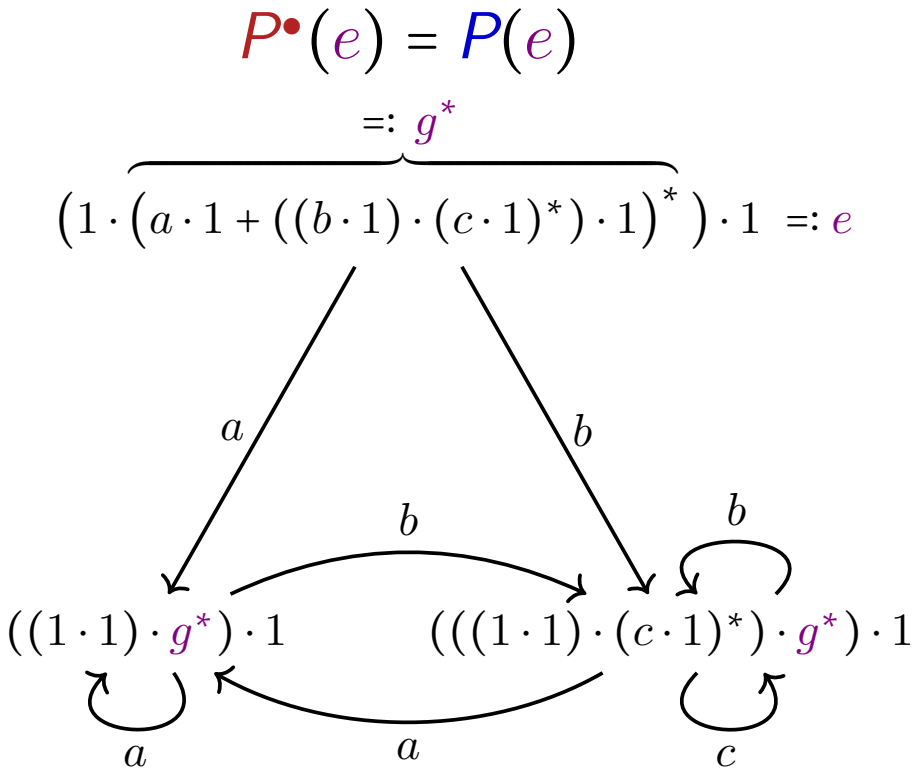


LLEE($\underline{G_6}$) 1-LLEE($(\underline{G_6}]$), ¬LLEE($(\underline{G_6}]$)

Refined extraction from 1-LLEE graph







LLEE(G_6) 1-LLEE($(G_6]$), ¬LLEE($(G_6]$)

Characterizing the image of $P^\bullet$

Lemma [Refined extraction, prevent sharing]

From every graph G with 1-LLEE
a regular expression e can be extracted such that $G \simeq P^\bullet(e)$.

Characterizing the image of $P^\bullet$

Lemma [Refined extraction, prevent sharing]

From every graph G with 1-LLEE
a regular expression e can be extracted such that $G \simeq P^\bullet(e)$.

Lemma *(G, LICS 2024)*

$P(e)$ satisfies 1-LLEE, for all regular expressions.

Holds also for $P^\bullet(e)$.

Characterizing the image of $P^\bullet$

Lemma [Refined extraction, prevent sharing]

From every graph G with 1-LLEE
a regular expression e can be extracted such that $G \simeq P^\bullet(e)$.

Lemma *(G, LICS 2024)*

$P(e)$ satisfies 1-LLEE, for all regular expressions.

Holds also for $P^\bullet(e)$.

Statement

A process graph G is $P^\bullet$-expressible by a regular expression
$\iff$ G is finite, and G satisfies 1-LLEE.

Characterizing the image of $P^\bullet$

Lemma [Refined extraction, prevent sharing]

From every graph G with 1-LLEE
a regular expression e can be extracted such that $G \simeq P^\bullet(e)$.

Lemma *(G, LICS 2024)*

$P(e)$ satisfies 1-LLEE, for all regular expressions.

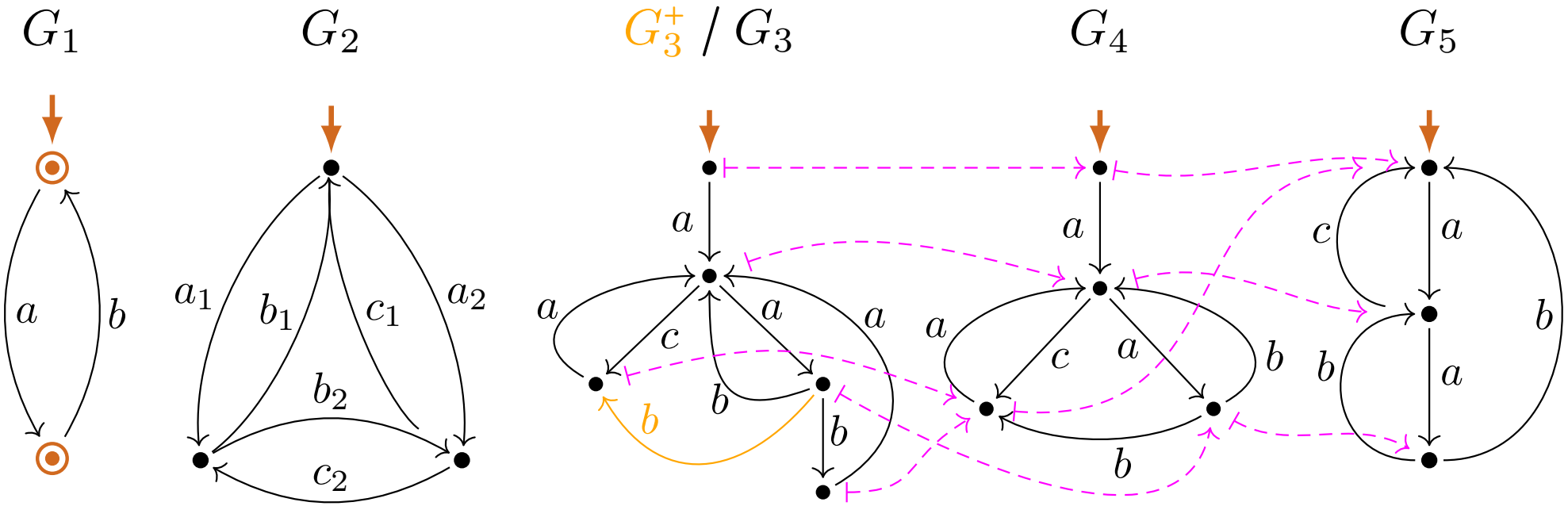
Holds also for $P^\bullet(e)$.

Statement

A process graph G is $P^\bullet$-expressible by a regular expression
$\iff$ G is finite, and G satisfies 1-LLEE.

That is: $\mathrm{im}(P^\bullet) = \{G \mid G \text{ is finite and satisfies } 1\text{-LLEE}\}$
$= \{(\underline{G}] \mid 1\text{-graph } \underline{G} \text{ is finite and satisfies LLEE}\}$.

$P^\bullet$-expressibility (final results)



G_1	G_2	G_3^+ / G_3	G_4	G_5
not $P^\bullet$-expressible	**not** $P^\bullet$-expressible	**not** $P^\bullet$-expressible / $P^\bullet((*/\perp))$-expr.	$P^\bullet((*/\perp))$-expr.	$P^\bullet((*/\perp))$-expr.
not $[\![\cdot]\!]_P$-expressible	**not** $[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible	$[\![\cdot]\!]_P$-expressible
not LLEE	**not** LLEE	**not** LLEE / LLEE	LLEE	LLEE
not 1-LLEE	**not** 1-LLEE	**not** 1-LLEE/ 1-LLEE	1-LLEE	1-LLEE

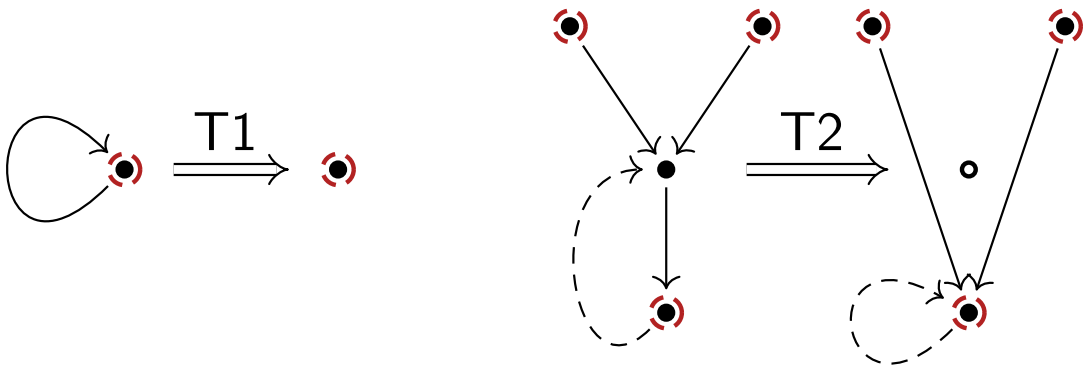
Relationship with Co-Reducibility

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- characterisation of image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- characterization of image of $P^\bullet$
- Relationship with co-reducibility

LLEE = co-reducible$^+$ to acyclic graph

Connection to work by Laarakker and Kappé (TERMGRAPH 2026) on:

- co-reducibility of state graphs,
- based on flow graph reducibility by Hecht and Ullman (1972).

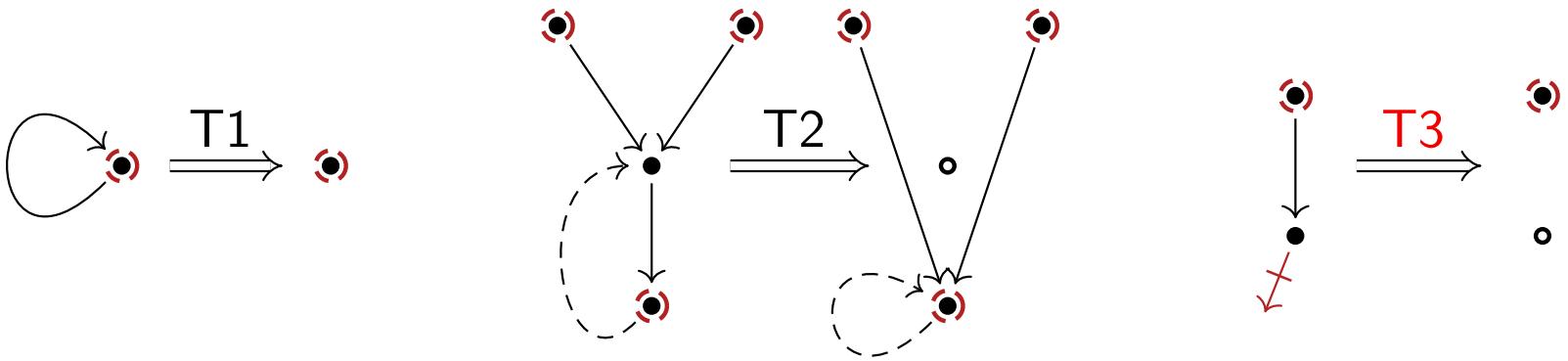


LLEE = co-reducible$^+$ to acyclic graph

Connection to work by Laarakker and Kappé (TERMGRAPH 2026) on:

- co-reducibility of state graphs,
- based on flow graph reducibility by Hecht and Ullman (1972).

We extend co-reducibility to co-reducibility$^+$ by adding a prune operation:

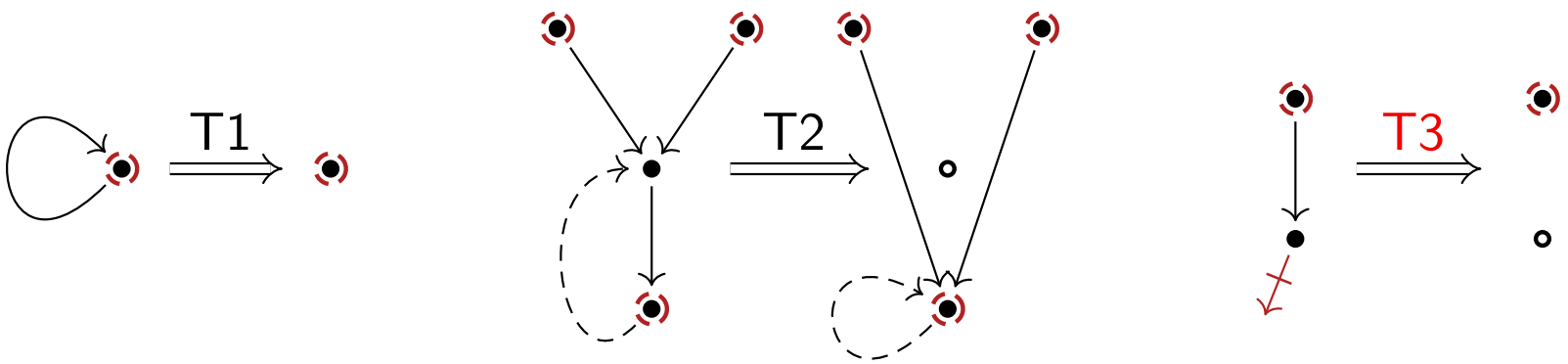


LLEE = co-reducible$^+$ to acyclic graph

Connection to work by Laarakker and Kappé (TERMGRAPH 2026) on:

- co-reducibility of state graphs,
- based on flow graph reducibility by Hecht and Ullman (1972).

We extend co-reducibility to co-reducibility$^+$ by adding a prune operation:



Statement

A process graph G has the property LLEE

$\iff$ the state graph of G is co-reducible$^+$ to an acyclic graph.

Summary

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\dot{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\dot{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- LLEE $\hat{=}$ image of $P^\bullet$ restricted to $(*/\dot{1})$ expressions
- 1-LLEE $\hat{=}$ image of $P^\bullet$
- Relationship with co-reducibility

Summary

- Regular expressions (unary/binary star, 1-free/under-star-1-free $(*/\not{1})$)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
- Layered Loop existence and elimination (LLEE/1-LLEE)
  - interpretation/extraction correspondences with $(*/\not{1})$/all reg. expr.
  - LLEE is preserved by bisimulation collapse (1-LLEE is not)
- TERMGRAPH 2024: compact process interpretation $P^\bullet$
  - image of $P^\bullet$ is closed under bisimulation collapse (image of P is not)
- LLEE $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$ expressions
- 1-LLEE $\hat{=}$ image of $P^\bullet$
- Relationship with co-reducibility

IWC and IFIP-1.6 Talks

IWC (Friday, 24 July)

CG: *Loop Elimination in Process Graphs is Confluent When Pruning Steps are Added*

Ext. Abstract https://clegra.github.io/lf/le-conf-IWC-26.pdf

- critical-pair like analysis of loop elimination steps $\longrightarrow_{\text{elim}}$
- $\longrightarrow_{\text{elim}}$ forks are joinable by $\longrightarrow_{\text{elim}}$ steps and prune steps $\longrightarrow_{\text{prune}}$

IFIP-1.6 meeting (yesterday)

CG: *Finding Small Steps for Hard Problems, in Work on the Process Interpretation of Regular Expressions*

Slides https://clegra.github.io/lf/IFIP-1_6-2026.pdf

- Reducing for Graphs Representing Regular Expressions under Bisimilarity

Resources

- Slides/extended abstract on clegra.github.io
  - slides: .../lf/TG-26.pdf
  - extended abstract: .../lf/pi-struc-TG-26.pdf
- CG: Closing the Image of the Process Interpretation of 1-Free Regular Expressions Under Bisimulation Collapse
  - TERMGRAPH 2024, extended abstract
- CG: The Image of the Process Interpretation of Regular Expressions is Not Closed under Bisimulation Collapse
  - arXiv:2303.08553, 2021/2023.
- CG: Milner's Proof System for Regular Expressions Modulo Bisimilarity is Complete
  - LICS 2022, arXiv:2209.12188, poster.
- CG, Wan Fokkink: A Complete Proof System for 1-Free Regular Expressions Modulo Bisimilarity,
  - LICS 2020, arXiv:2004.12740, video on youtube.
- CG: Modeling Terms by Graphs with Structure Constraints
  - TERMGRAPH 2018, EPTCS 288, arXiv:1902.02010.