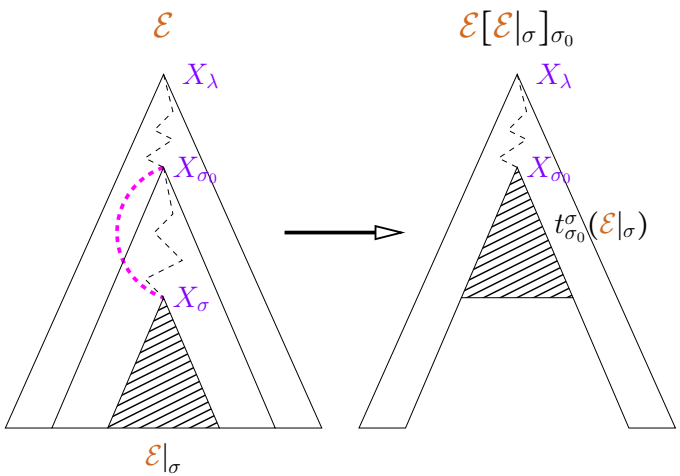
Finding Small Steps for Hard Questions, in work on the Process Semantics of Regular Expressions



Clemens Grabmayer

https://clegra.github.io

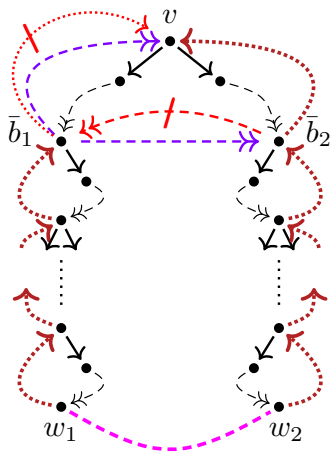
Department of Computer Science

G S GRAN SASSO SCIENCE INSTITUTE

S I SCHOOL OF ADVANCED STUDIES

Scuola Universitaria Superiore

L'Aquila, Italy



IFIP 1.6 Working Group Meeting

Lisbon

July 18, 2026

Overview

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

Overview

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

Overview

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

Overview

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

Regular Expression and Milner's Process Interpretation

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\triangleq$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

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

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

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

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

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

$a \xmapsto{P}$ 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 termination

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

$\mathbf{0} \xmapsto{P}$ 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 $P(e)$)

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 $\varnothing$

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 $\varnothing$, 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-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:

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

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)*

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

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

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

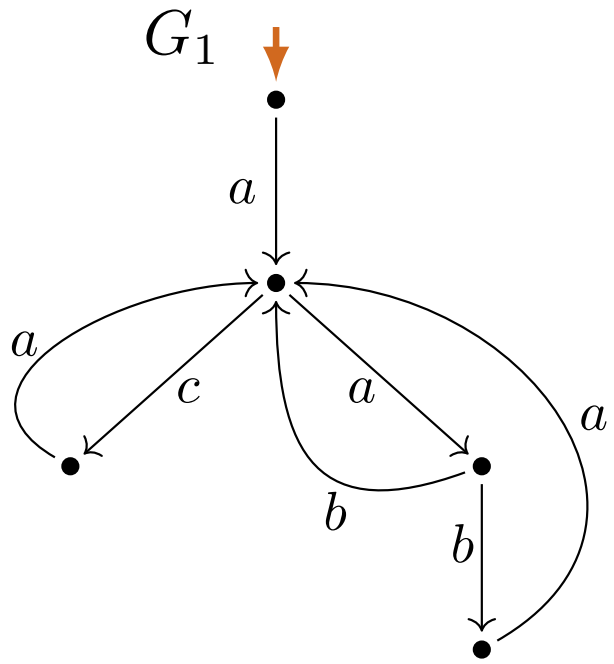
$e_1 + e_2 \xrightarrow{P}$ *(choice)* execute $P(e_1)$ or $P(e_2)$

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

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

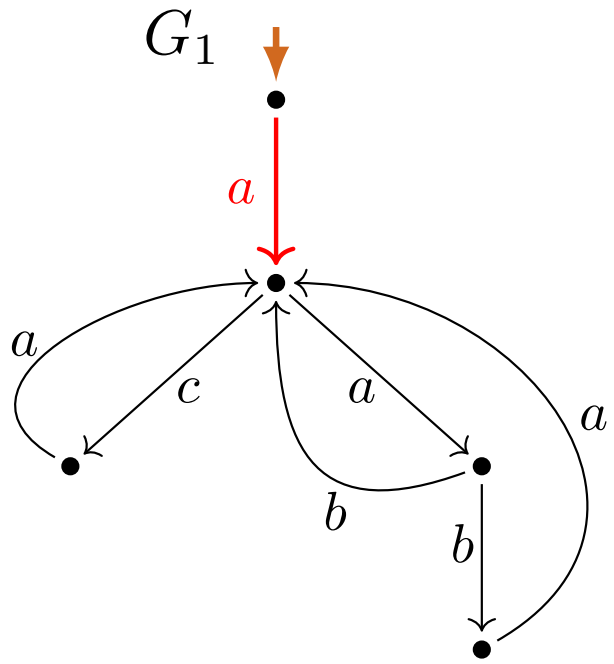
$[\![e]\!]_P := [P(e)]_{\leftrightarrow}$ (bisimilarity equivalence class of process $P(e)$)

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



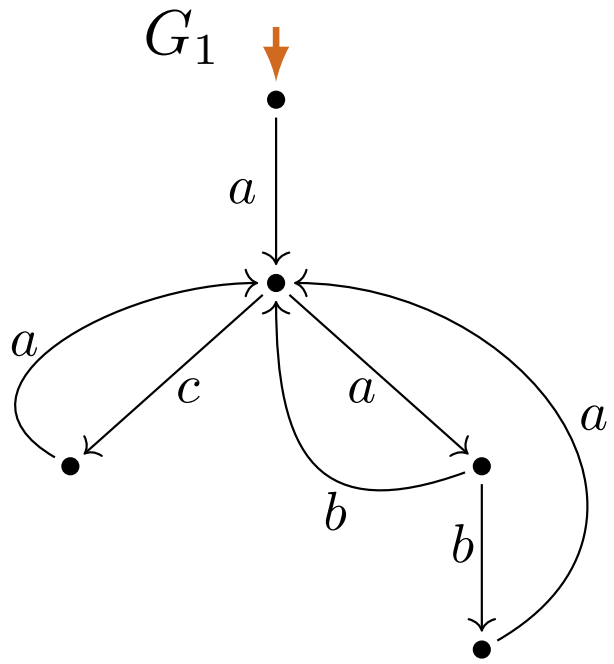
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0)}^{f}\Big)$$

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



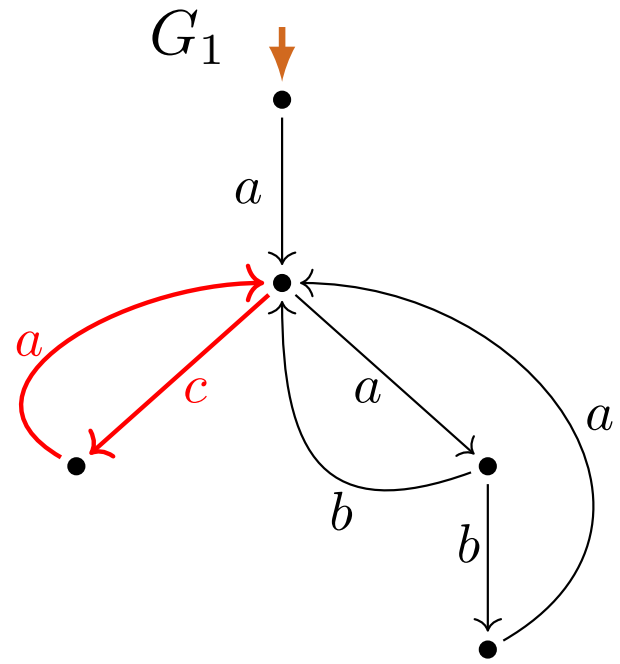
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0}^{f}\Big)$$

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



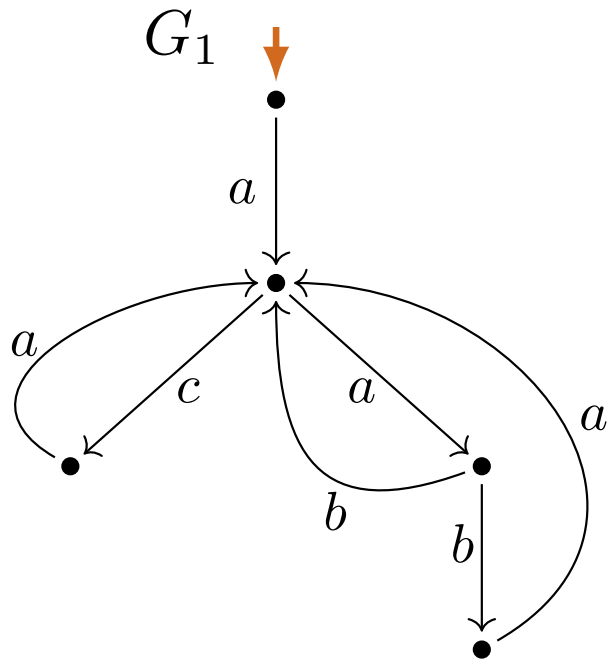
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0)}^{f}\Big)$$

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



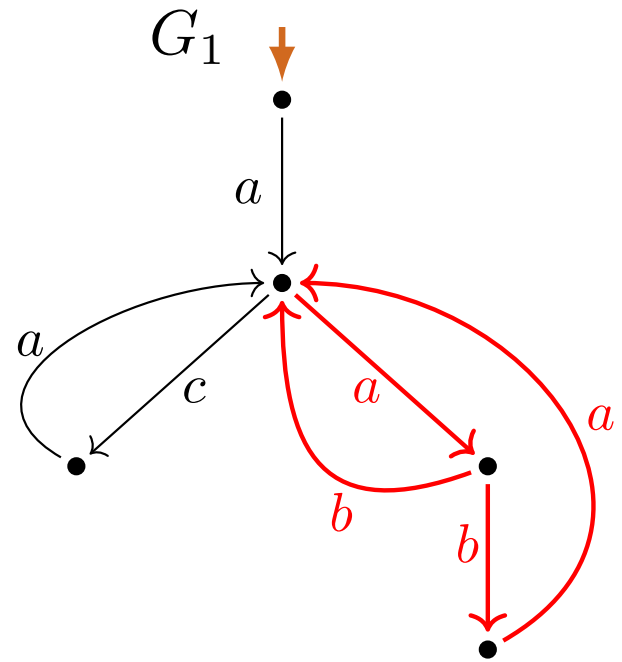
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0)}^{f}\Big)$$

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



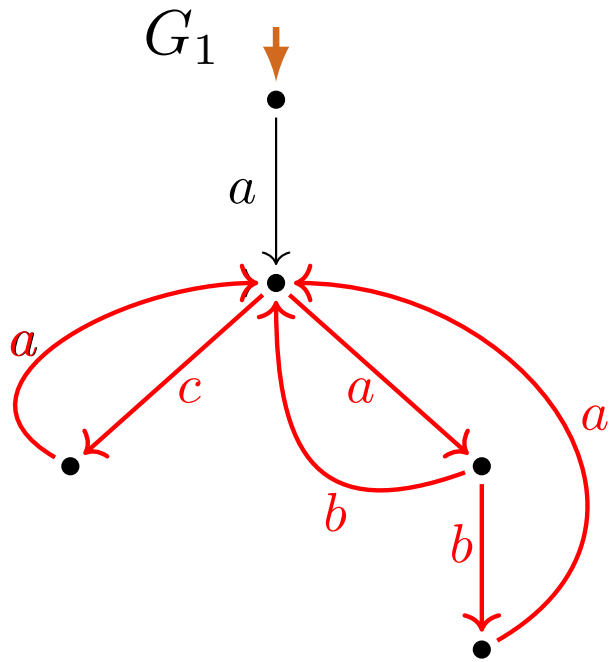
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0)}^{f}\Big)$$

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



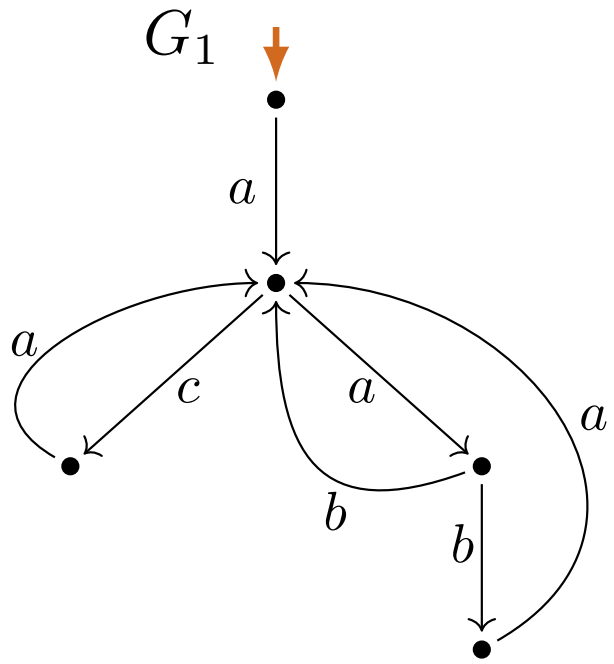
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0)}^{f}\Big)$$

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



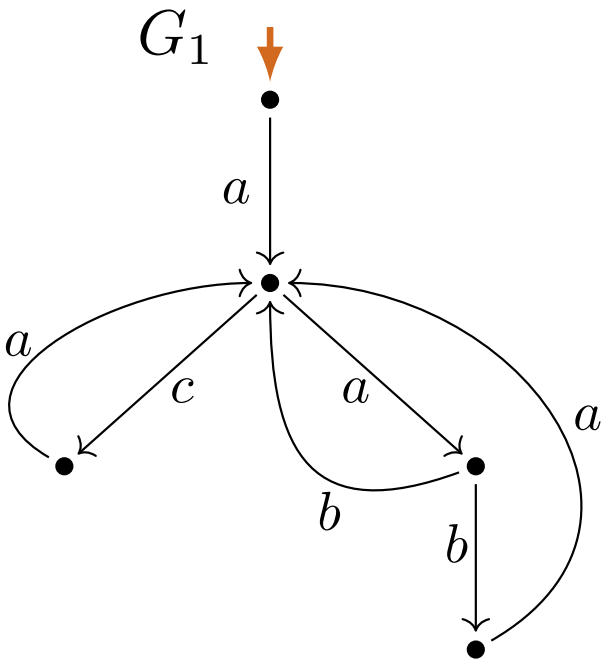
$$P\Big(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0)}^{f}\Big)$$

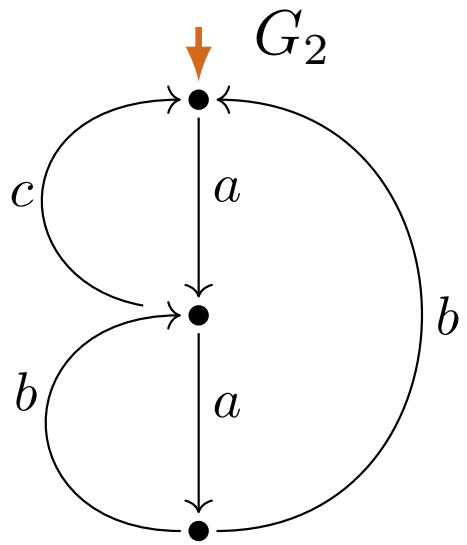
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



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

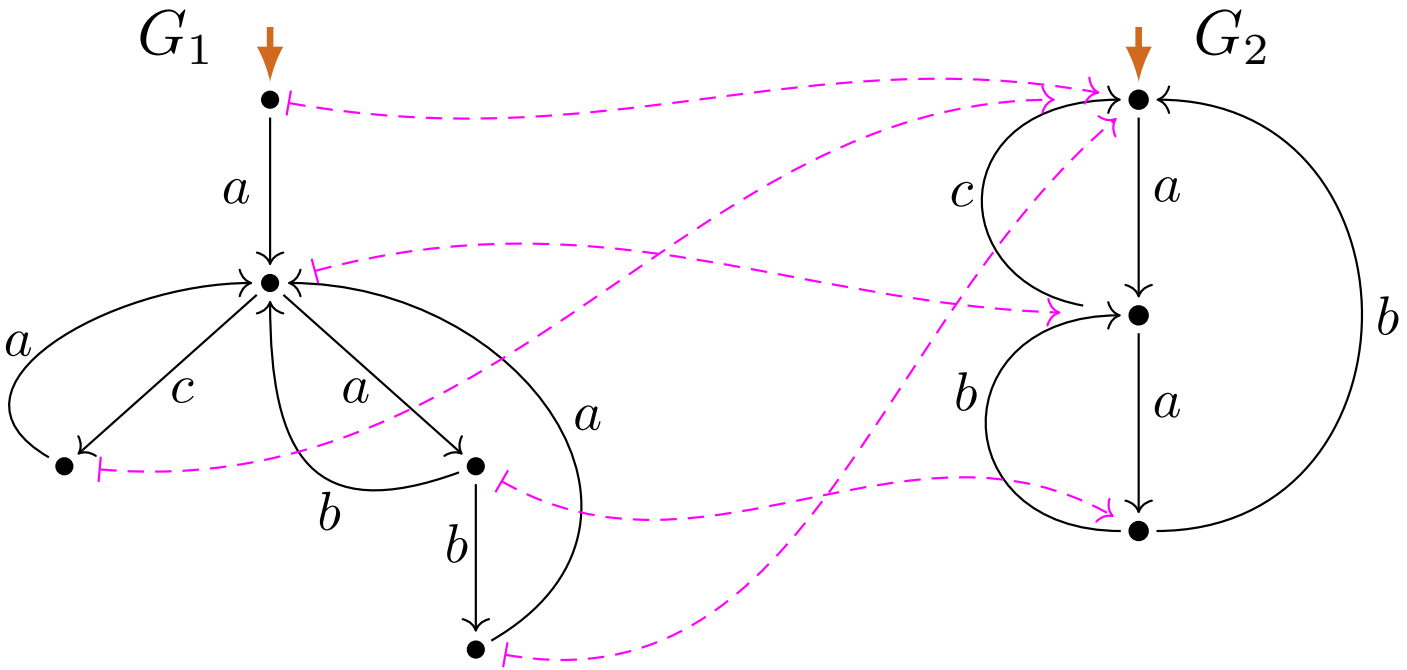
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)





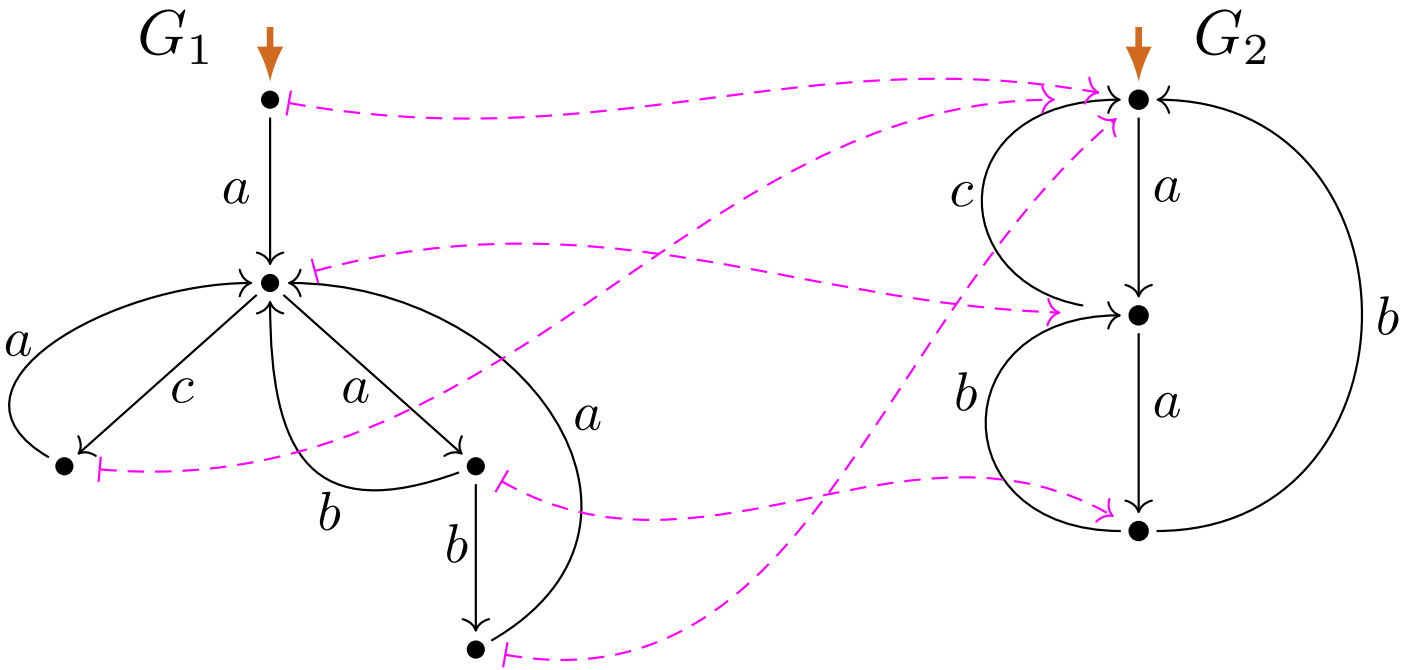
$$P(\overbrace{a \cdot (c \cdot a + a \cdot (b + b \cdot a))^* \cdot 0}^{f})$$

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



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

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



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

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

Process interpretation P (formally)

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

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

Process interpretation P (formally)

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

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

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

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 $\mathcal{T}$)

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

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

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

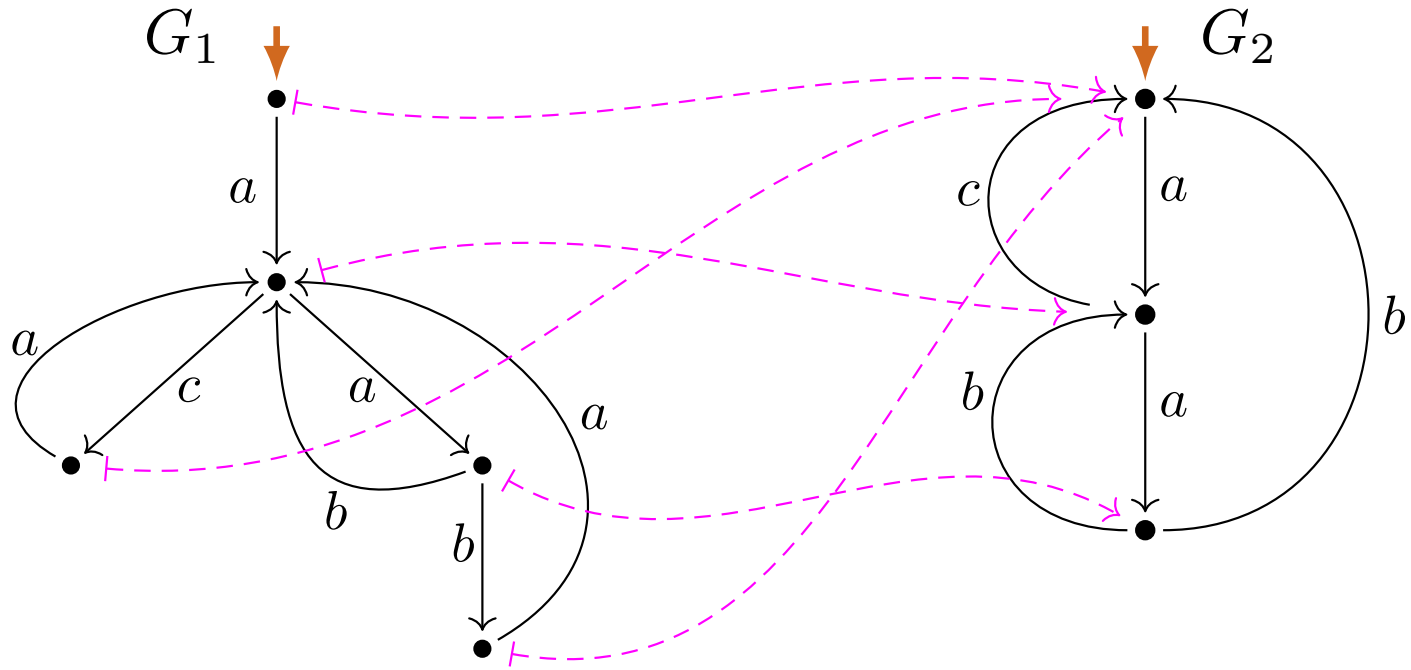
Definition

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: $[\![e]\!]_P := [P(e)]_{\underline{\leftrightarrow}}$.

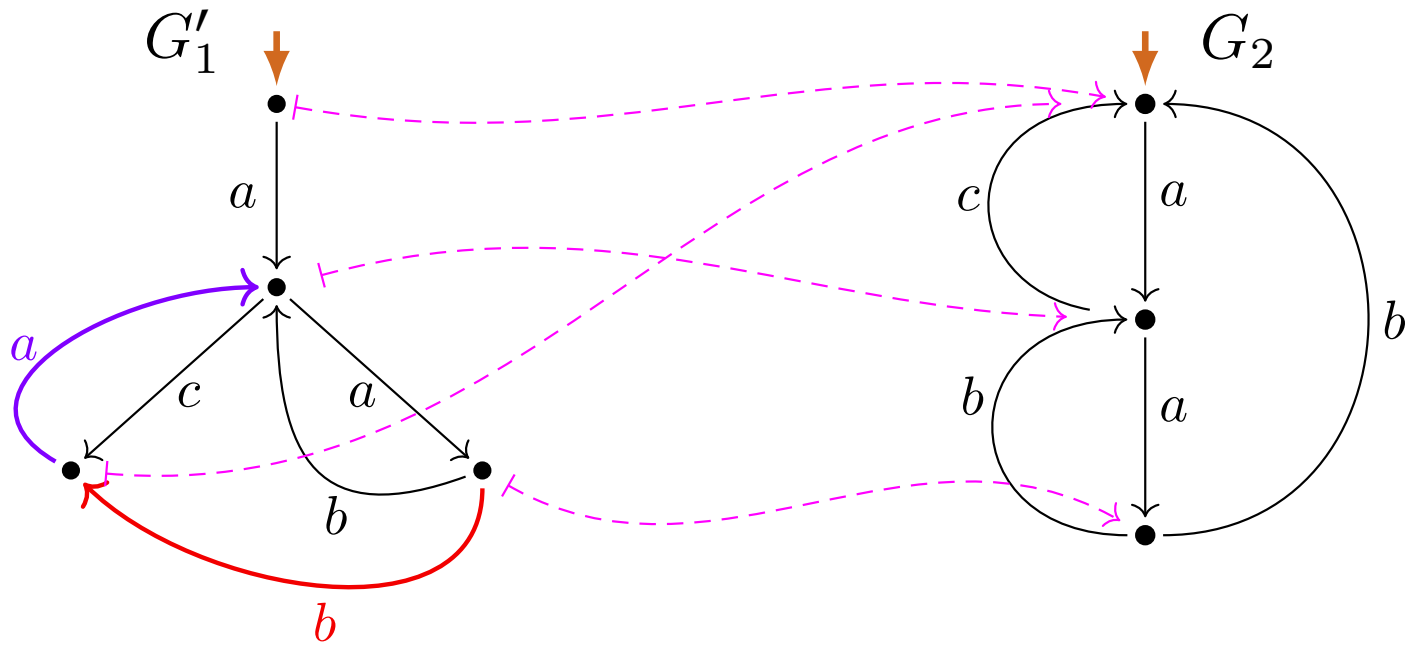
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, informally)



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

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

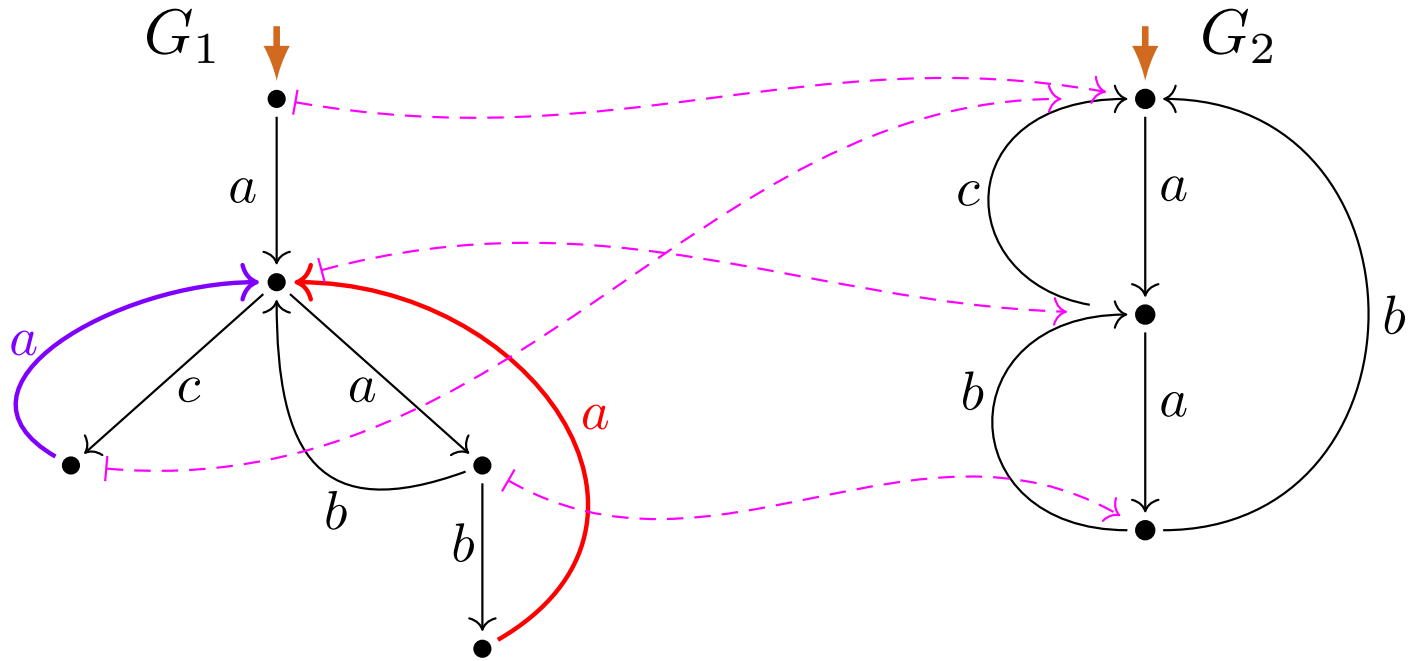
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, formally)



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

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

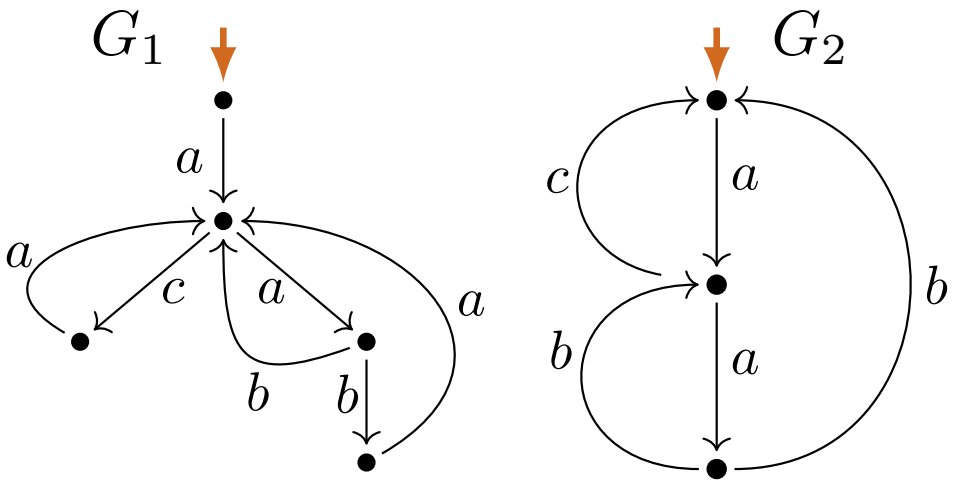
P-expressibility and $[\![\cdot]\!]_P$-expressibility (example, formally)



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

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

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

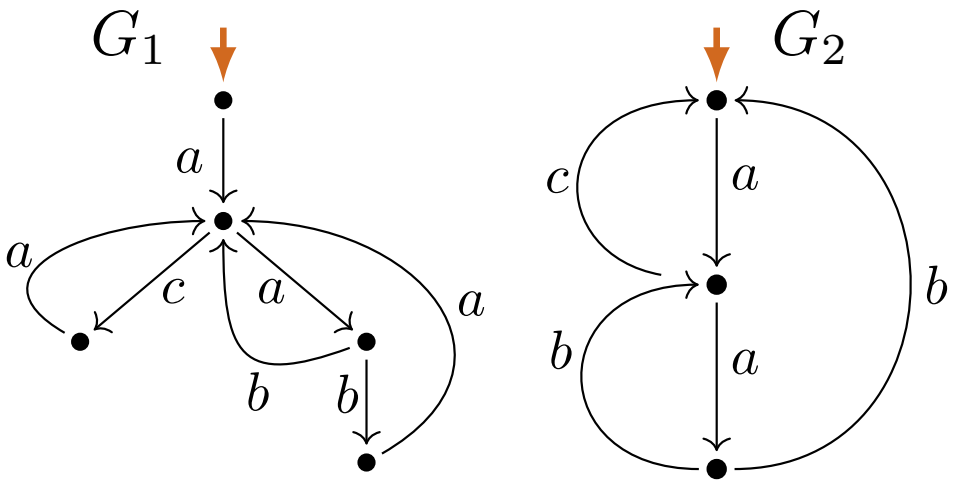


P-expressible

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

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

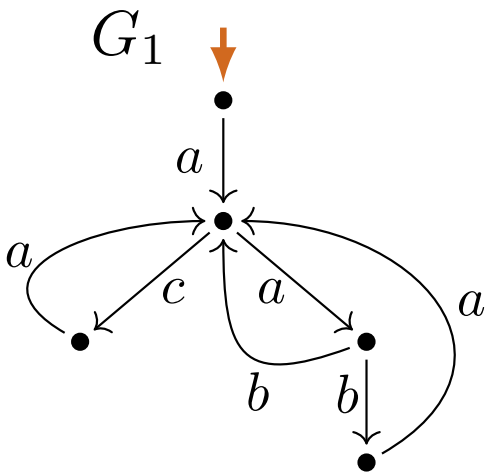
P-expressibility and $[\![\cdot]\!]_P$-expressibility (examples)

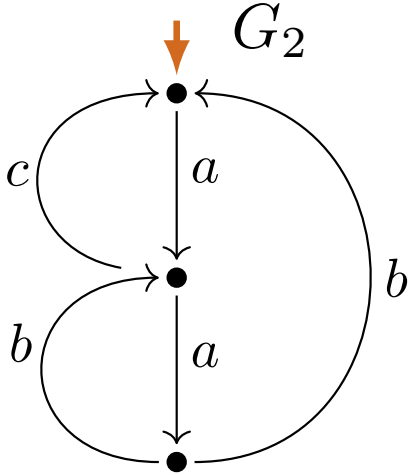


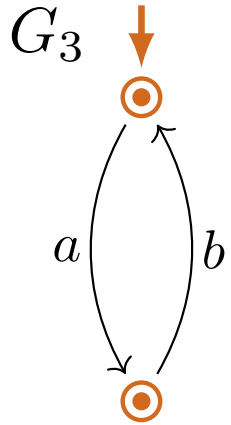
P-expressible ?

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

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







P-expressible

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

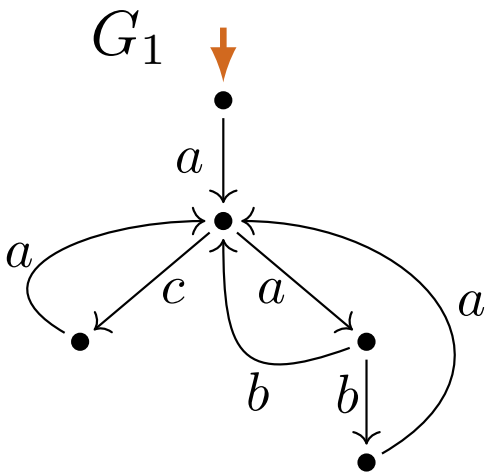
?

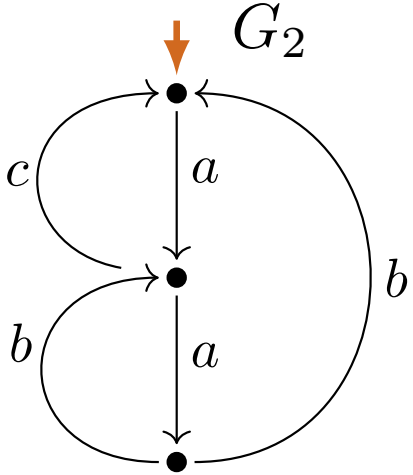
$[\![\cdot]\!]_P$-expressible

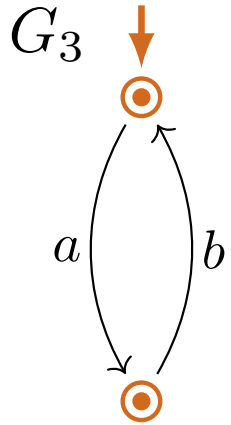
not P-expressible

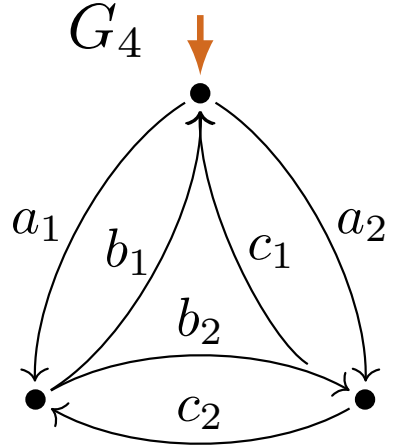
not $[\![\cdot]\!]_P$-expressible

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









P-expressible

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

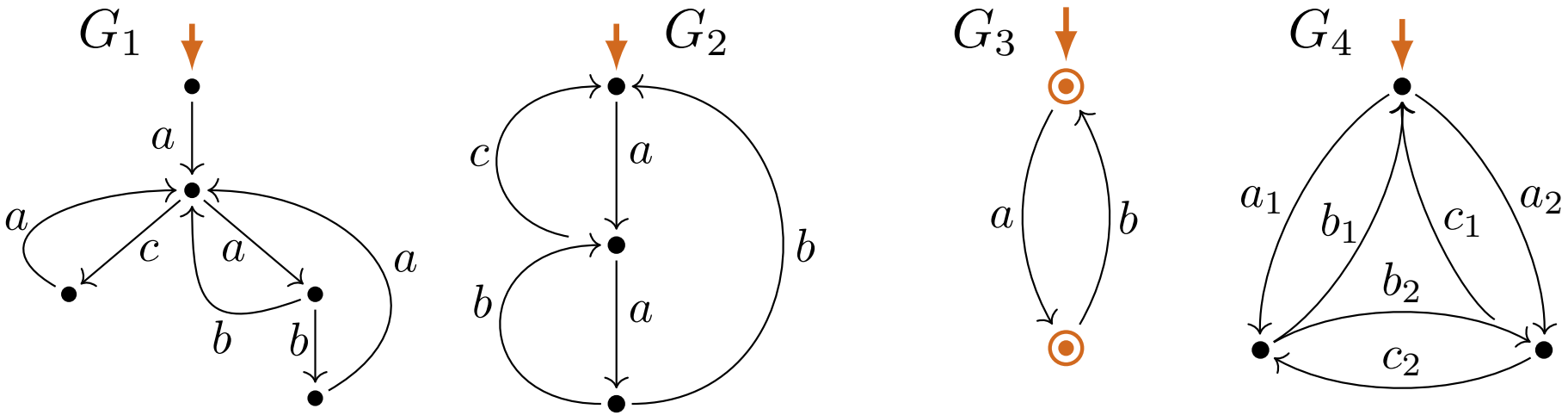
?

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

not P-expressible

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

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



P-expressible

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

?

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

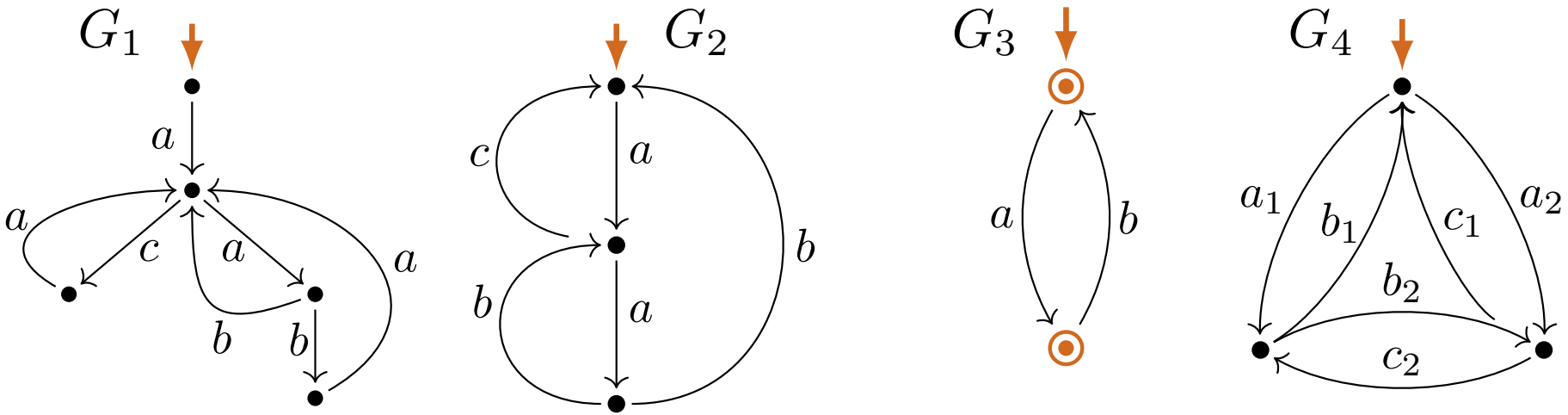
not P-expressible

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

Expressibility Questions:

How can P-expressibility and $[\![\cdot]\!]_P$-expressibility be characterised?

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



P-expressible

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

?

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

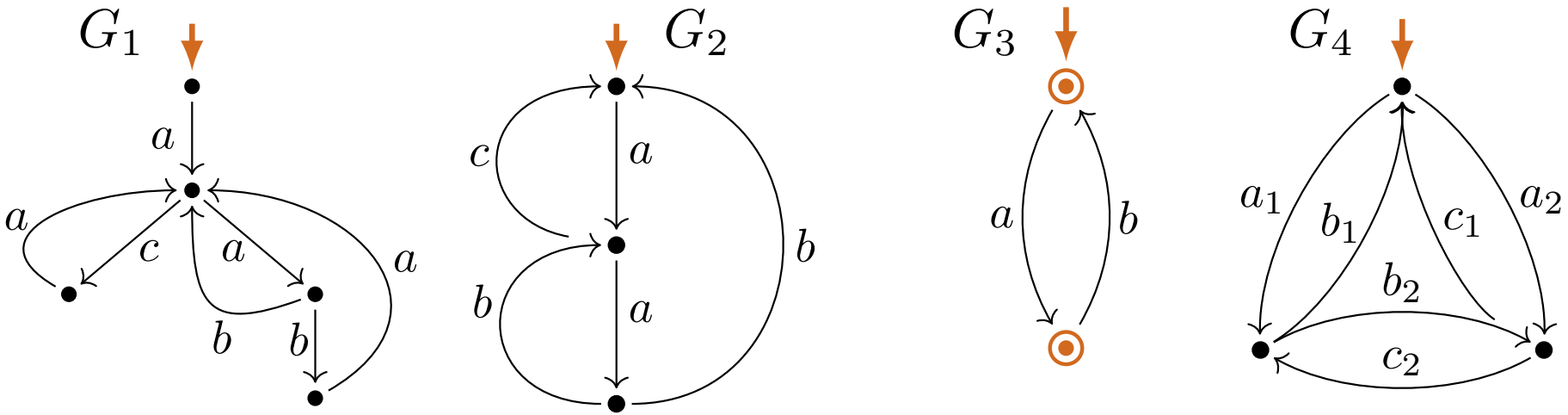
not P-expressible

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

Expressibility Questions:

What structural property of finite charts [process graphs]
is necessary and sufficient of star behaviour [$[\![\cdot]\!]_P$-expressibility] (Milner, 1984)

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



P-expressible

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

?

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

not P-expressible

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

Expressibility Questions:

How can P-expressibility and $[\![\cdot]\!]_P$-expressibility be characterised?

Overview

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

Small steps for hard questions

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

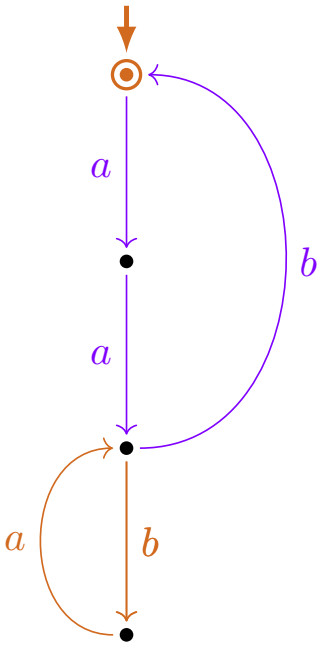
  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

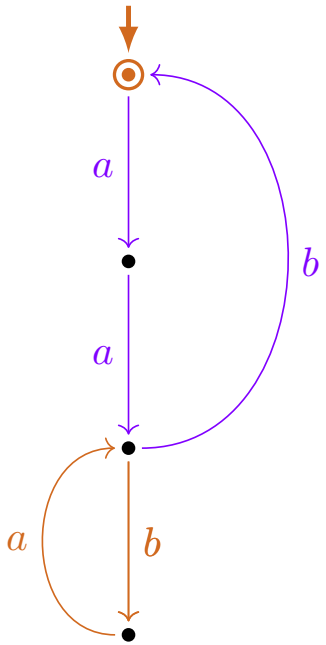
Well-behaved 1-graphs (with 1-transitions)

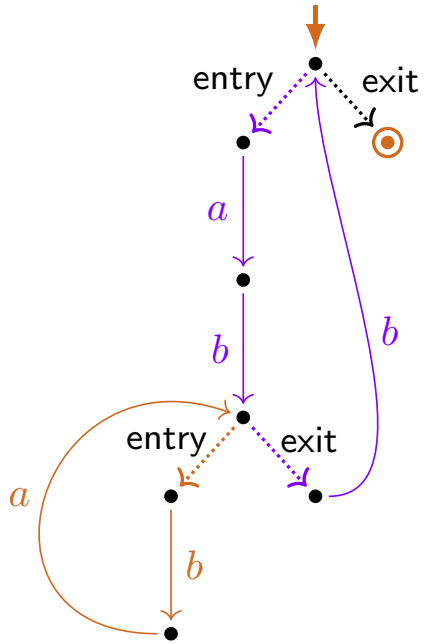


$\sim P((aa(ba)^*b)^*)$

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

Well-behaved 1-graphs (with 1-transitions)





well-behaved

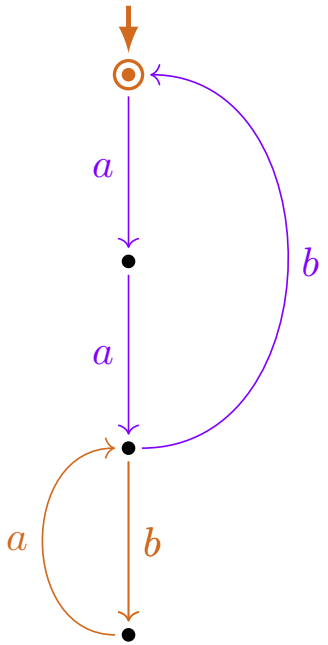
loop-entry/-exit palm tree

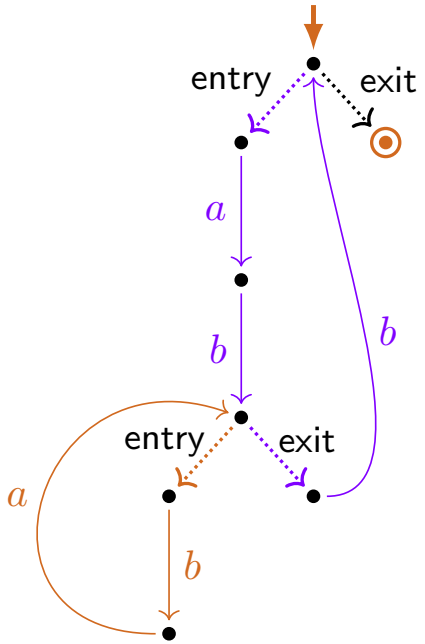
(Corradini, Baeten)

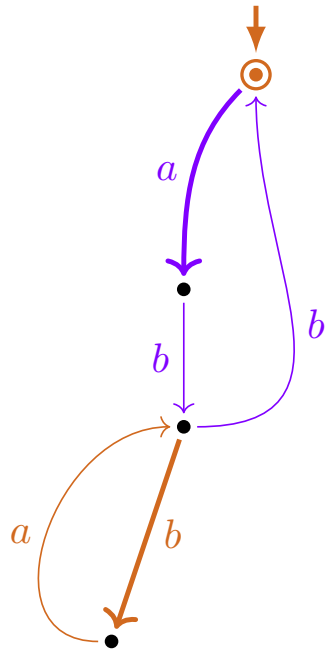
$\sim P((aa(ba)^*b)^*)$ $\sim P((1{\cdot}aa(1{\cdot}ba)^*1{\cdot}b)^*(1{\cdot}1))$

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

Well-behaved 1-graphs (with 1-transitions)







well-behaved

loop-entry/-exit palm tree

(Corradini, Baeten)

$\sim P((aa(ba)^*b)^*)$

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

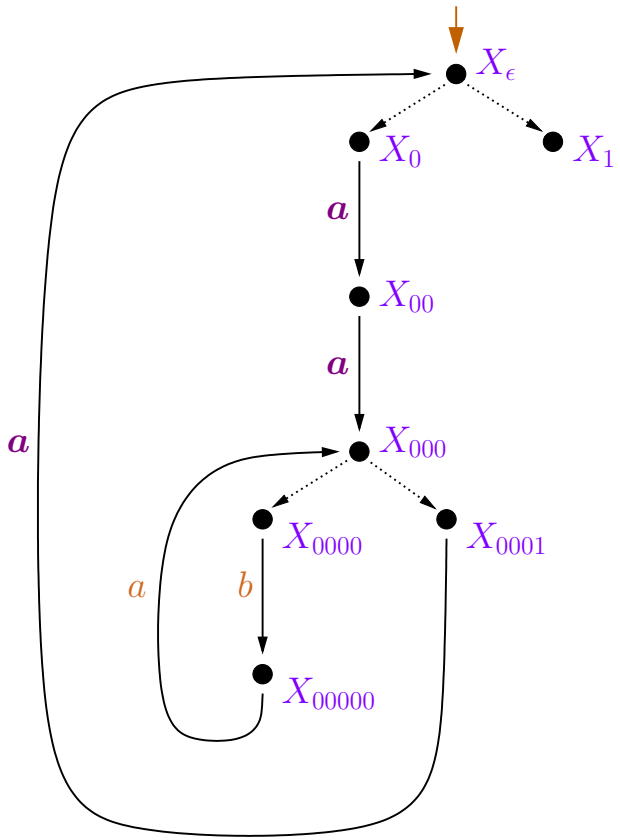
$\sim P((1{\cdot}aa(1{\cdot}ba)^*1{\cdot}b)^*(1{\cdot}1))$

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

$\sim P((aa(ba)^*b)^*)$

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

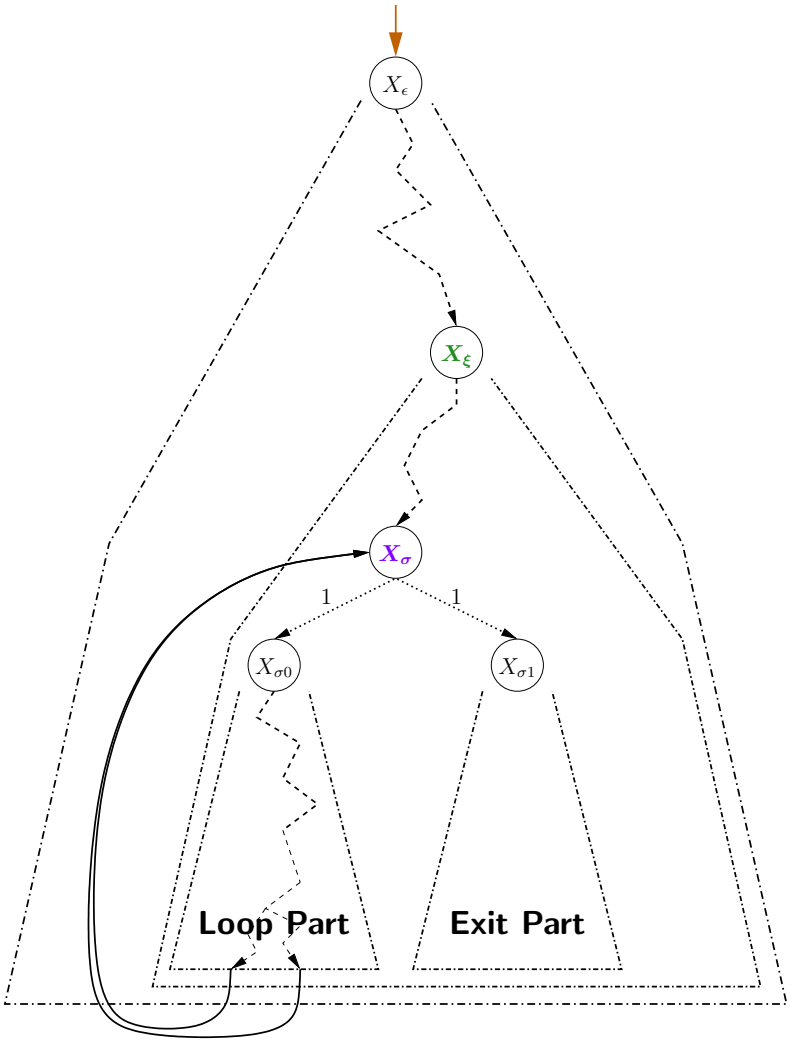
Well-Behaved specification (example)



$$X_\epsilon = 1 \cdot X_0 + 1 \cdot X_1$$
$$X_0 = \boldsymbol{a} \cdot X_{00}$$
$$X_{00} = \boldsymbol{a} \cdot X_{000}$$
$$X_{000} = 1 \cdot X_{0000} + 1 \cdot X_{0001}$$
$$X_{0000} = b \cdot X_{00000}$$
$$X_{00000} = a \cdot X_{000}$$
$$X_{0001} = \boldsymbol{a} \cdot X_\epsilon$$
$$X_1 = 0$$

$$P((\boldsymbol{a}\boldsymbol{a}(ba)^*\boldsymbol{a})^*.0)$$

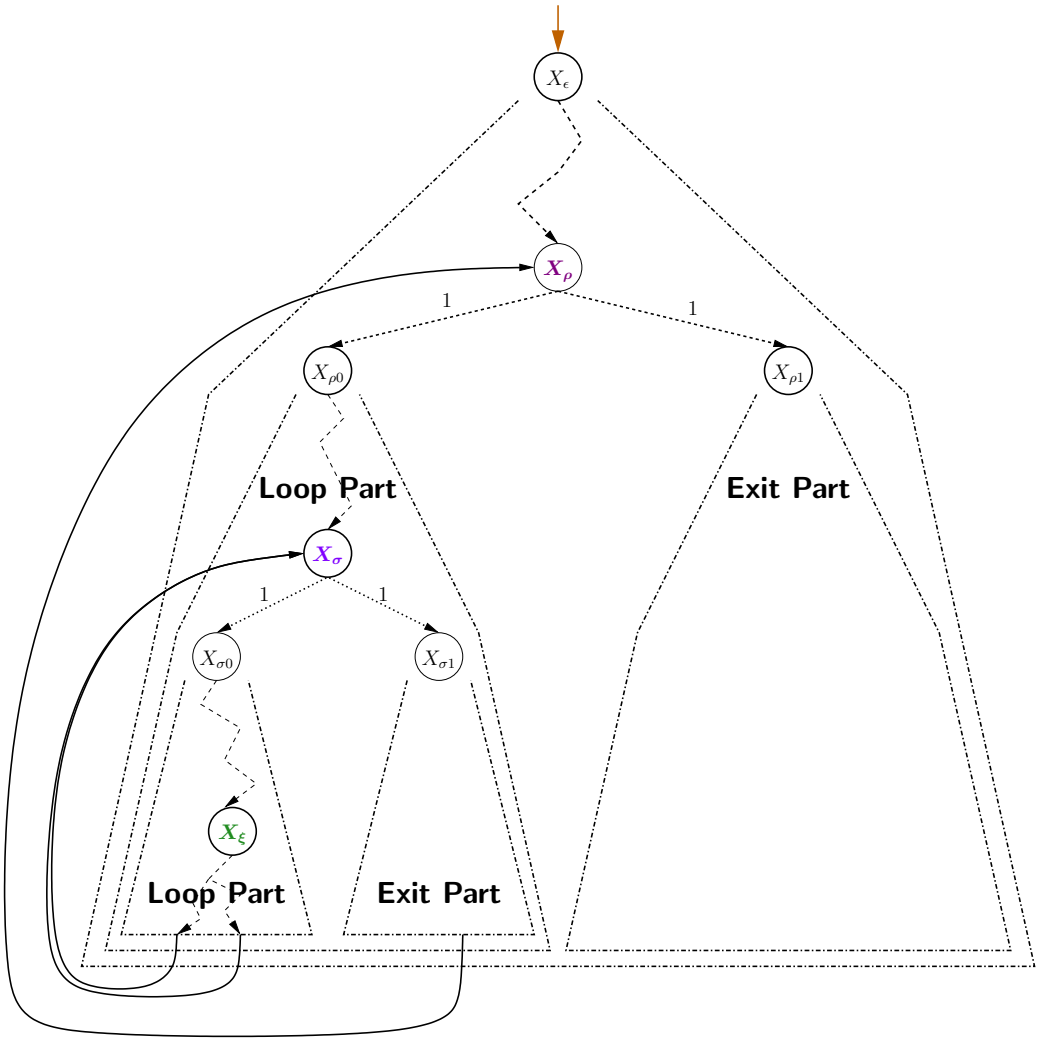
Well-Behaved specifications (well-beh./cycling variables)



X_ξ, X_ϵ ... *well-behaved variables*
(X_ξ "does not return" to a recursion variable above itself)

X_σ is a *cycling variable*
(some recursion variable below X_σ "returns to" X_σ)

Well-Behaved specifications (cycles back to)



X_σ, X_ρ ... cycling variables

X_ξ *cycles back to* X_σ

(The nearest return of X_ξ to a rec.var. above is to X_σ)

X_σ *cycles back to* X_ρ

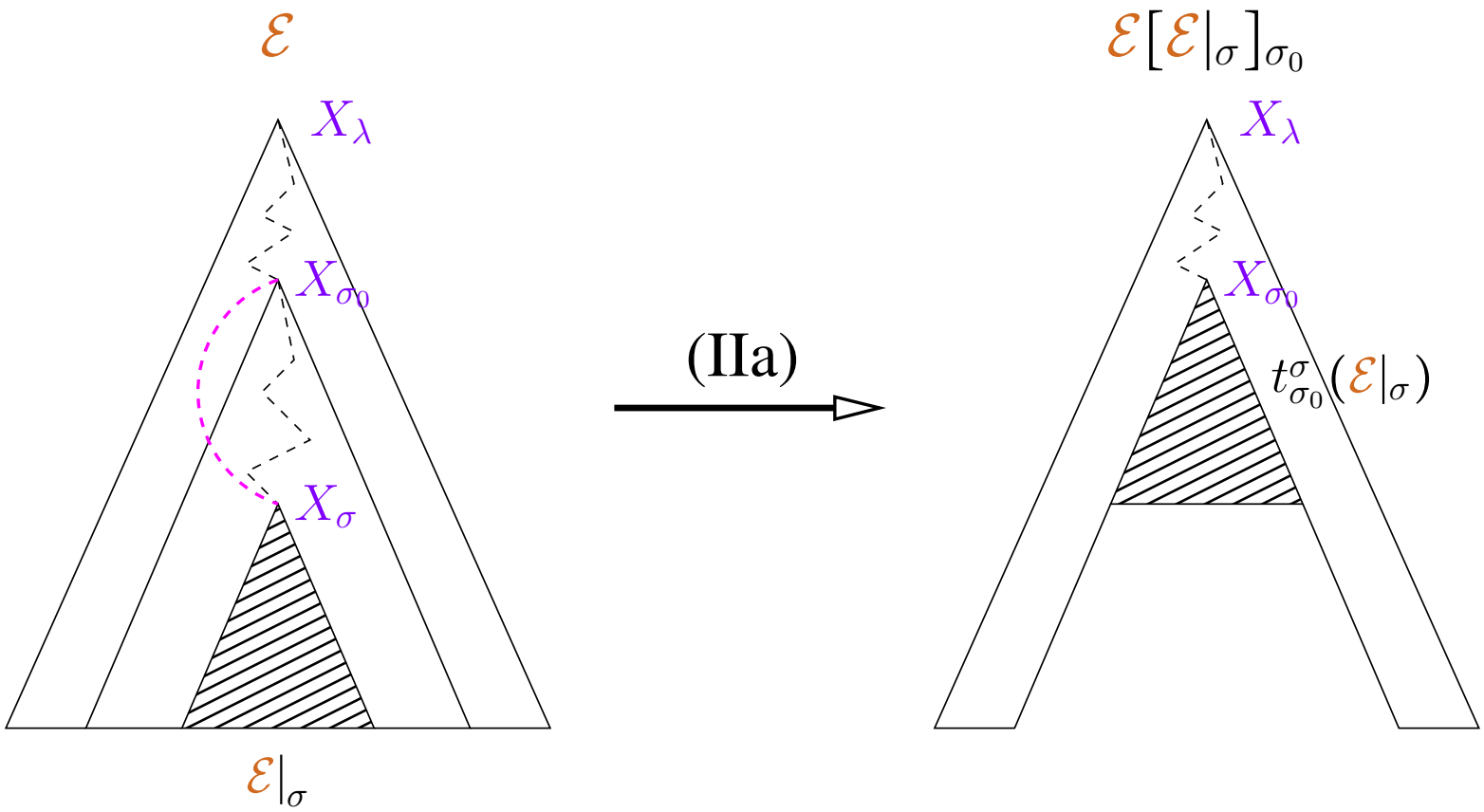
$[\![\cdot]\!]_P$-expressible = bisimilar to well-behaved

Theorem (Baeten/Corradini, 2005)

(i) *A graph* G *is* $[\![\cdot]\!]_P$*-expressible*
$\iff$ G *is the solution of a well-behaved specification.*

(ii) *A graph* G *is* $[\![\cdot]\!]_P$*-expressible*
$\iff$ *there is a well-behaved* 1*-graph* $\underline{G_{wb}}$ *such that* $G \leftrightarrows G_{wb}$.

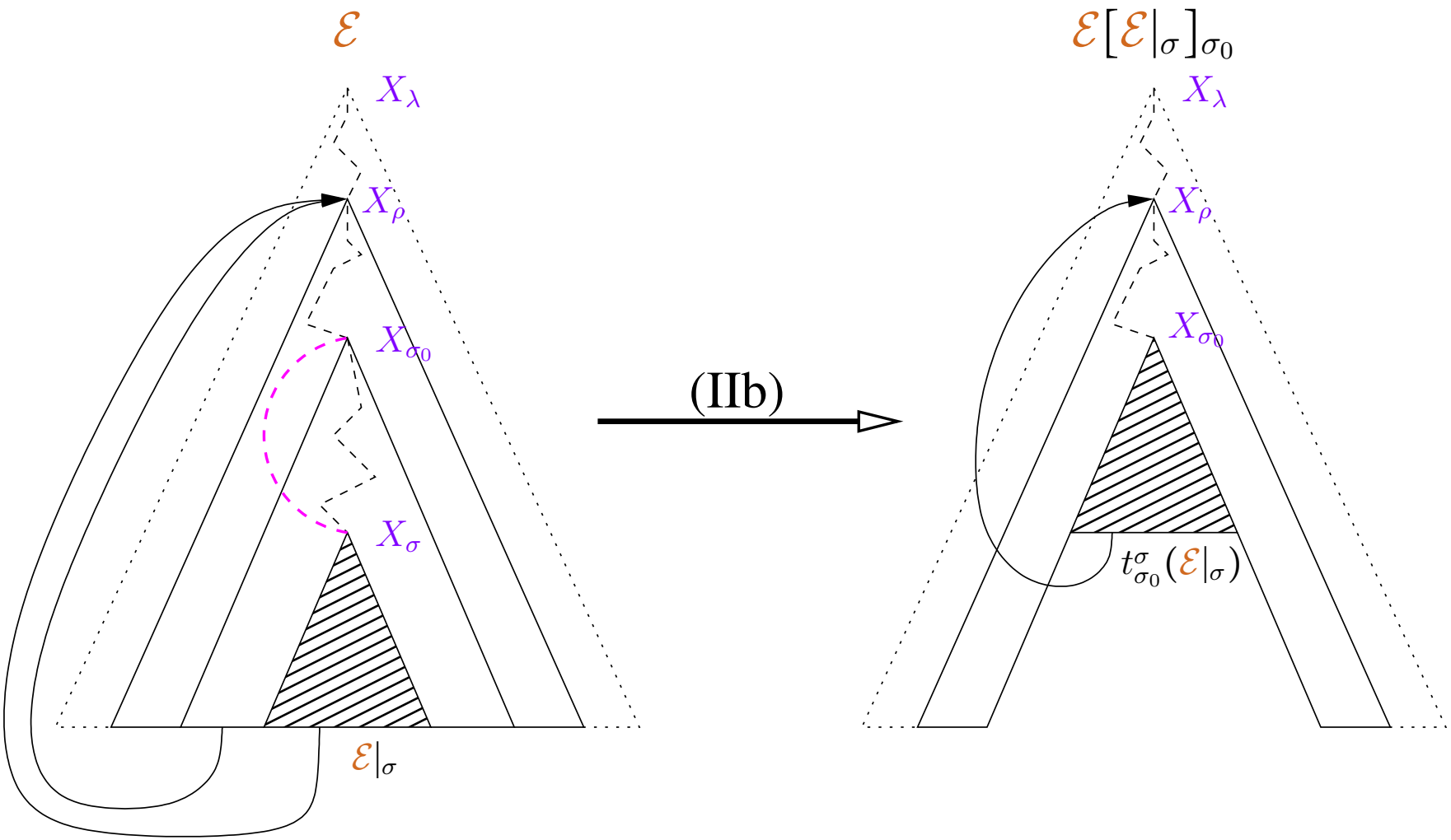
Reduction steps (IIa) for well-behaved 1-graphs



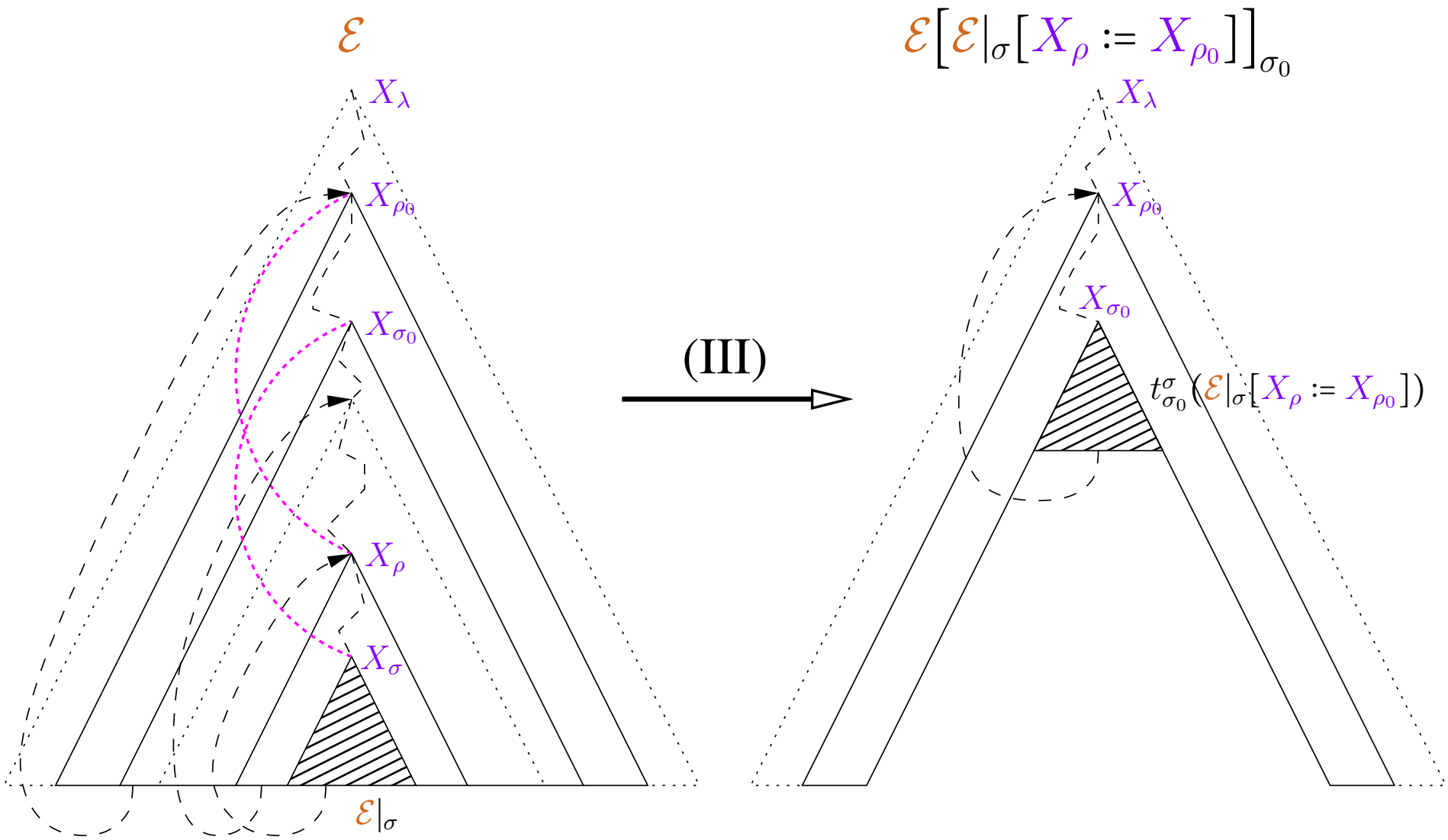
$\langle X_\sigma \mid \mathcal{E} \rangle \leftrightarrows \langle X_{\sigma_0} \mid \mathcal{E} \rangle$

X_σ, X_{σ_0} are well-behaved

Reduction steps (IIb) for well-behaved 1-graphs



Reduction steps (III) for well-behaved 1-graphs



Result facilitated by steps (st1)

Theorem (*Baeten–Corradini–G, 2005*)

$[\![\cdot]\!]_P$-*Expressibility of graphs is decidable (at least double-exponentially).*

Result facilitated by steps (st1)

Reduction Lemma for well-behaved 1-graphs *(G, 2005)*

Let G be graph with n states and max. out-degree k. Suppose that $\underline{G}_{wb}$ is a well-behaved 1-graph with $\underline{G}_{wb} \leftrightarrows G$ ($\Rightarrow G, \underline{G}_{wb}$ are $[\![\cdot]\!]_P$-expressible).

Then $\underline{G}_{wb}$ can be reduced under bisimilarity by reduction steps *(I)–(III)* to a well-behaved 1-graph $\underline{G}_{wb}^{(0)}$ with:

- $\underline{G}_{wb}^{(0)} \leftrightarrows \underline{G}_{wb} (\leftrightarrows G)$,
- depth $\leq (n+1)^3 \cdot 2^{3k}$, and max. out-degree $\leq k$.

Theorem (*Baeten–Corradini–G, 2005*)

$[\![\cdot]\!]_P$-Expressibility of graphs is decidable (at least double-exponentially).

Result facilitated by steps (st1)

Reduction Lemma for well-behaved 1-graphs *(G, 2005)*

Let G be graph with n states and max. out-degree k. Suppose that $\underline{G}_{wb}$ is a well-behaved 1-graph with $\underline{G}_{wb} \leftrightarrows G$ ($\Rightarrow G, \underline{G}_{wb}$ are $[\![\cdot]\!]_P$-expressible).

Then $\underline{G}_{wb}$ can be reduced under bisimilarity by reduction steps *(I)–(III)* to a well-behaved 1-graph $\underline{G}^{(0)}_{wb}$ with:

- $\underline{G}^{(0)}_{wb} \leftrightarrows \underline{G}_{wb} (\leftrightarrows G)$,
- depth $\leq (n+1)^3 \cdot 2^{3k}$, and max. out-degree $\leq k$.

Effective Expressibility Lemma *(G, 2005)*

A graph G with n states and max. out-degree k is $[\![\cdot]\!]_P$-expressible

$\iff G \leftrightarrows \underline{G}^{(0)}_{wb}$ for a well-behaved 1-graph $\underline{G}^{(0)}_{wb}$ with depth $\leq (n+1)^3 \cdot 2^{3k}$ and max. out-degree $\leq k$.

Theorem (*Baeten–Corradini–G, 2005*)

$[\![\cdot]\!]_P$-Expressibility of graphs is decidable (at least double-exponentially).

Small steps for hard questions

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

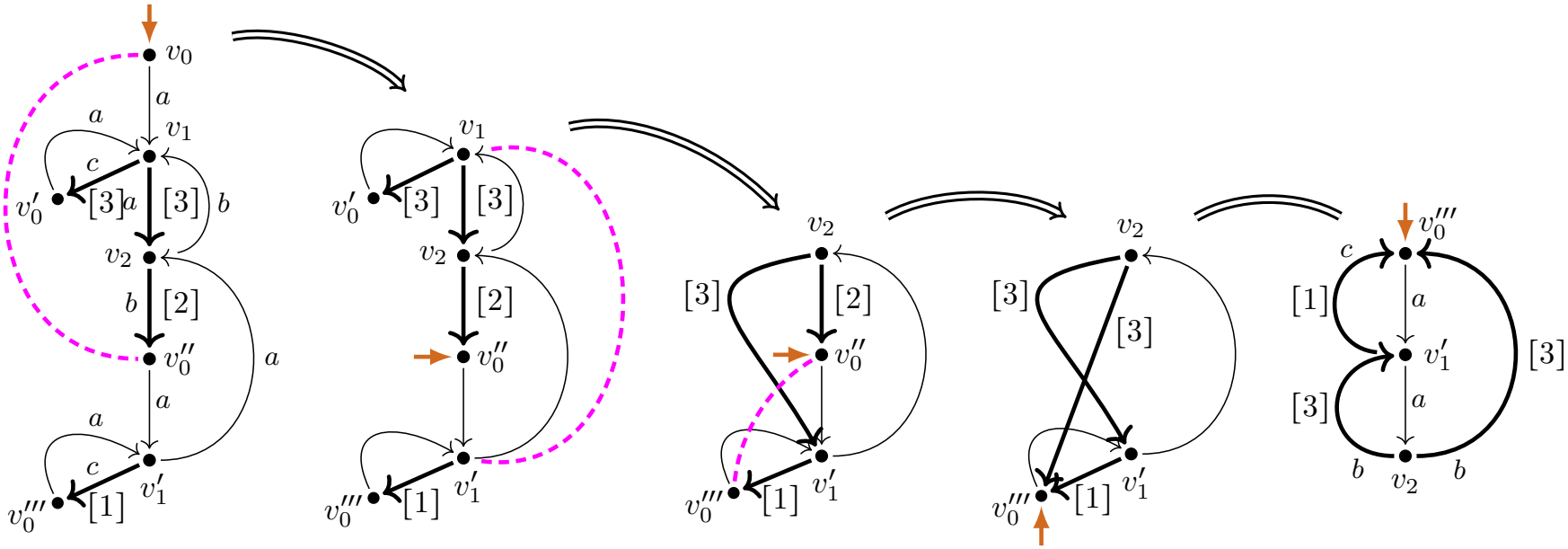
  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

Collapse steps for LLEE graphs under bisimilarity



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) Immediate termination is only possible at the start vertex.

Loop graphs (interpretations of innermost iterations without 1)

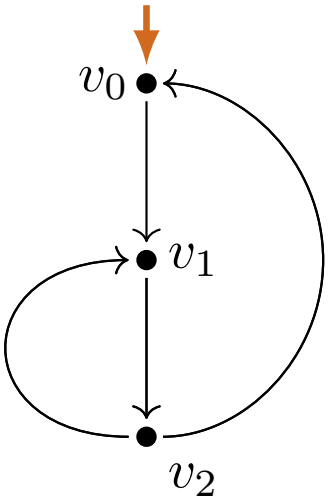
Definition

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) Immediate termination is only possible at the start vertex.



Loop graphs (interpretations of innermost iterations without 1)

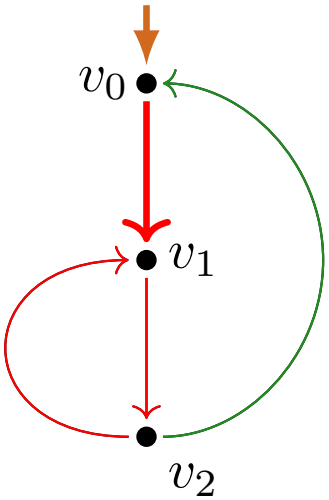
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

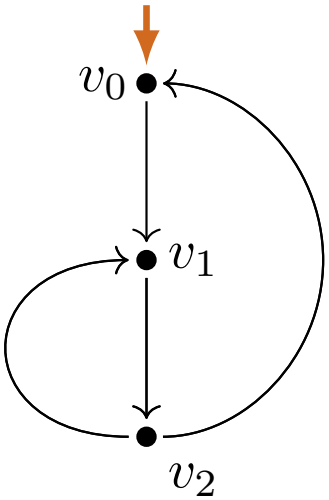
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

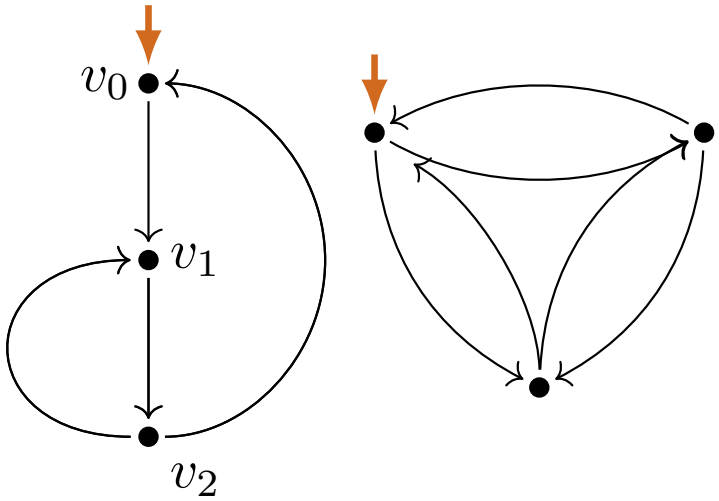
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

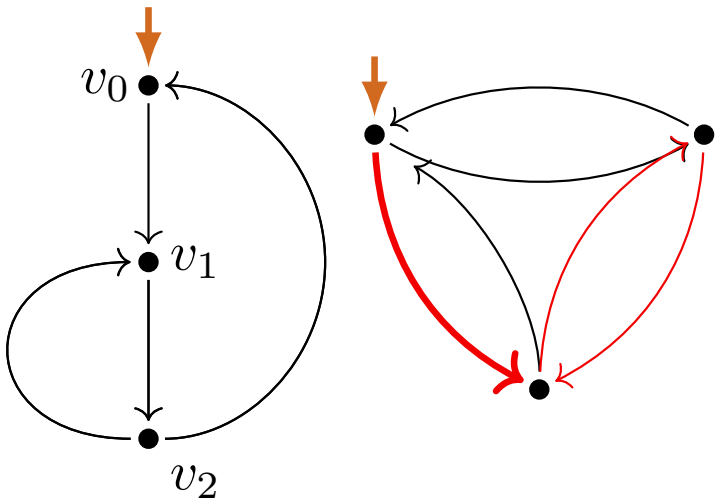
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

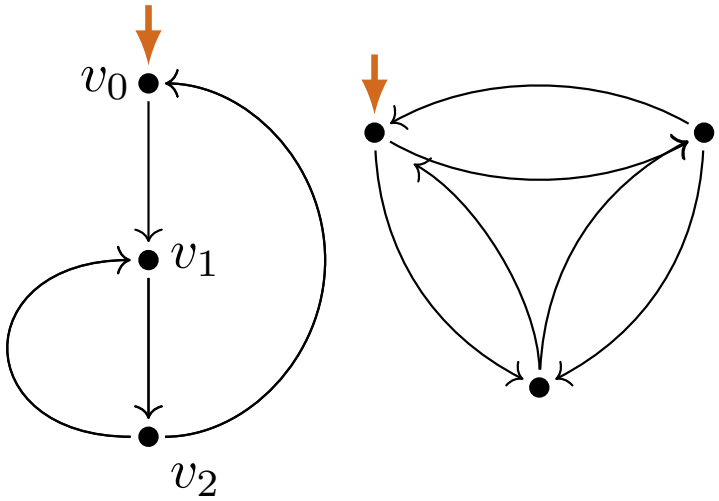
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

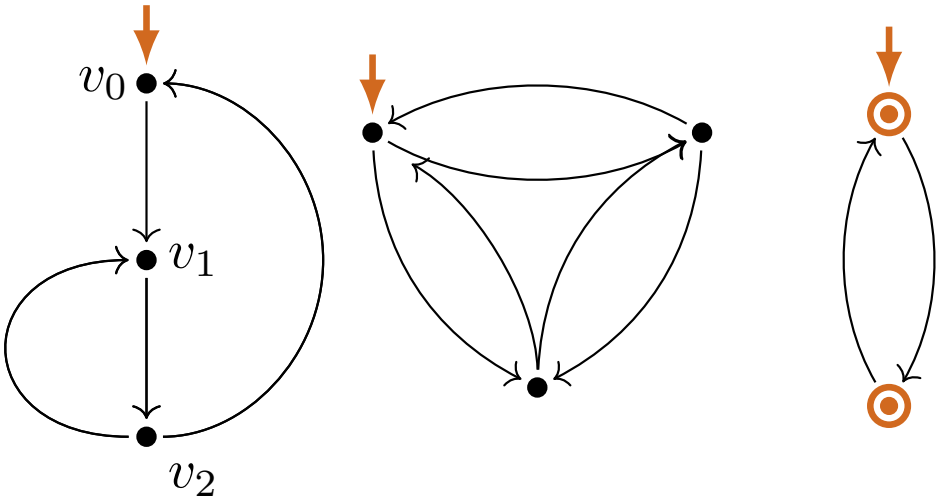
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

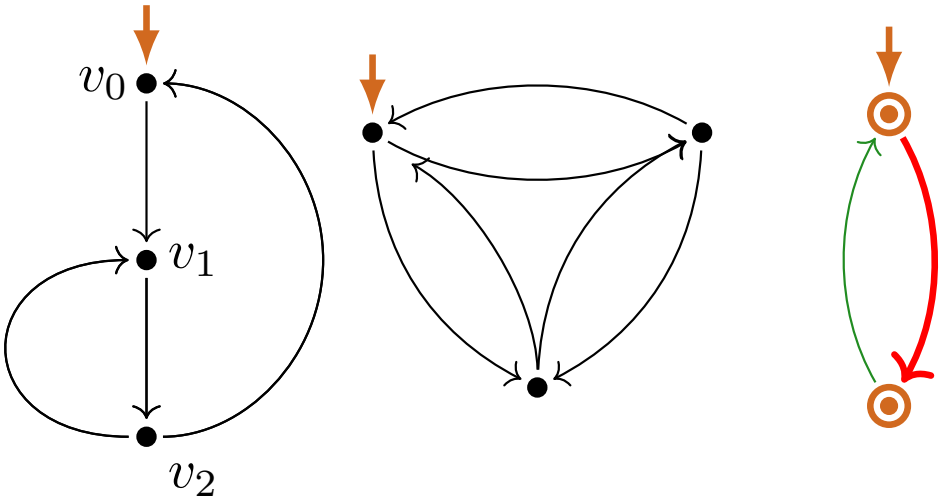
Definition

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) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

Loop graphs (interpretations of innermost iterations without 1)

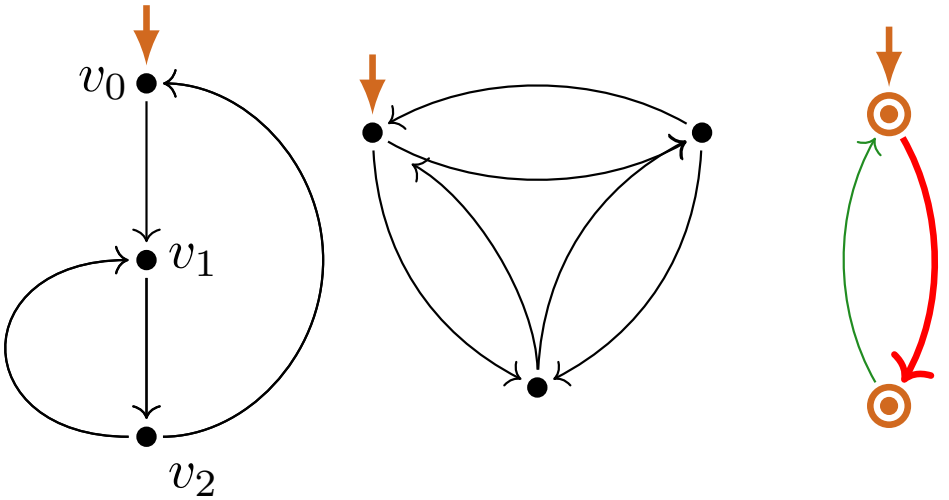
Definition

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) Immediate termination is only possible at the start vertex.



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

Loop graphs (interpretations of innermost iterations without 1)

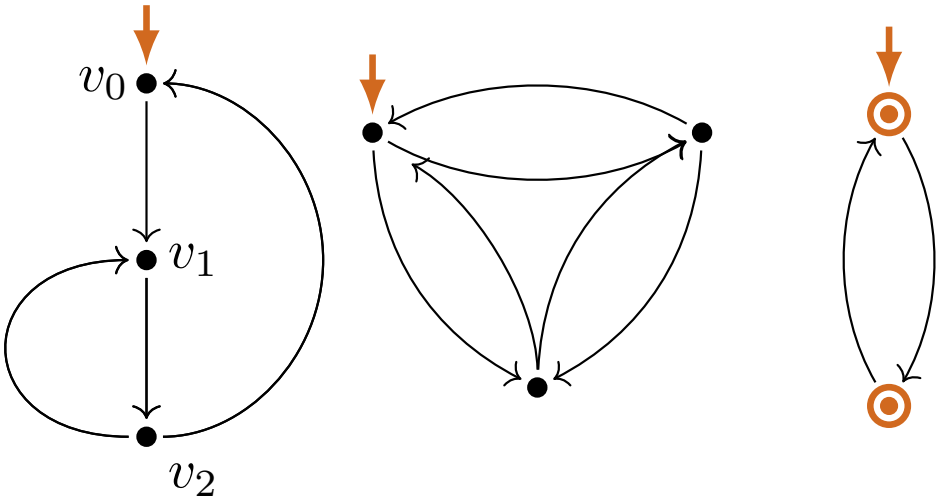
Definition

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) Immediate termination is only possible at the start vertex.



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

Loop graphs (interpretations of innermost iterations without 1)

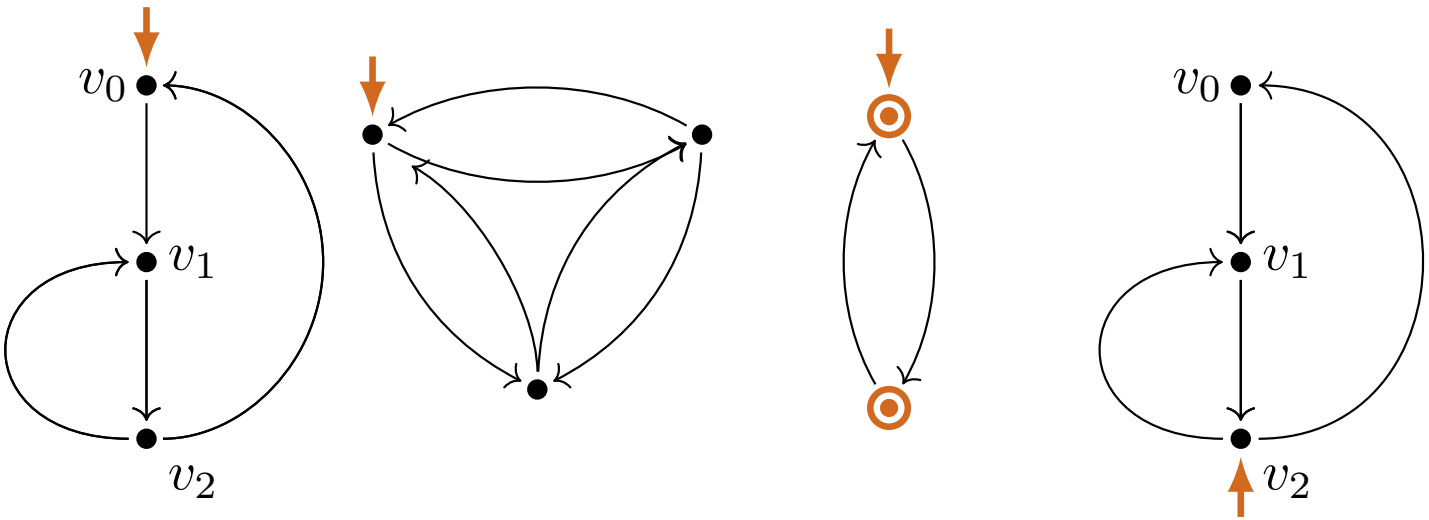
Definition

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) Immediate termination is only possible at the start vertex.



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

Loop graphs (interpretations of innermost iterations without 1)

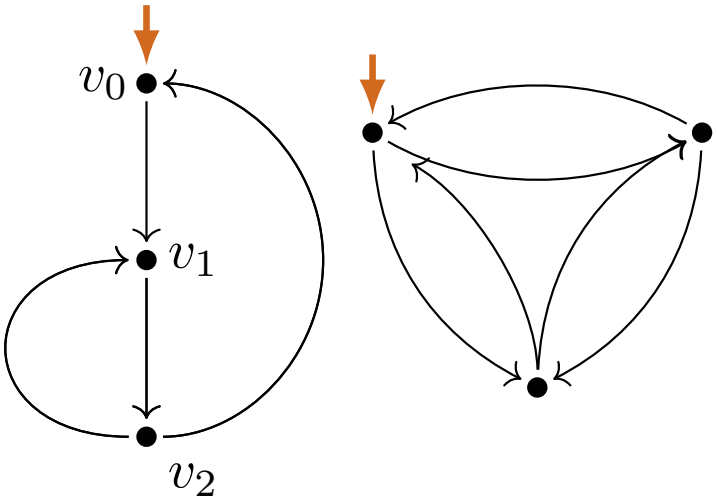
Definition

A process graph is a loop graph if:

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

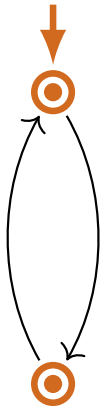
(L2) Every infinite path from the start vertex returns to it.

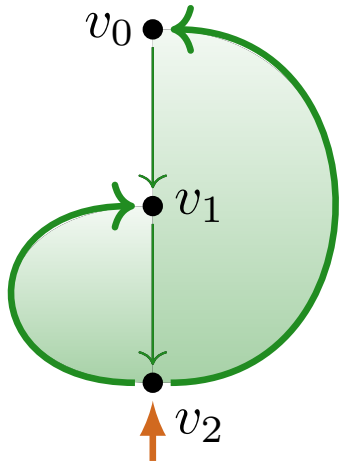
(L3) Immediate termination is only possible at the start vertex.



(L1),~~(L2)~~

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





Loop graphs (interpretations of innermost iterations without 1)

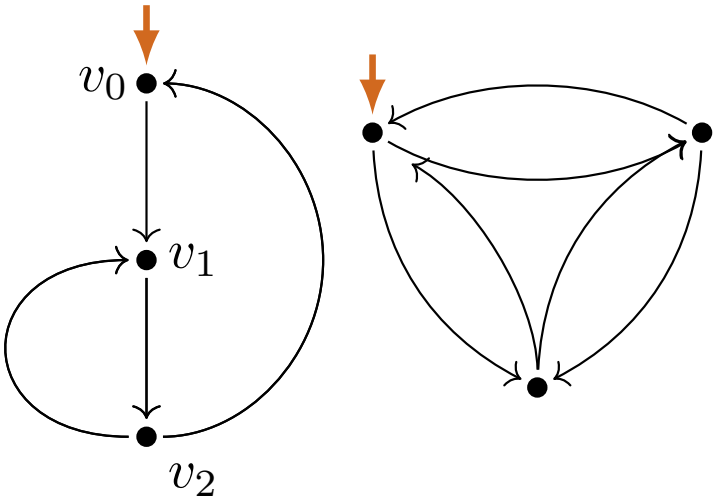
Definition

A process graph is a loop graph if:

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

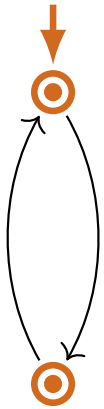
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

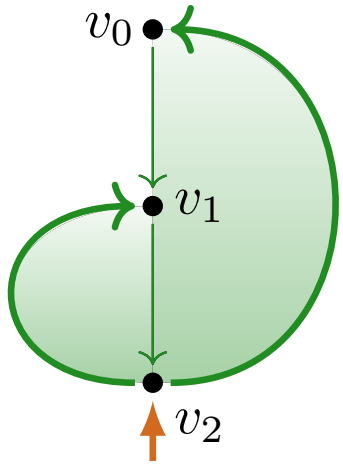


(L1),~~(L2)~~

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



loop chart



Loop graphs (interpretations of innermost iterations without 1)

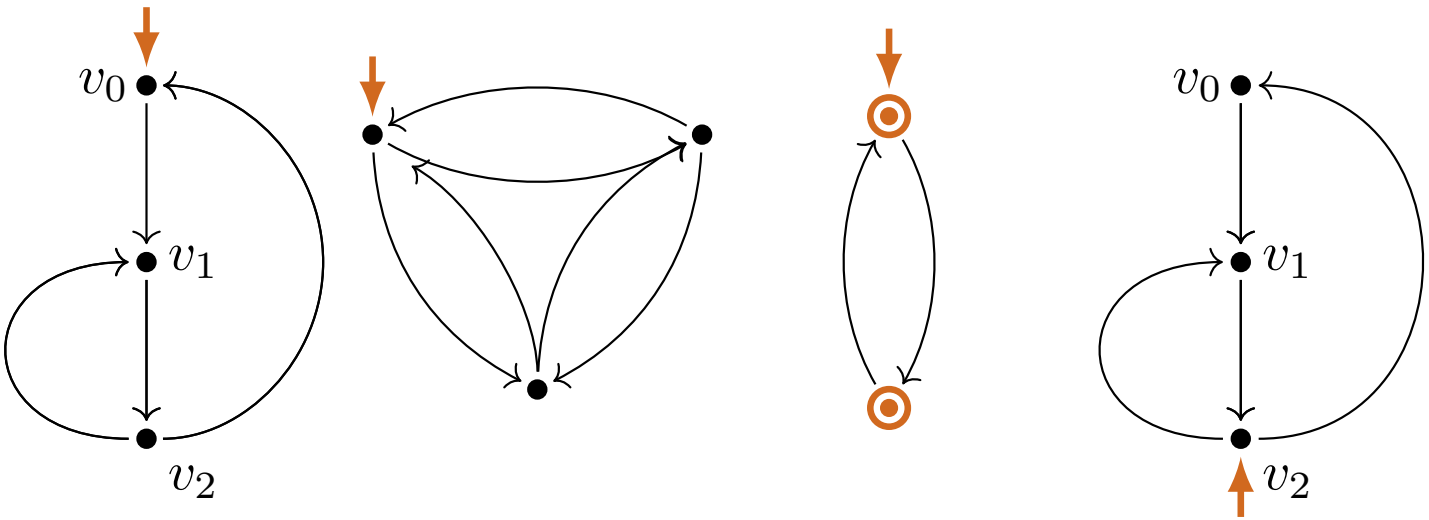
Definition

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) Immediate termination is only possible at the start vertex.



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

Loop graphs (interpretations of innermost iterations without 1)

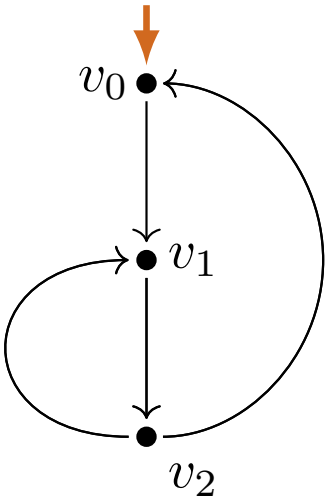
Definition

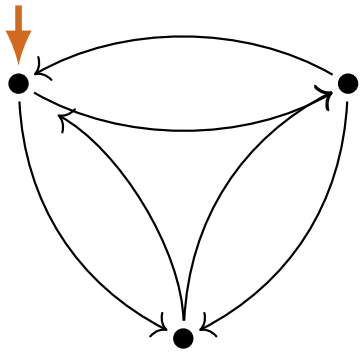
A process graph is a loop graph if:

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

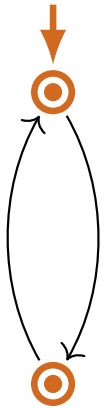
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

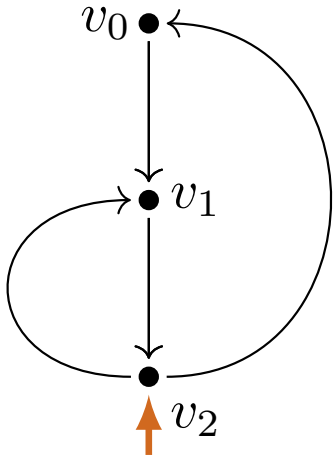


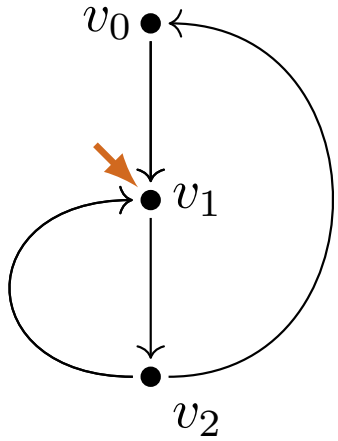


(L1),~~(L2)~~



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





loop chart

Loop graphs (interpretations of innermost iterations without 1)

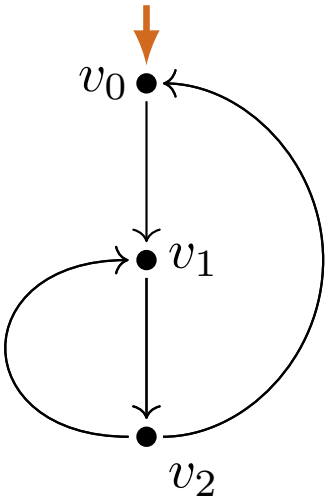
Definition

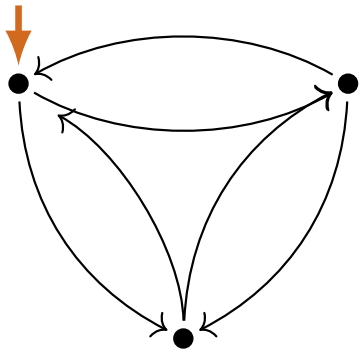
A process graph is a loop graph if:

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

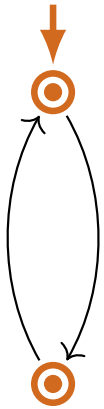
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

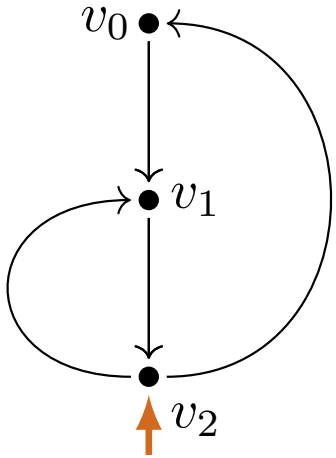




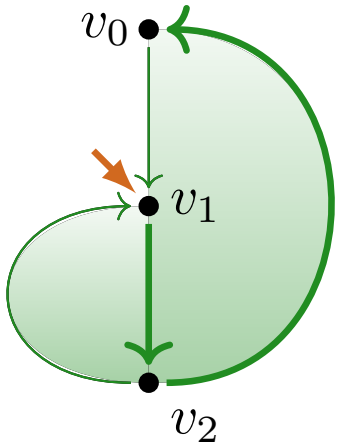
(L1),~~(L2)~~



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



loop chart



Loop graphs (interpretations of innermost iterations without 1)

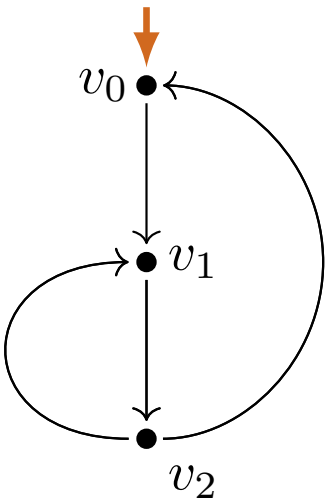
Definition

A process graph is a loop graph if:

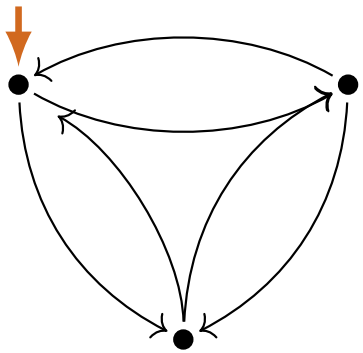
(L1) There is an infinite path from the start vertex.

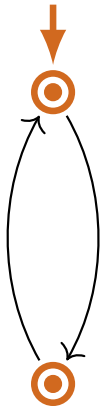
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

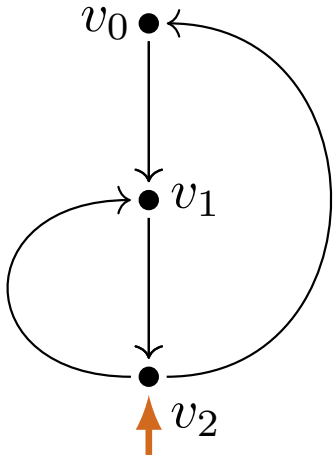


(L1),~~(L2)~~

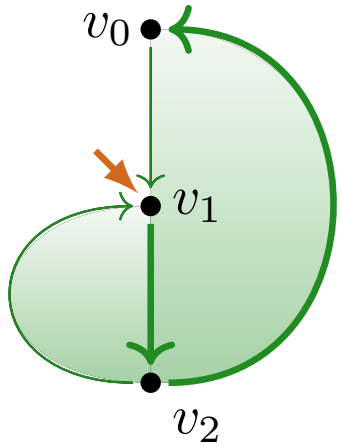




(L1),(L2),~~(L3)~~



loop chart



loop chart

Loop graphs (interpretations of innermost iterations without 1)

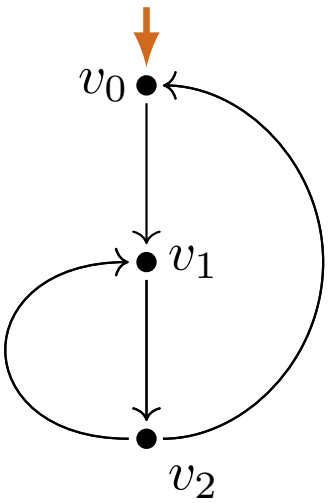
Definition

A process graph is a loop graph if:

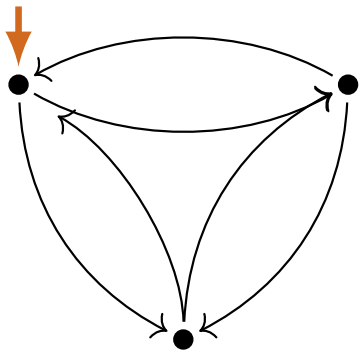
(L1) There is an infinite path from the start vertex.

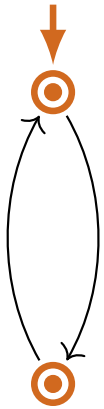
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

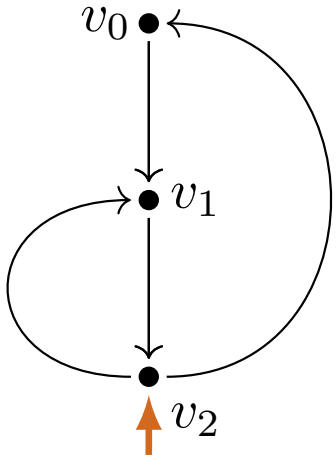


(L1),~~(L2)~~

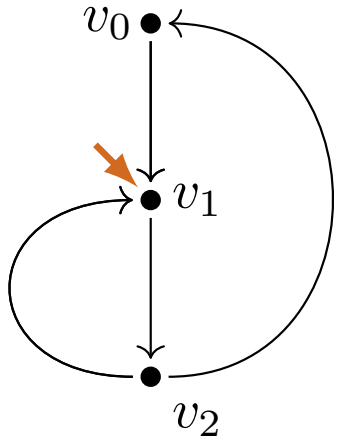




(L1),(L2),~~(L3)~~



loop chart



loop chart

Loop graphs (interpretations of innermost iterations without 1)

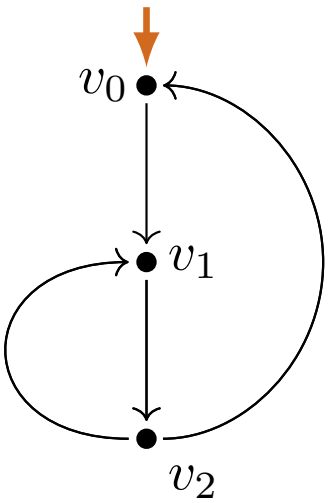
Definition

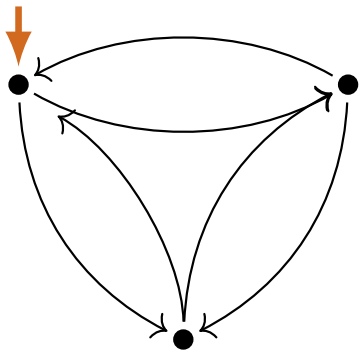
A process graph is a loop graph if:

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

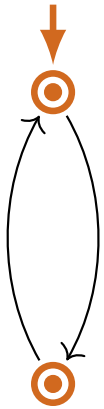
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

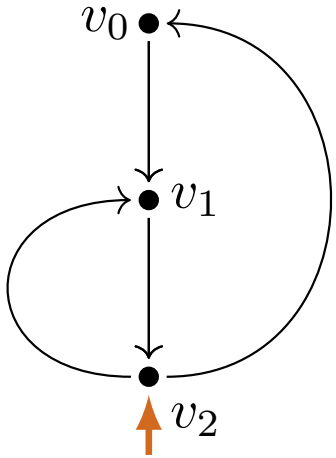




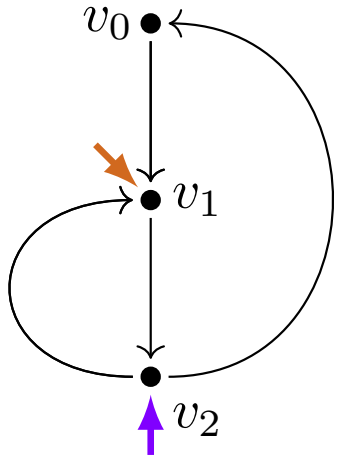
(L1),~~(L2)~~



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



loop chart



Loop graphs (interpretations of innermost iterations without 1)

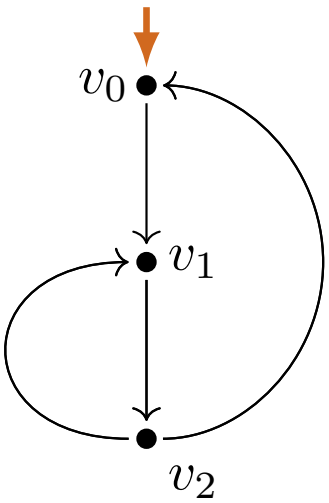
Definition

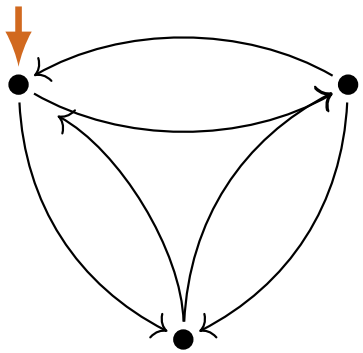
A process graph is a loop graph if:

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

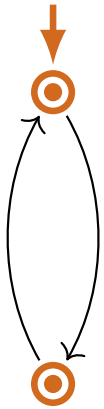
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

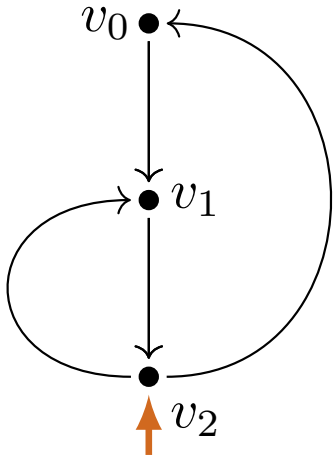




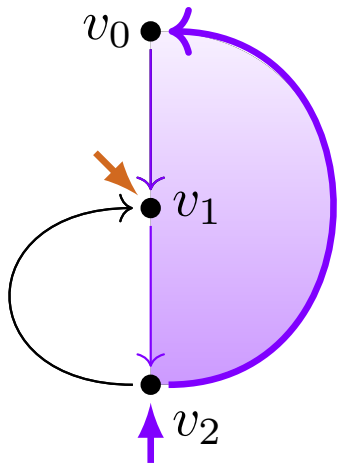
(L1),~~(L2)~~



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



loop chart



Loop graphs (interpretations of innermost iterations without 1)

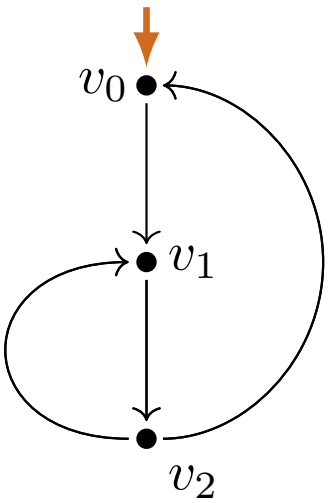
Definition

A process graph is a loop graph if:

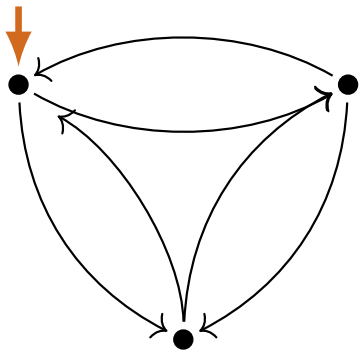
(L1) There is an infinite path from the start vertex.

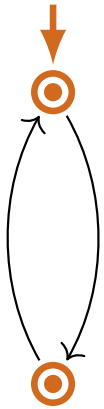
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

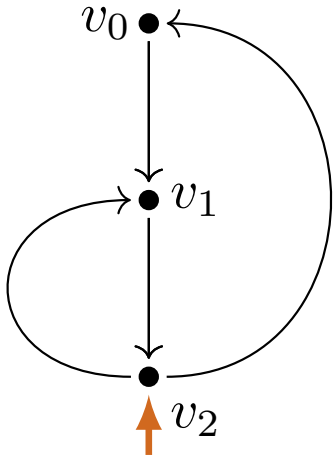


(L1),~~(L2)~~

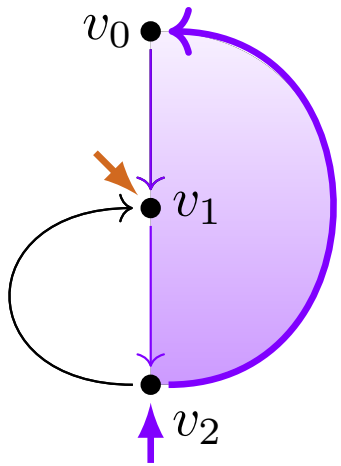




(L1),(L2),~~(L3)~~



loop chart



loop subchart

Loop graphs (interpretations of innermost iterations without 1)

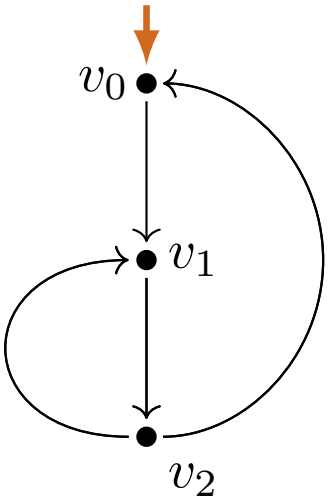
Definition

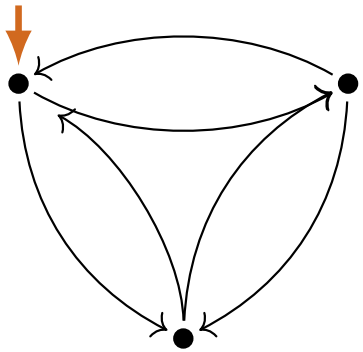
A process graph is a loop graph if:

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

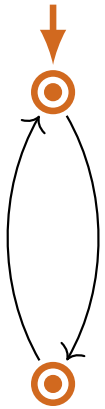
(L2) Every infinite path from the start vertex returns to it.

(L3) Immediate termination is only possible at the start vertex.

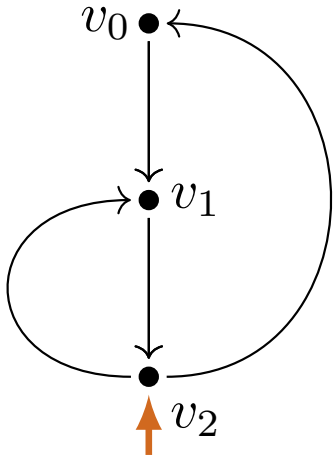


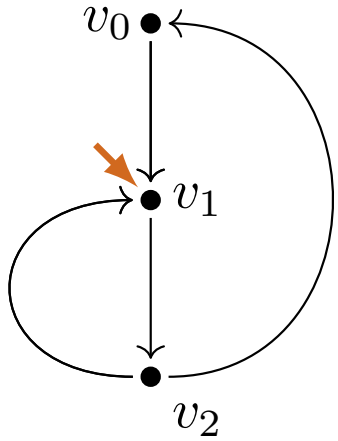


(L1),~~(L2)~~



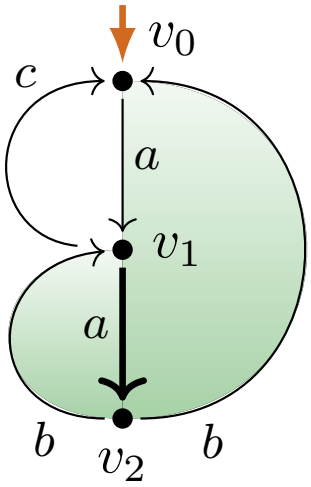
(L1),(L2),~~(L3)~~



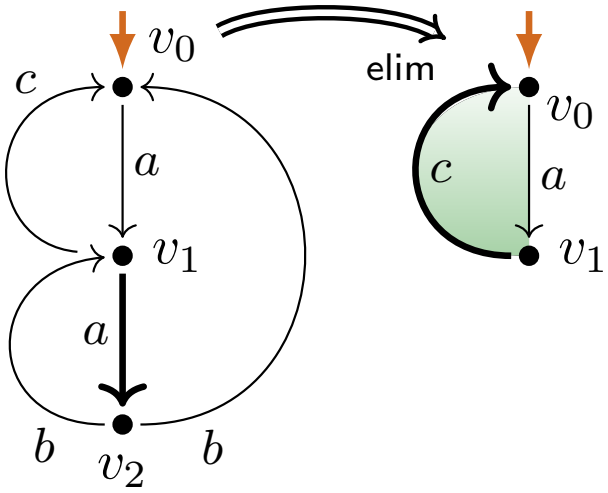


loop chart

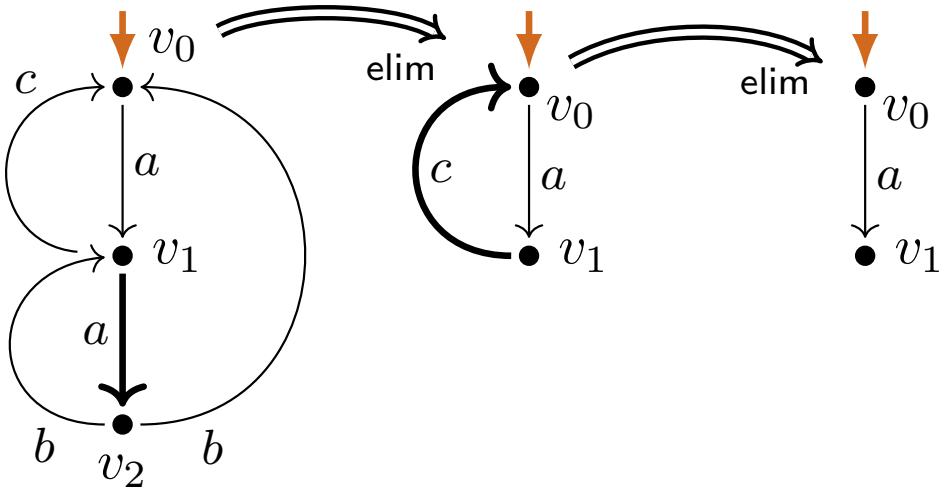
Loop existence and elimination



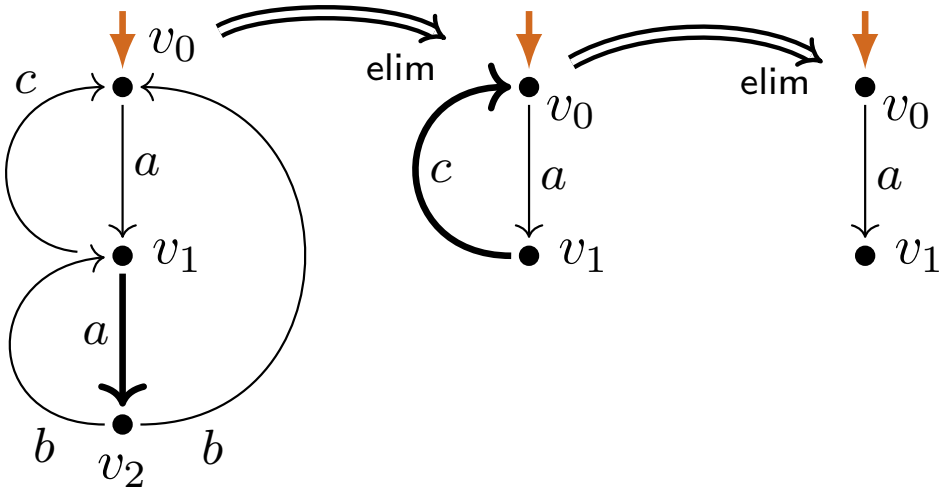
Loop existence and elimination

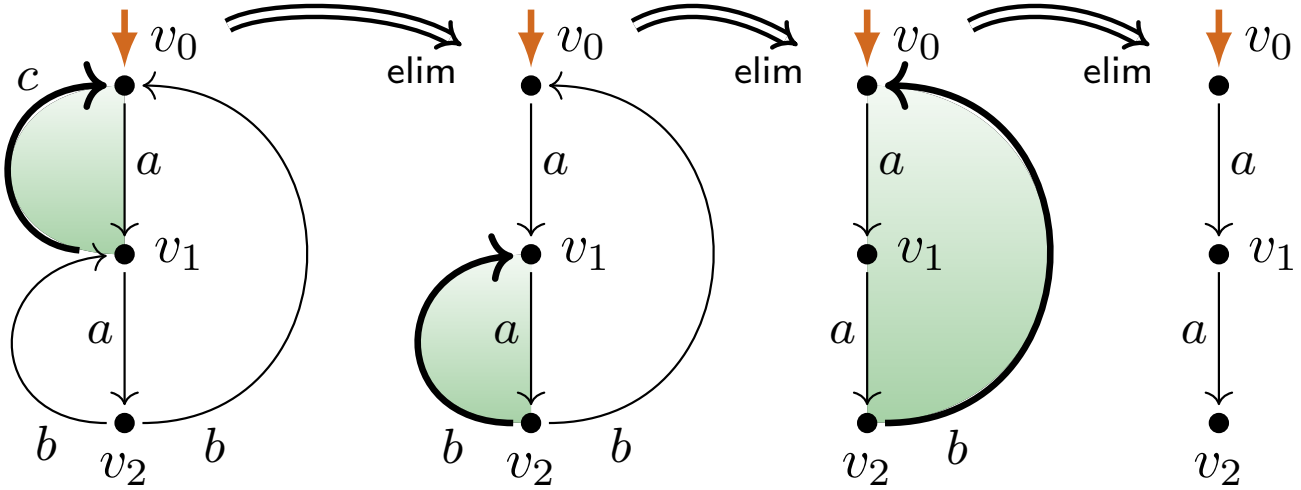


Loop existence and elimination



Loop existence and elimination





LEE

Definition

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

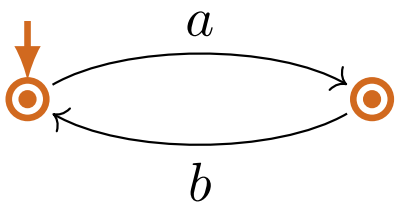
$$\exists G_0 \left(G \longrightarrow^*_{\text{elim}} G_0 \not\longrightarrow_{\text{elim}} \right.$$
$$\left. \wedge\ G_0 \text{ permits no infinite path} \right).$$

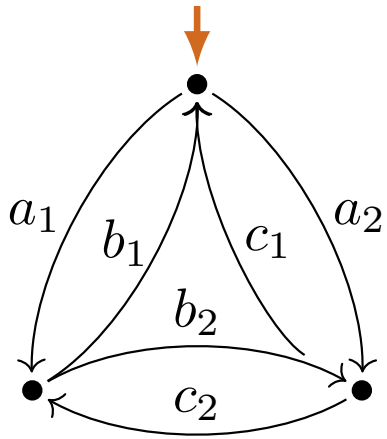
LEE

Definition

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

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



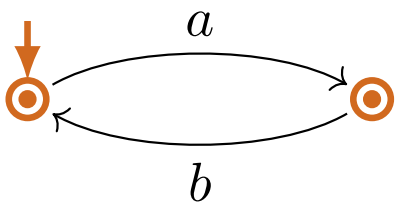


LEE

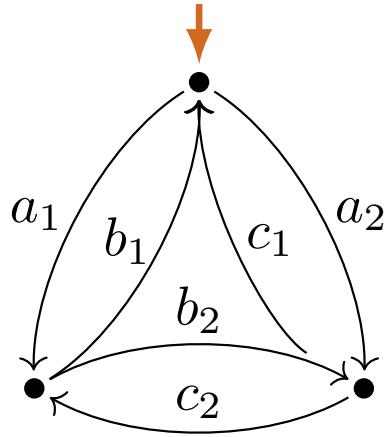
Definition

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

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

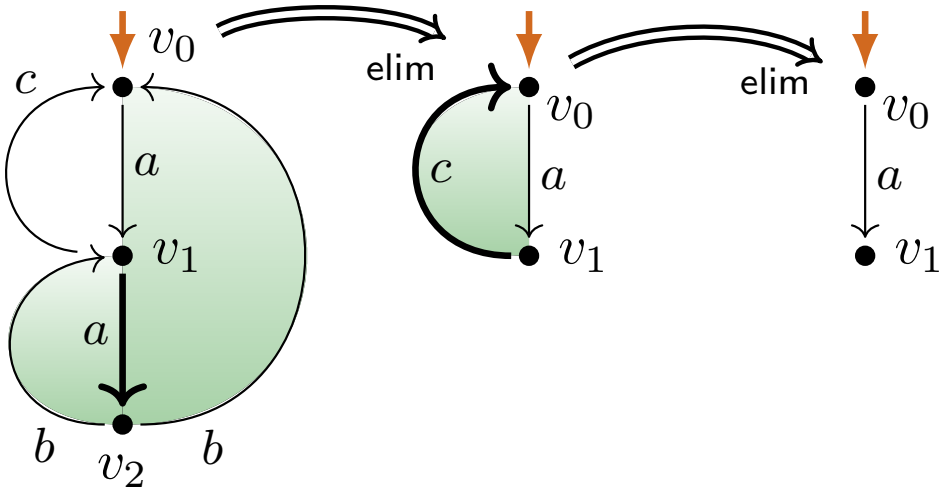


not LEE

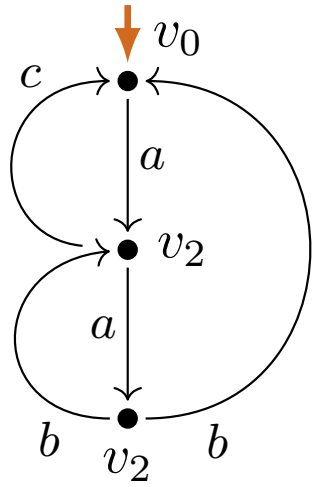


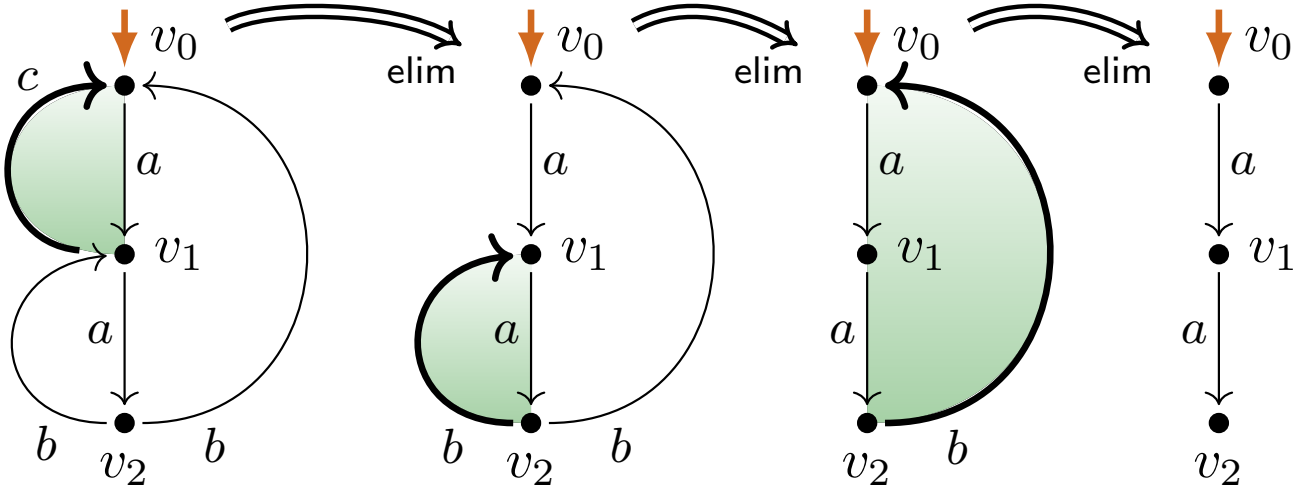
not LEE

LEE

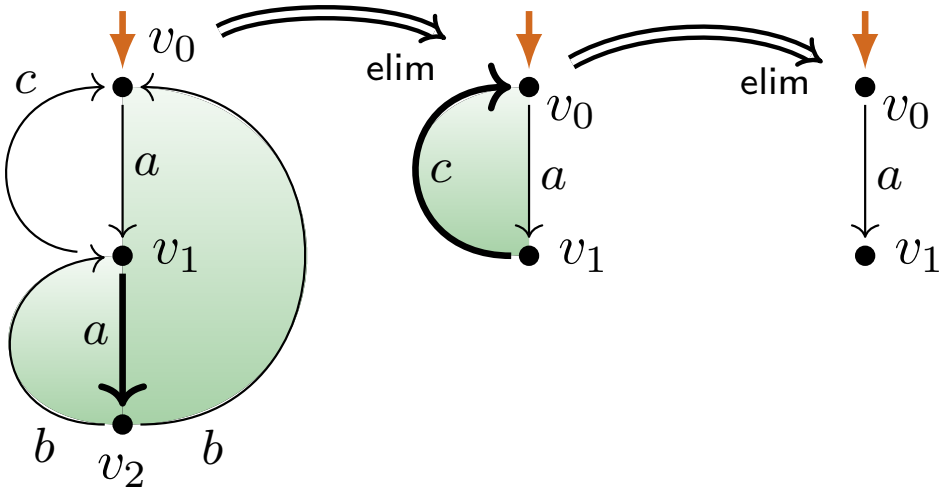

v_0
c
a
v_1
a
b
v_2
b
elim
v_0
c
a
v_1
elim
v_0
a
v_1

LEE

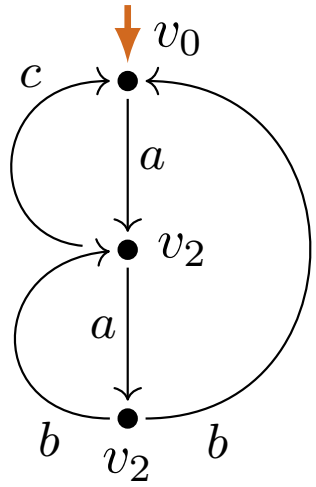

v_0
c
a
v_2
a
b
v_2
b

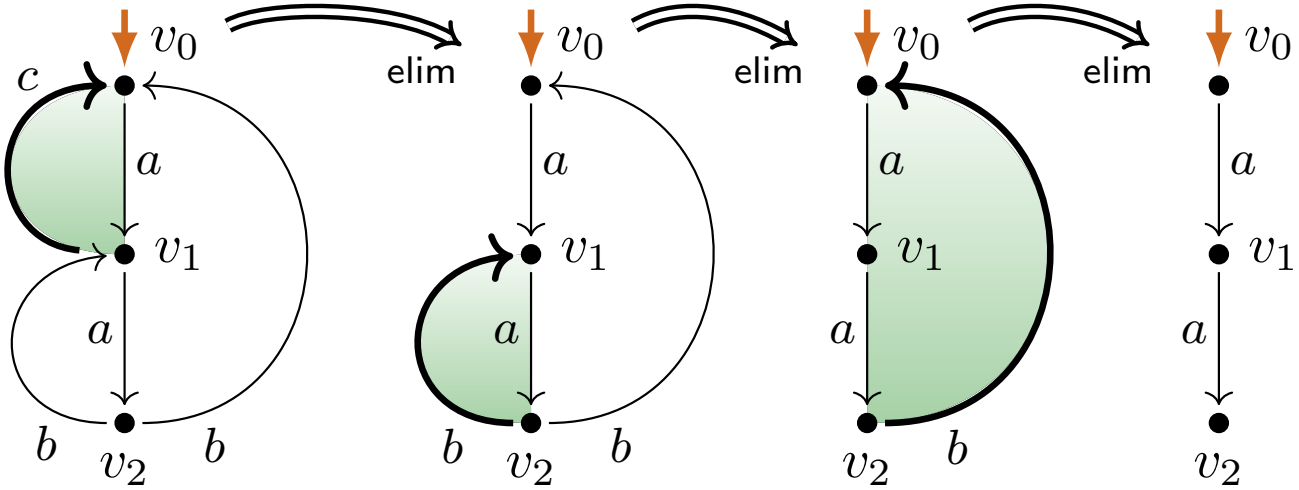

v_0
c
a
v_1
a
b
v_2
b
elim
v_0
a
v_1
a
b
v_2
b
elim
v_0
a
v_1
a
v_2
b
elim
v_0
a
v_1
a
v_2

Layered LEE

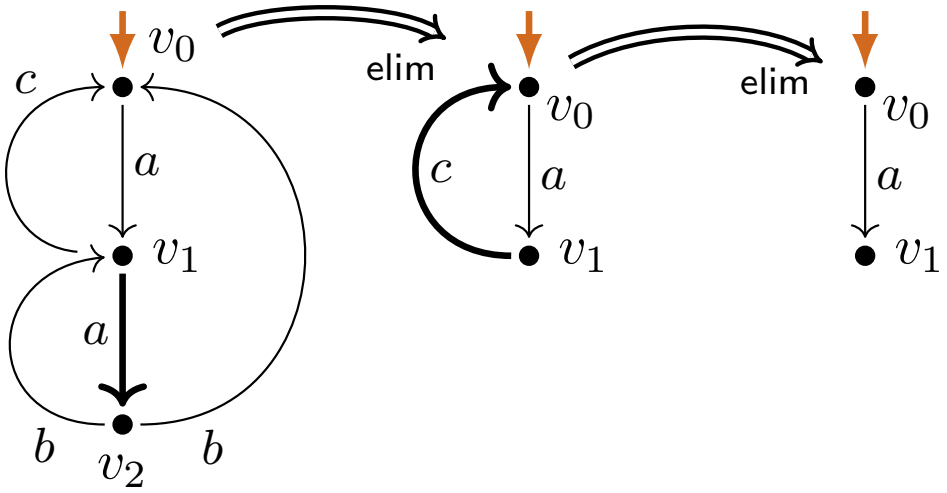


layered LEE
LLEE

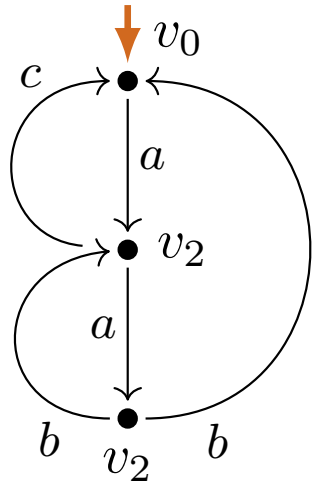


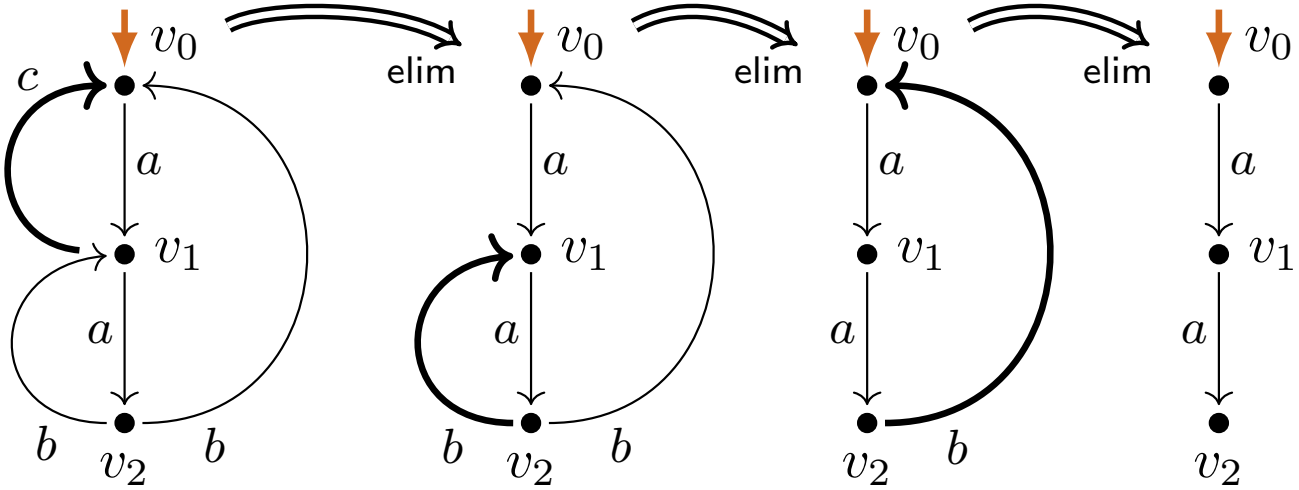


Layered LEE

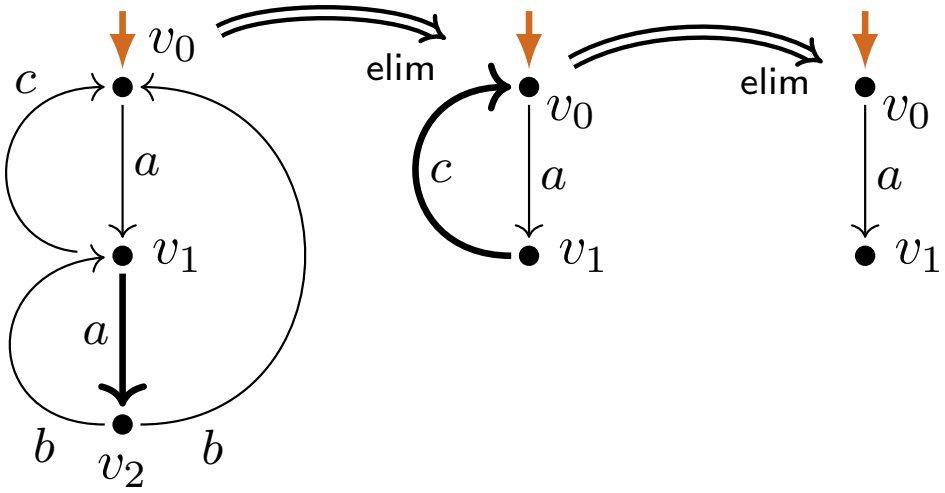


layered LEE
LLEE



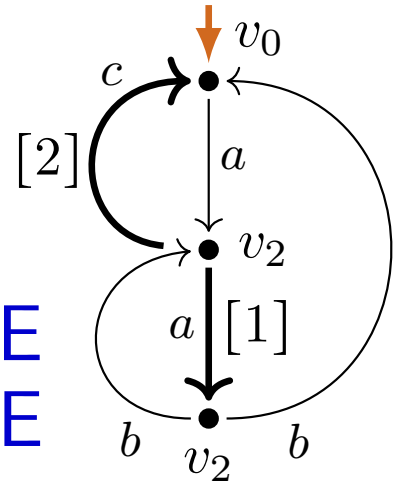


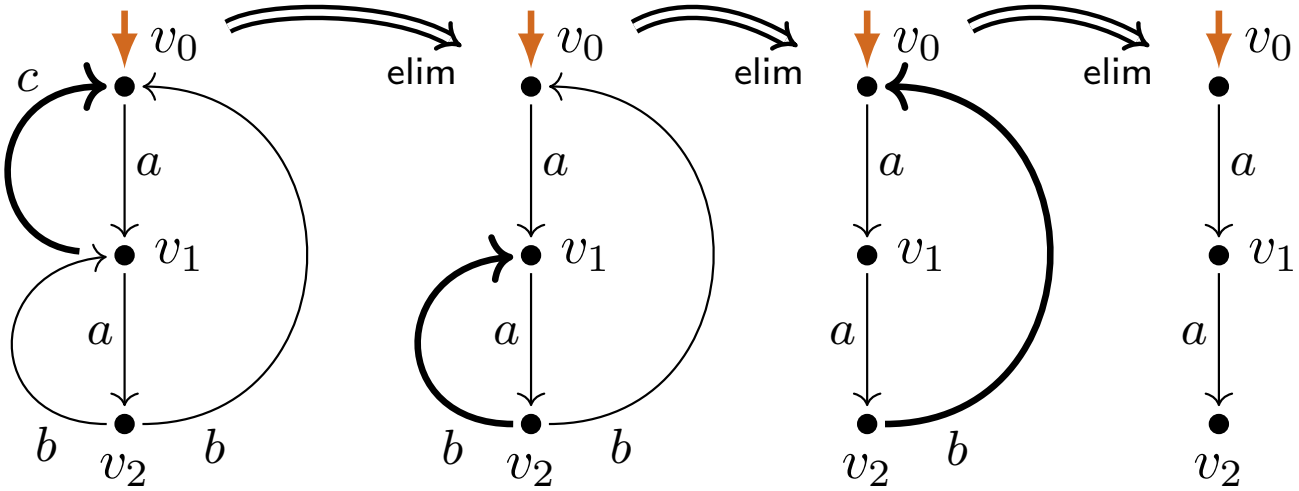
Layered LEE



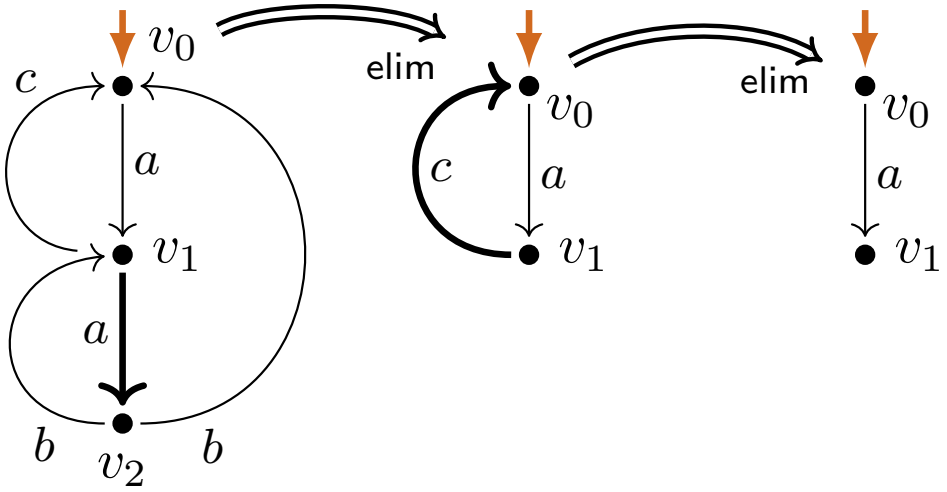
layered LEE

LLEE



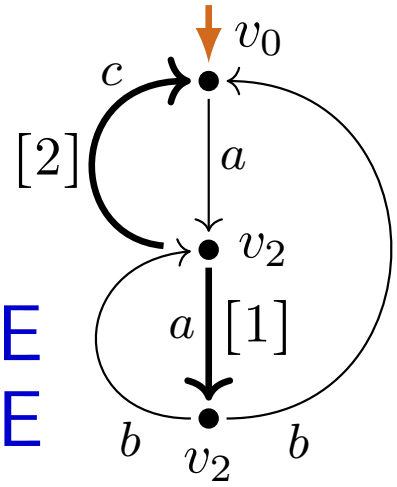


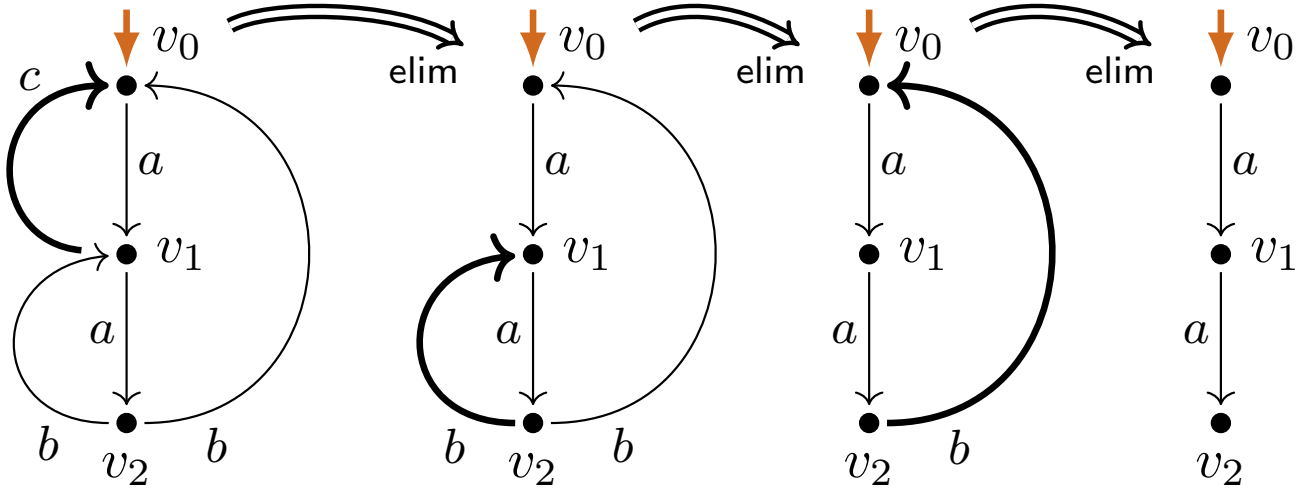
LLEE-witness and LLEE-graph

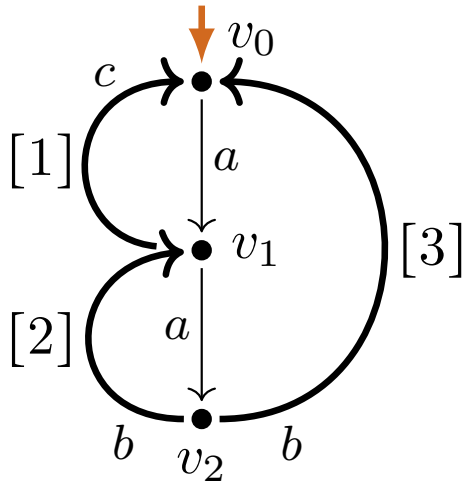


layered LEE

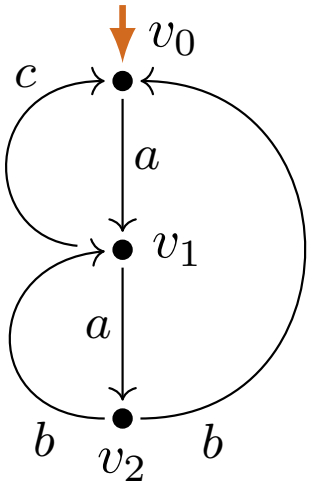
LLEE



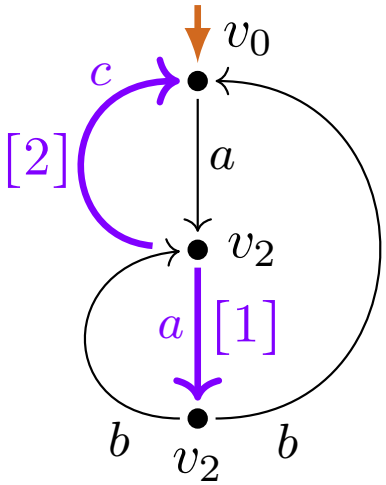




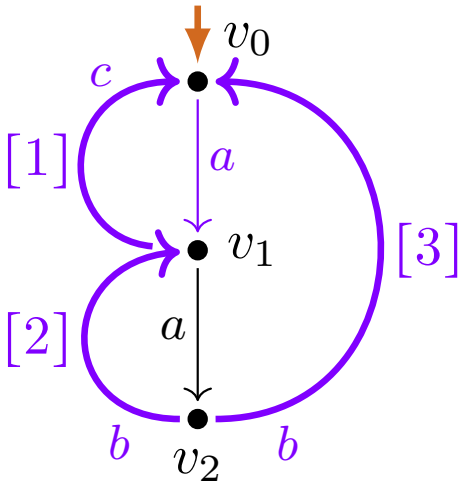
LLEE-witness and LLEE-graph



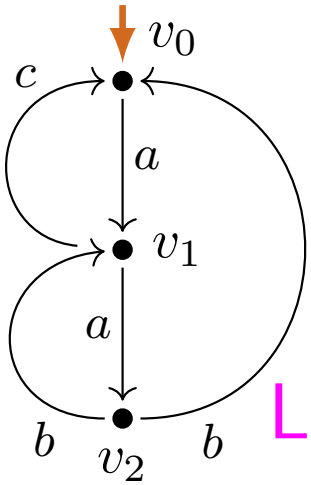
LLEE-witness



LLEE-witness

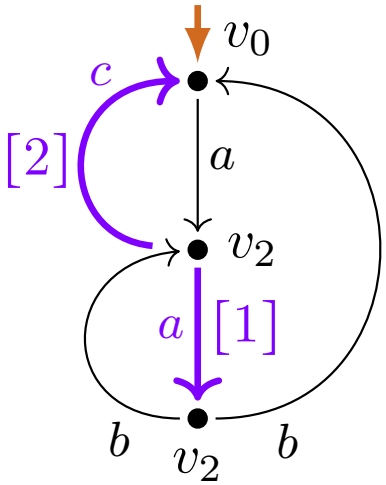


LLEE-witness and LLEE-graph

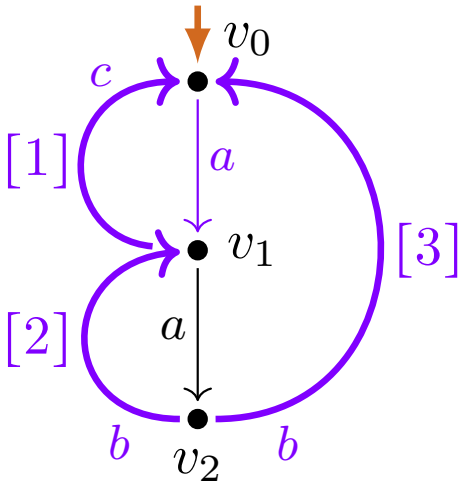


LLEE-graph

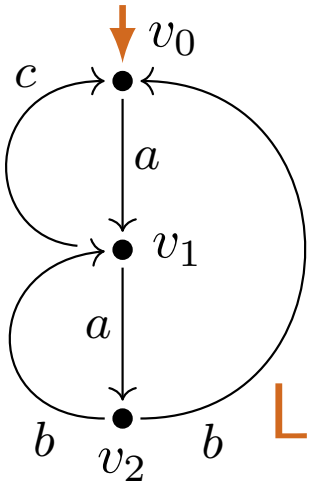
LLEE-witness



LLEE-witness

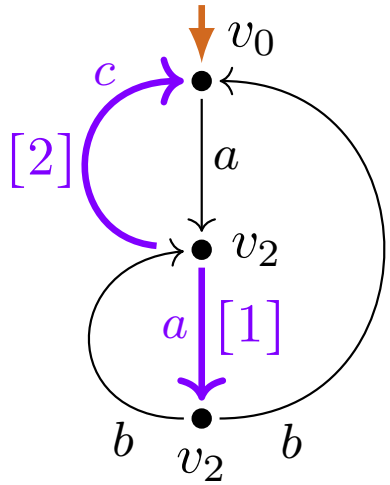


LLEE-witness and LLEE-graph

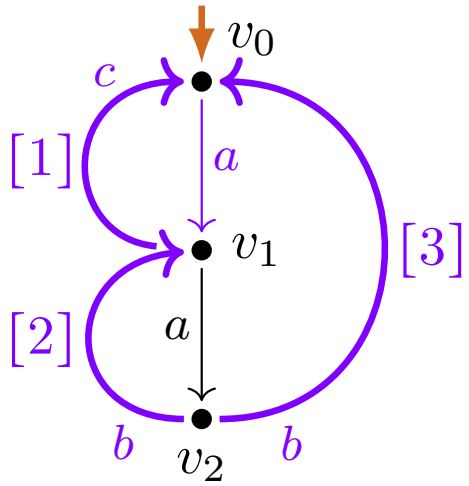


LLEE-graph

layered LLEE-witness



layered LLEE-witness



Properties of LEE-graphs

Theorem *(⇐ G–Fokkink, 2020)*

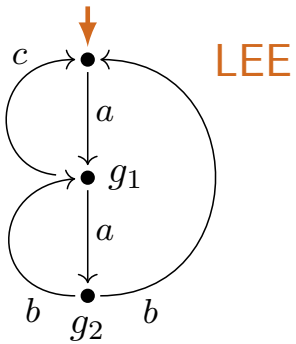
A process graph G
is $[\![\cdot]\!]_P$-expressible by an $(*/\not{1})$ regular expression
(i.e. P-expressible modulo bisimilarity by an $(*/\not{1})$ reg. expr.)
if and only if
the bisimulation collapse of G satisfies LEE.

Properties of LEE-graphs

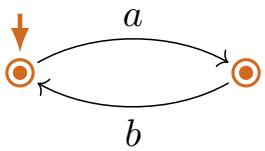
Theorem *(⇐ G–Fokkink, 2020)*

A process graph G
is $[\![\cdot]\!]_P$-expressible by an $(*/\not{1})$ regular expression
(i.e. P-expressible modulo bisimilarity by an $(*/\not{1})$ reg. expr.)
if and only if
the bisimulation collapse of G satisfies LEE.

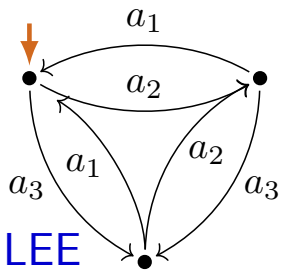
Hence $[\![\cdot]\!]_P$-expressible | **not** $[\![\cdot]\!]_P$-expressible by $(*/\not{1})$ regular expressions:



LEE



not LEE

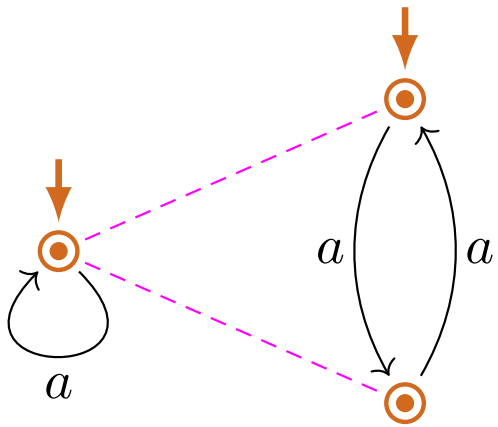


not LEE

LEE under bisimulation

Observation

- LEE is not invariant under bisimulation.

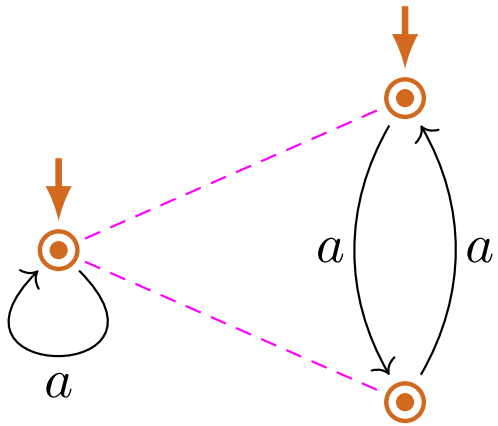


LEE ¬LEE

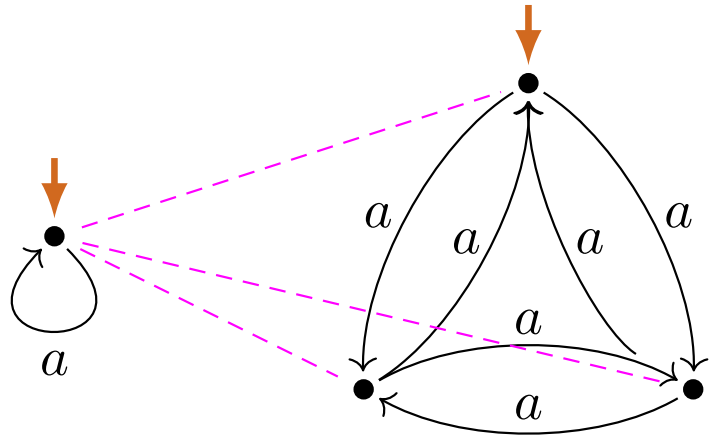
LEE under bisimulation

Observation

- LEE is not invariant under bisimulation.



LEE ¬LEE

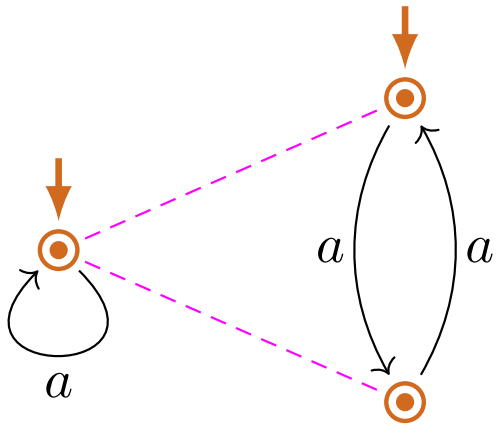


LEE ¬LEE

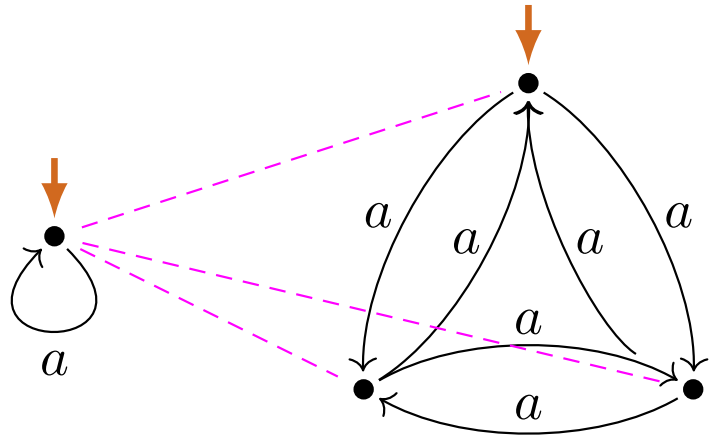
LEE under bisimulation

Observation

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



LEE ¬LEE



LEE ¬LEE

LEE under functional bisimulation

Lemma

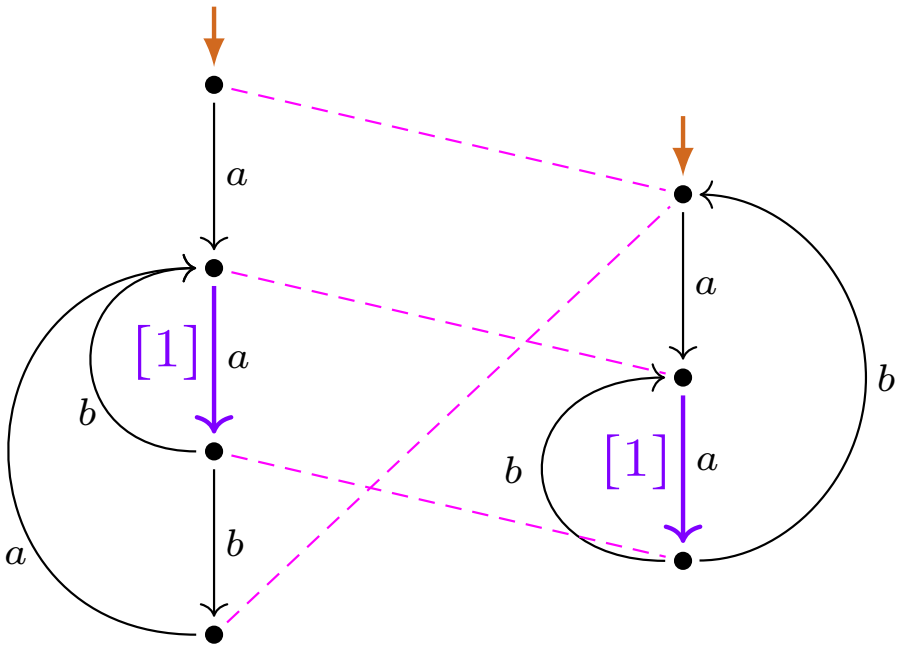
(i) LEE *is preserved by functional bisimulations:*

$$\text{LEE}(G_1) \wedge G_1 \twoheadrightarrow G_2 \implies \text{LEE}(G_2)\,.$$

Proof (Idea).

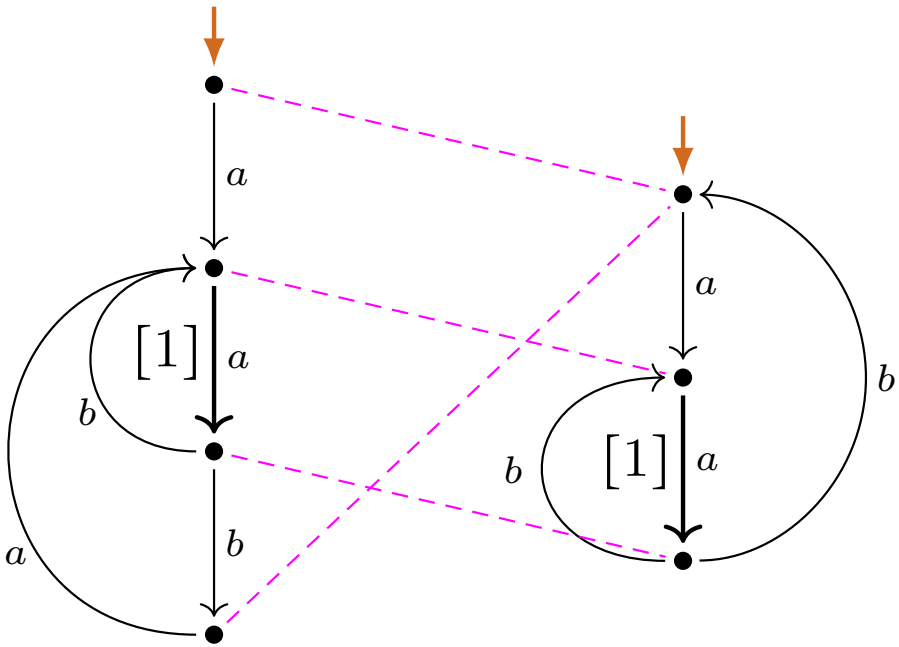
Use loop elimination in G_1 to carry out loop elimination in G_2.

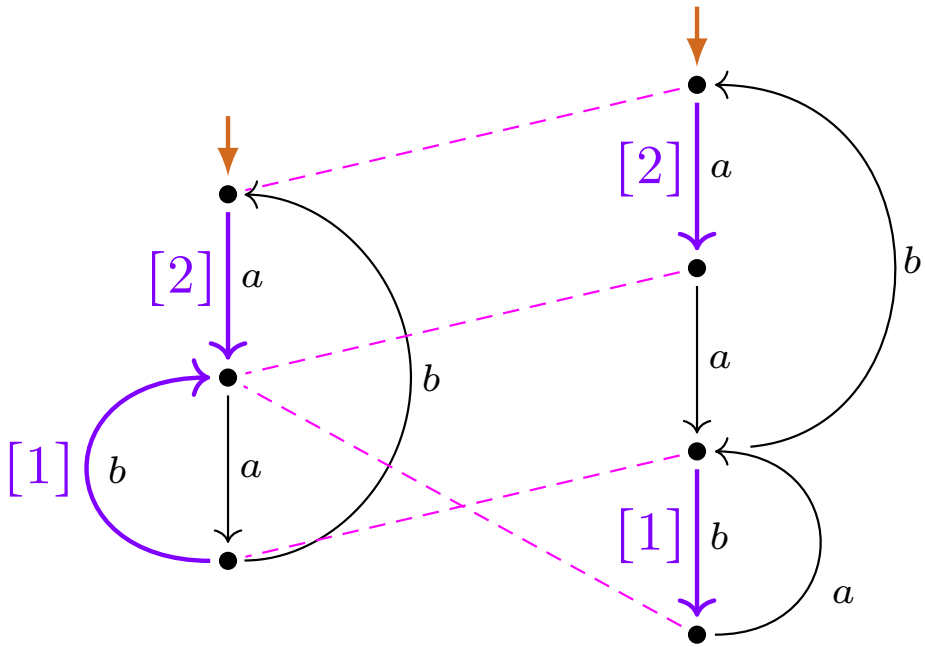
Collapsing LLEE-witnesses: global argument



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

Collapsing LLEE-witnesses: global argument





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

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

LEE under functional bisimulation / bisimulation collapse

Lemma

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

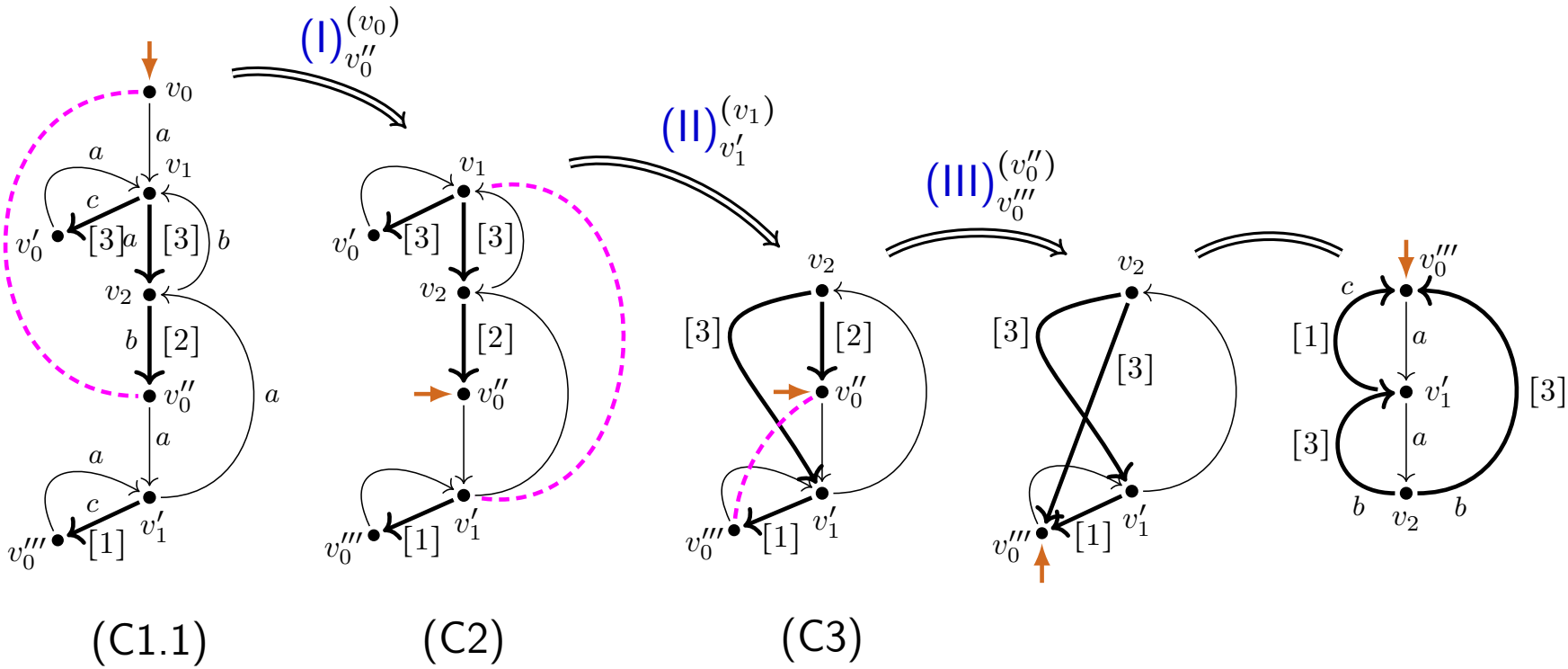
$$\mathsf{LEE}(G_1) \wedge G_1 \twoheadrightarrow G_2 \implies \mathsf{LEE}(G_2)\,.$$

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

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



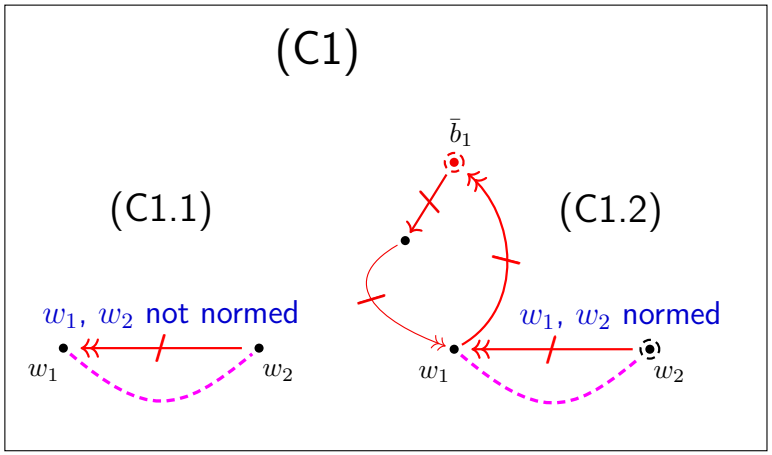
Lemma

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

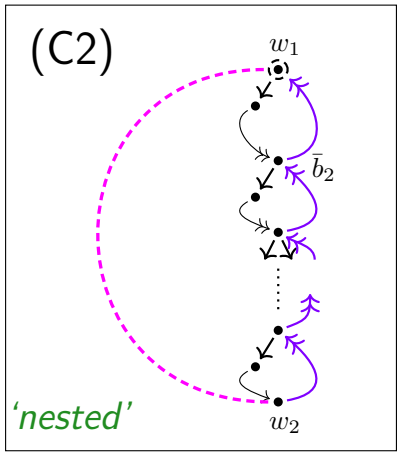
Reduced bisimilarity redundancies in LLEE-graphs (no 1-trans.!)

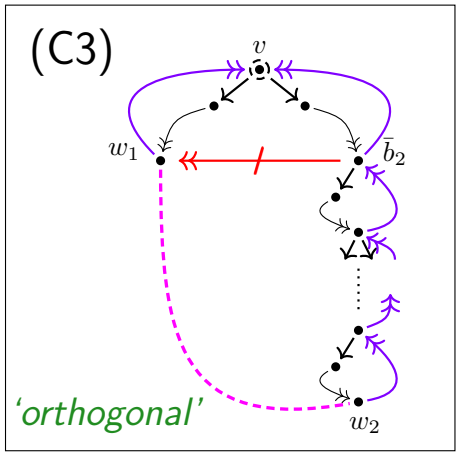
(G–Fokkink, LICS'20)

w_1, w_2 in different scc's



w_1, w_2 in the same scc





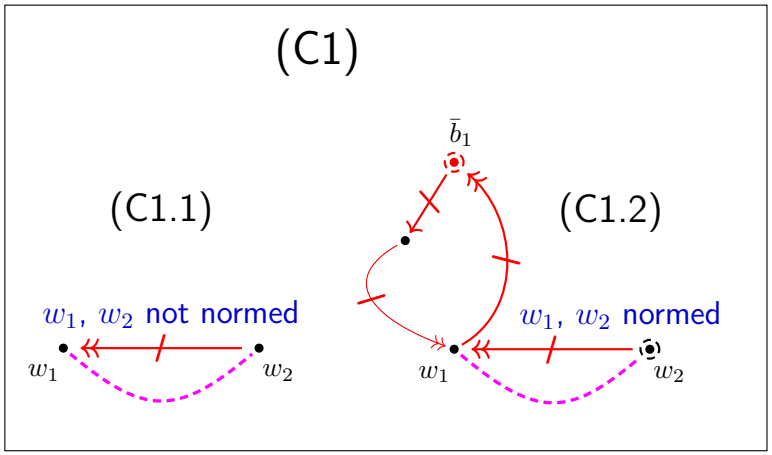
Lemma

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

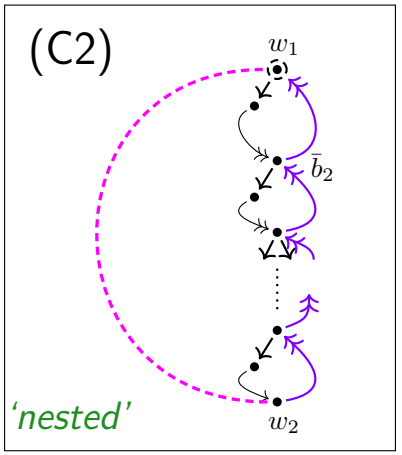
Reduced bisimilarity redundancies in LLEE-graphs (no 1-trans.!)

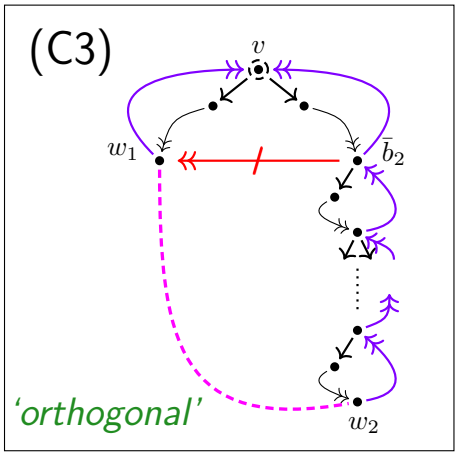
(G–Fokkink, LICS'20)

w_1, w_2 in different scc's



w_1, w_2 in the same scc





Lemma

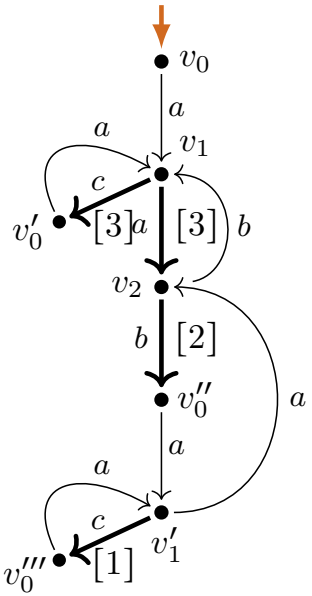
Every not collapsed LLEE-graph contains bisimilar vertices $w_1 \neq w_2$ of kind (C1), (C2), *or* (C3) *(a reduced bisimilarity redundancy $\langle w_1, w_2 \rangle$):*

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



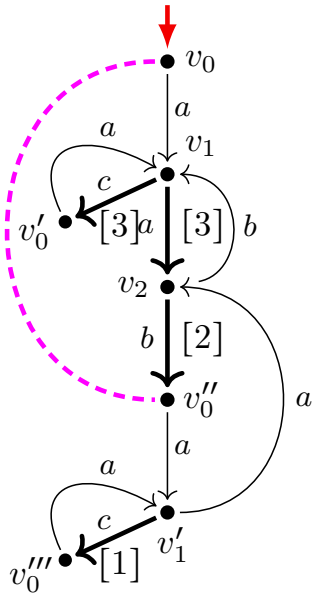
(C1.1)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



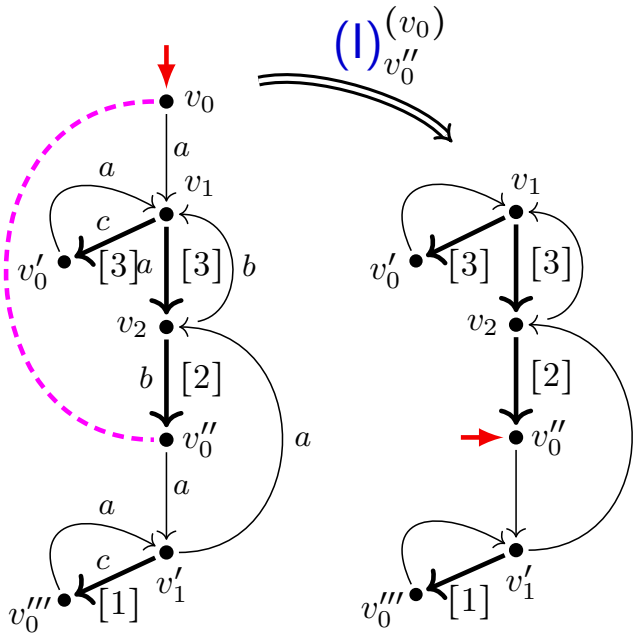
(C1.1)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



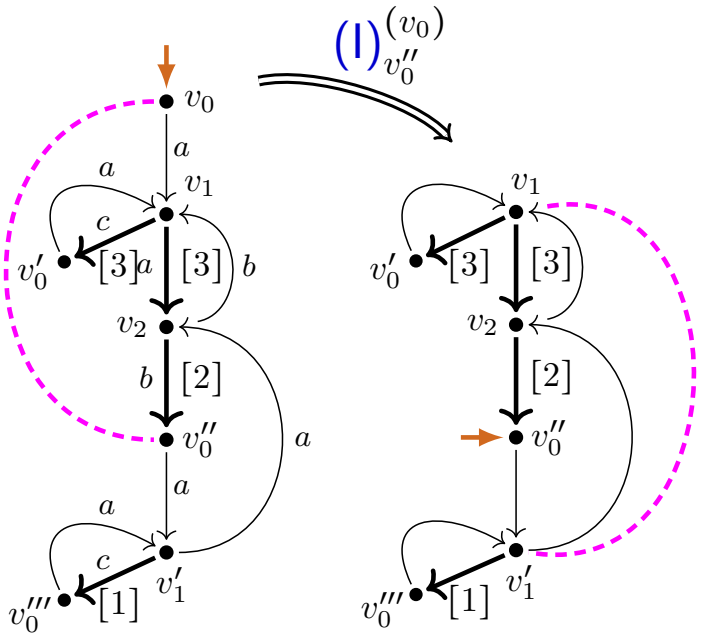
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



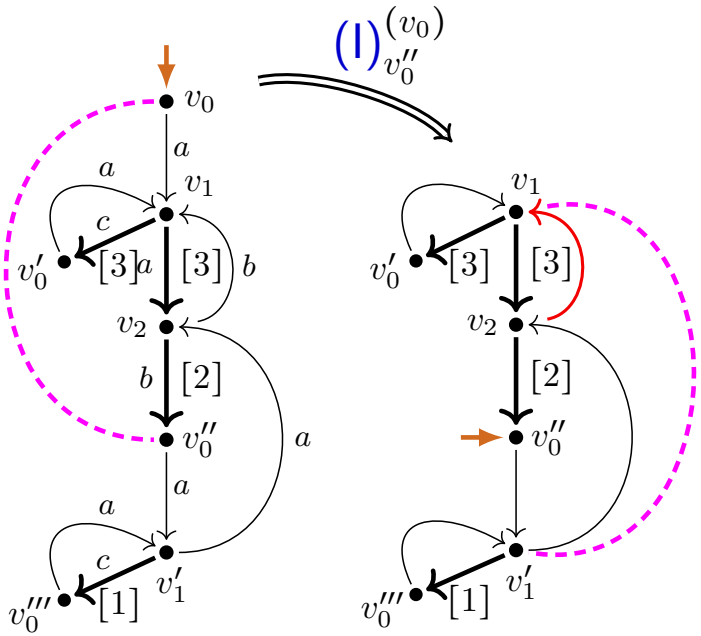
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



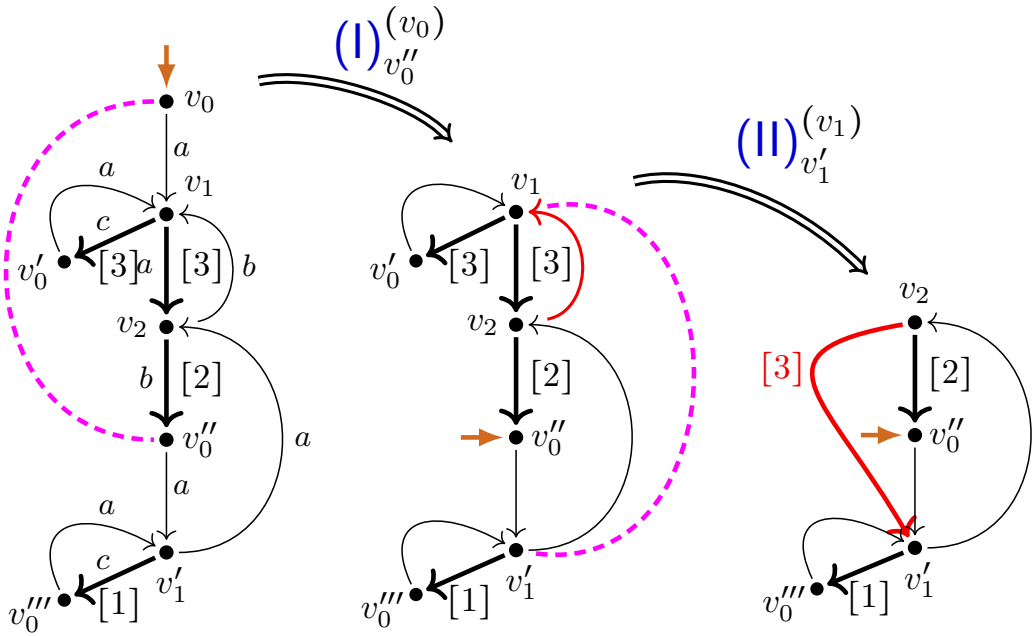
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



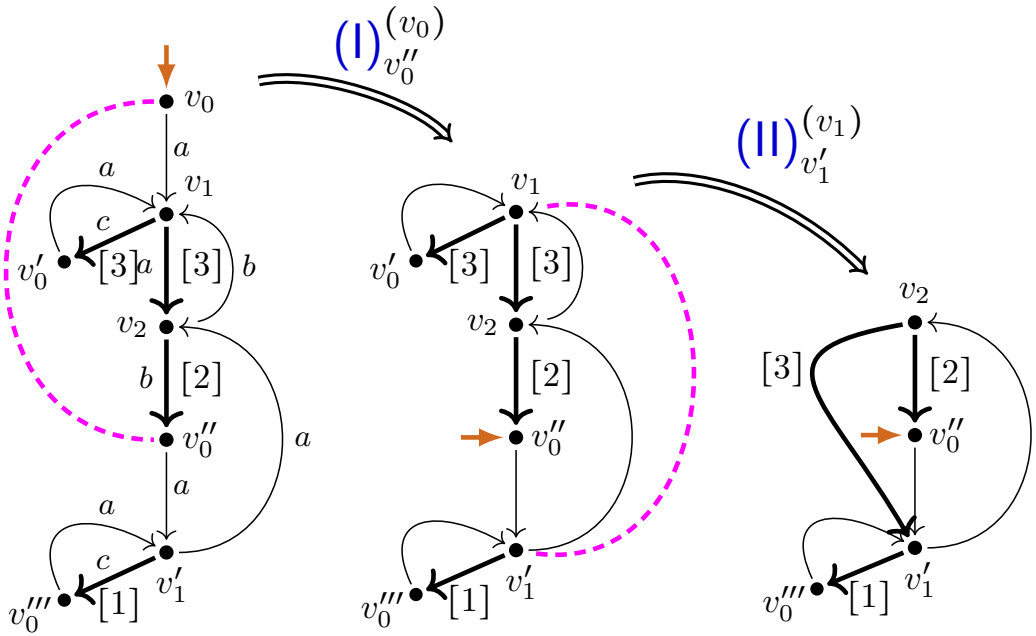
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



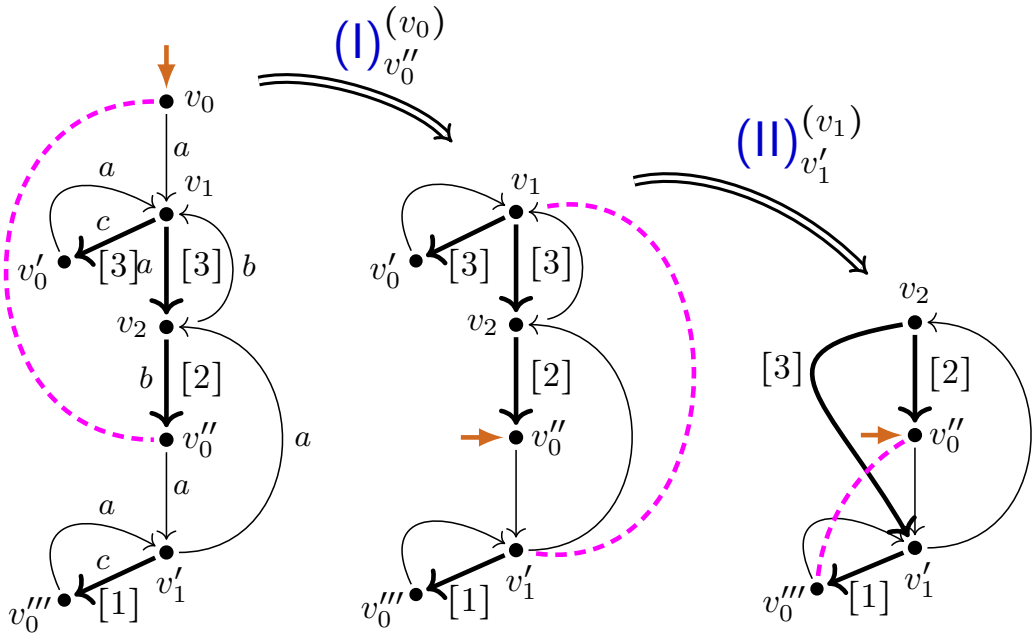
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



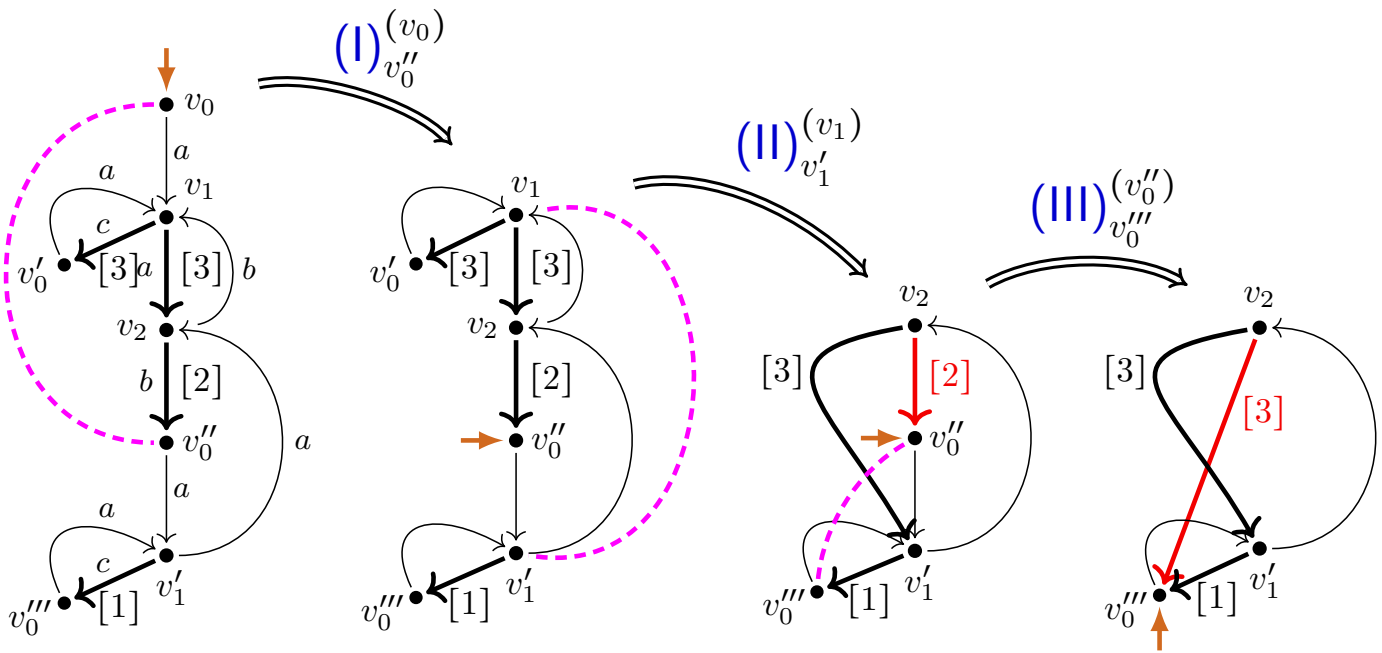
(C1.1) (C2)

Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)

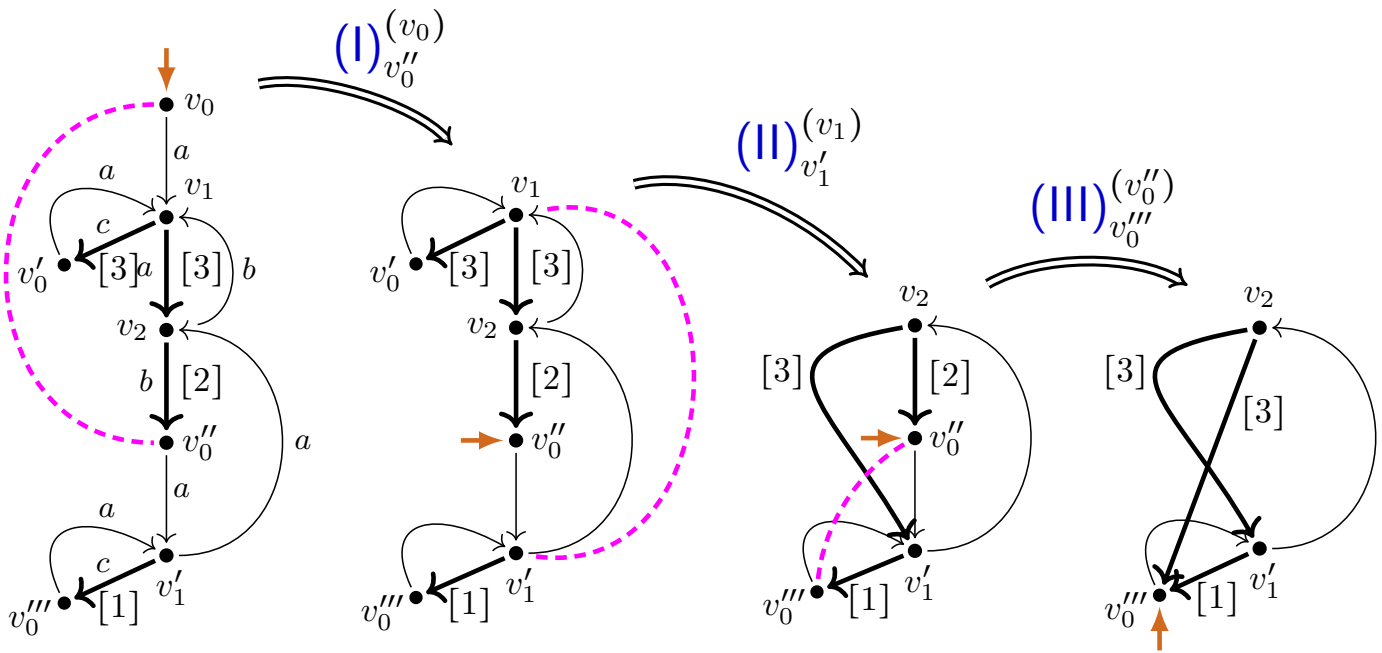


Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)

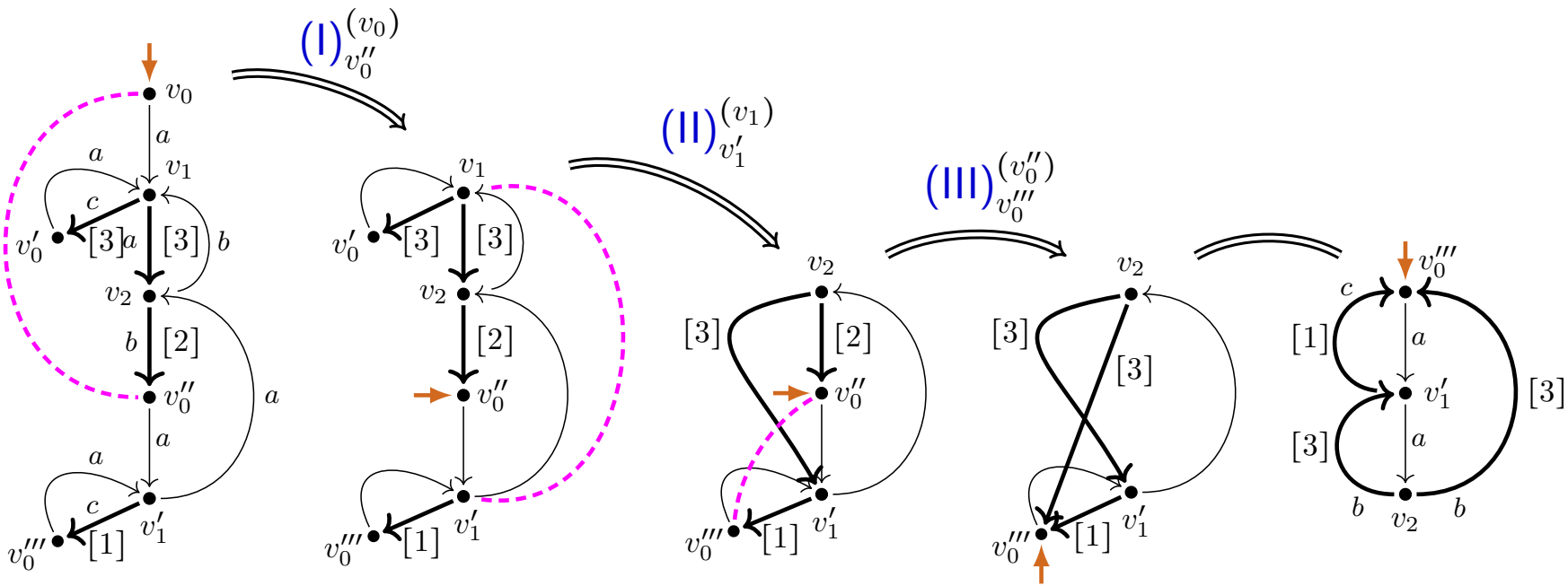


Lemma

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

LLEE-preserving collapse of LLEE-graphs: stepwise proof

(no 1-transitions!)



Lemma

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

Results facilitated by steps (st2)

Corollary *(⇐ G–Fokkink, LICS 2020)*

For every process graph G, TFAE:

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

(ii) The bisimulation collapse of G satisfies LLEE.

(iii) The bisim. collapse of G is $[\![\cdot]\!]_P$-expressible by a $(*/\pm)$ reg. expression.

Results facilitated by steps (st2)

Corollary *(⇐ G–Fokkink, LICS 2020)*

For every process graph G, TFAE:

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

(ii) The bisimulation collapse of G satisfies LLEE.

(iii) The bisim. collapse of G is $[\![\cdot]\!]_P$-expressible by a $(*/\pm)$ reg. expression.

Lemma *(IWC 2026)*

LEE, LLEE $\in$ PTIME ($\leq$ cubic in graph size).

Corollary

$[\![\cdot]\!]_P$-expressibility by a $(*/\pm)$ regular expression $\in$ PTIME.

Results facilitated by steps (st2)

Corollary *(⇐ G–Fokkink, LICS 2020)*

For every process graph G, TFAE:

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

(ii) The bisimulation collapse of G satisfies LLEE.

(iii) The bisim. collapse of G is $[\![\cdot]\!]_P$-expressible by a $(*/\pm)$ reg. expression.

Lemma *(IWC 2026)*

LEE, LLEE $\in$ PTIME ($\leq$ cubic in graph size).

Corollary

$[\![\cdot]\!]_P$-expressibility by a $(*/\pm)$ regular expression $\in$ PTIME.

Theorem *(G–Fokkink, LICS 2020)*

Mil restricted to $(*/\pm)$ regular expressions is complete.

BBP (Bergstra, Bethke, Ponse)

Small steps for hard questions

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

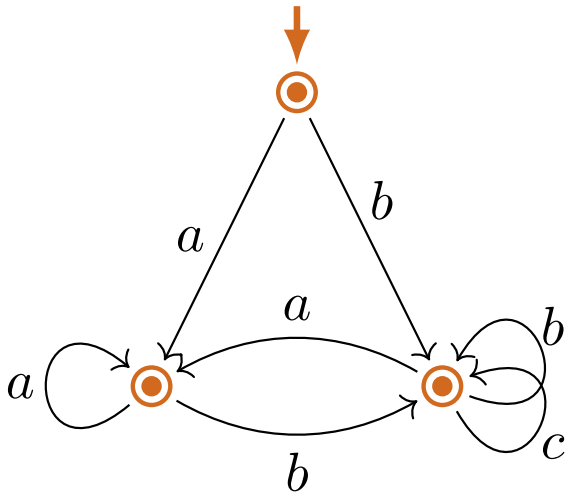
  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\triangleq$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

1-Graphs with LLEE $\stackrel{\wedge}{=}$ (induced) graphs with 1-LLEE

graph

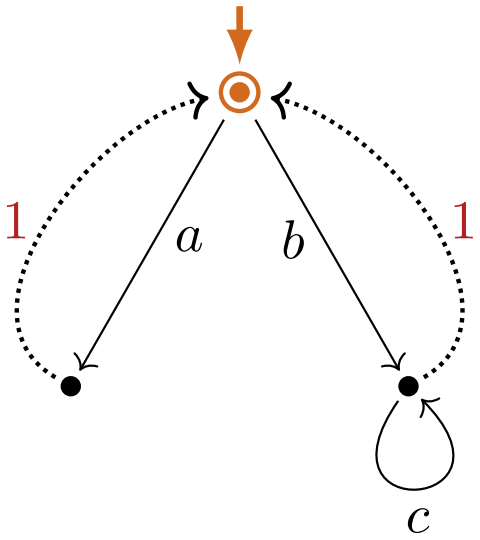


not LLEE

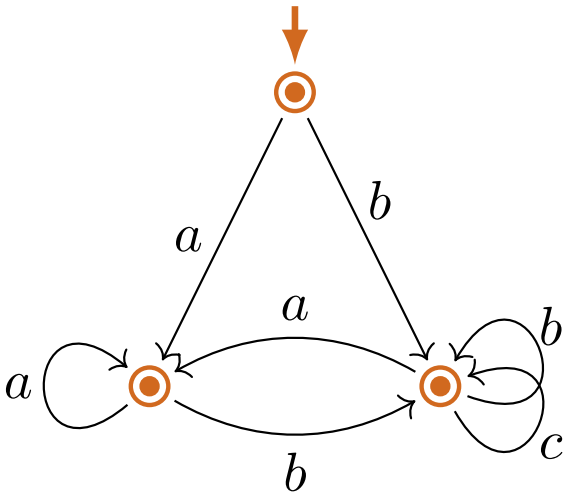
$$P((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$$

1-Graphs with LLEE $\hat{=}$ (induced) graphs with 1-LLEE

1-graph



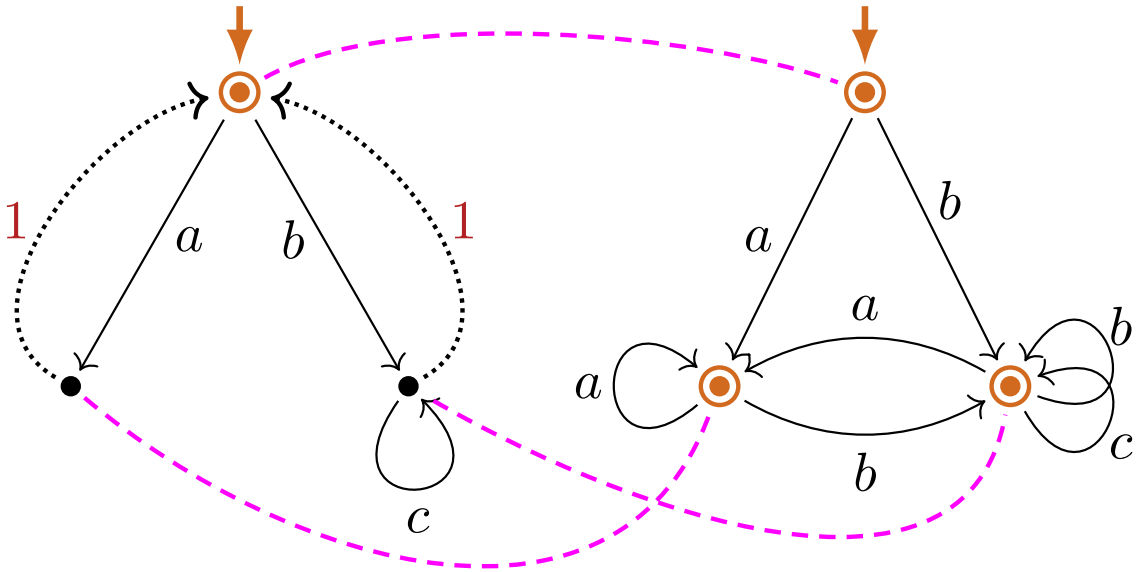
(induced) graph



$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$ $P((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

1-Graphs with LLEE $\hat{=}$ (induced) graphs with 1-LLEE

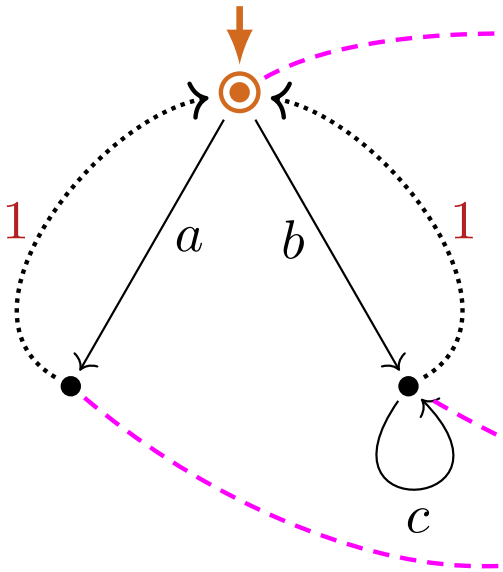
1-graph (induced) graph



$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$ $P((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

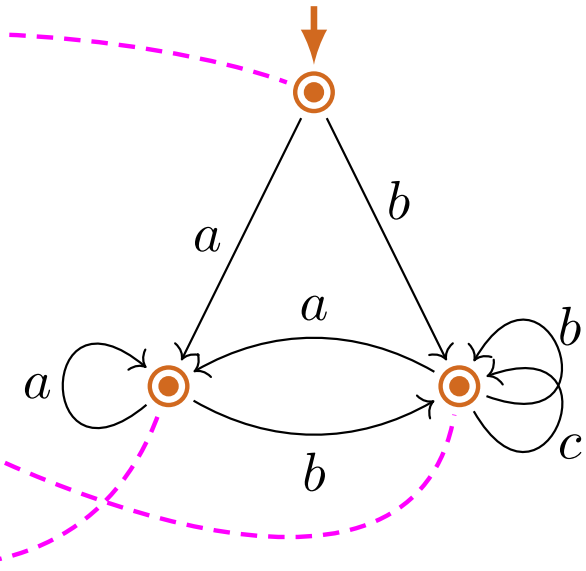
1-Graphs with LLEE $\hat{=}$ (induced) graphs with 1-LLEE

1-graph



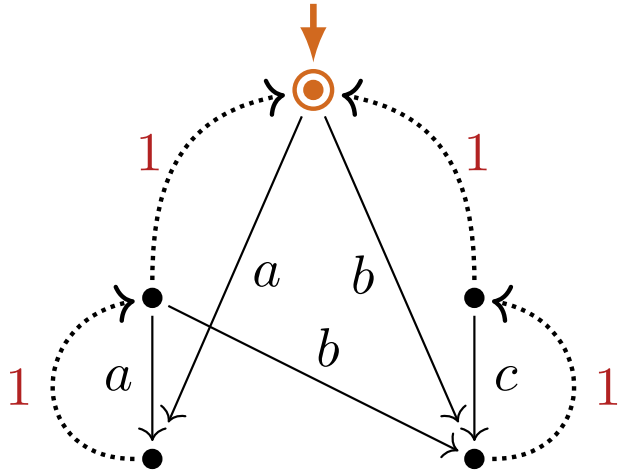
$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

(induced) graph



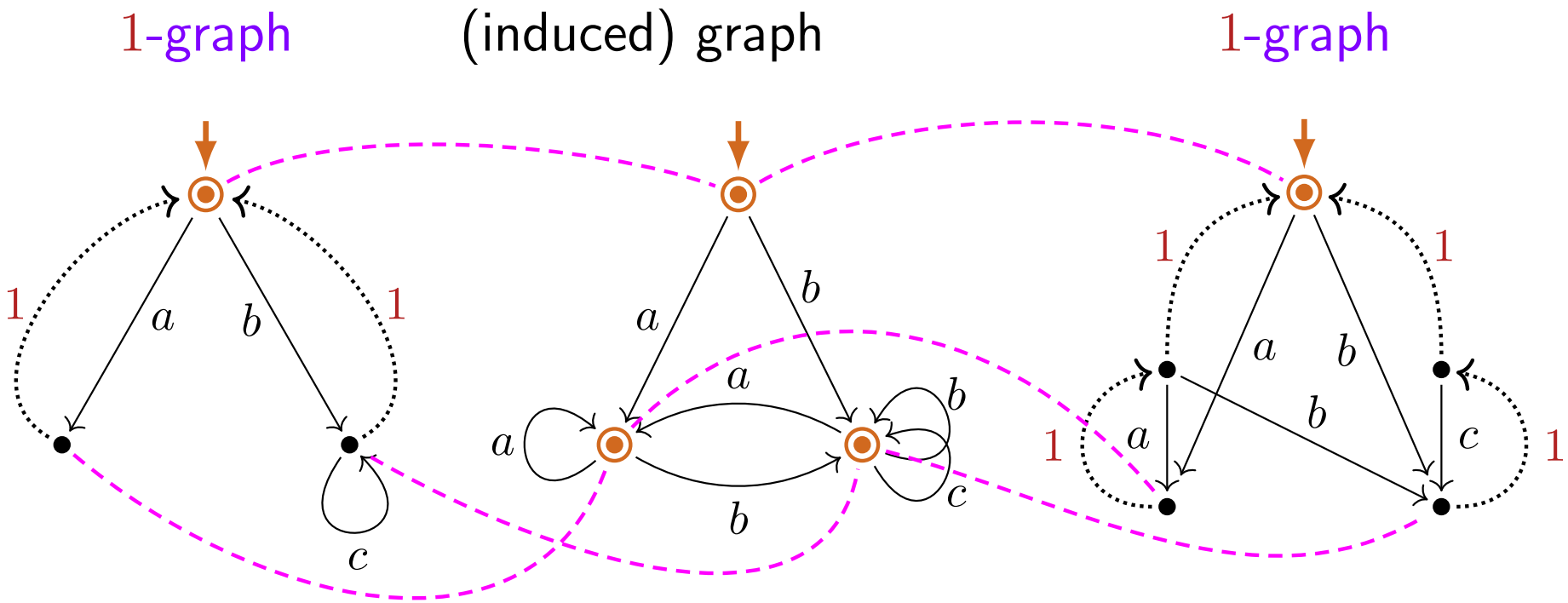
$P((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

1-graph



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

1-Graphs with LLEE $\hat{=}$ (induced) graphs with 1-LLEE

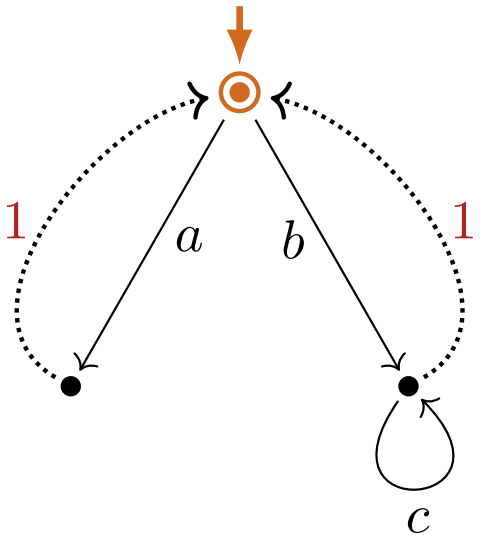


$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$ $P((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

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

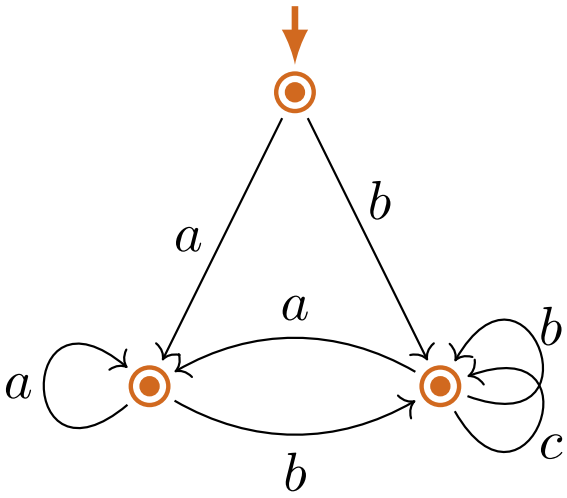
1-Graphs with LLEE $\stackrel{\wedge}{=}$ (induced) graphs with 1-LLEE

1-graph



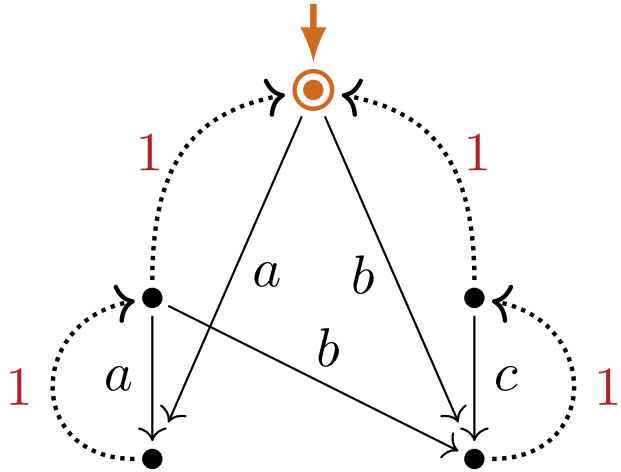
$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

(induced) graph



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

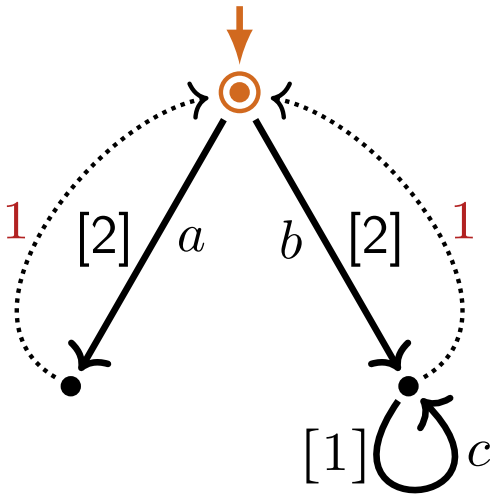
1-graph



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

1-Graphs with LLEE $\hat{=}$ (induced) graphs with 1-LLEE

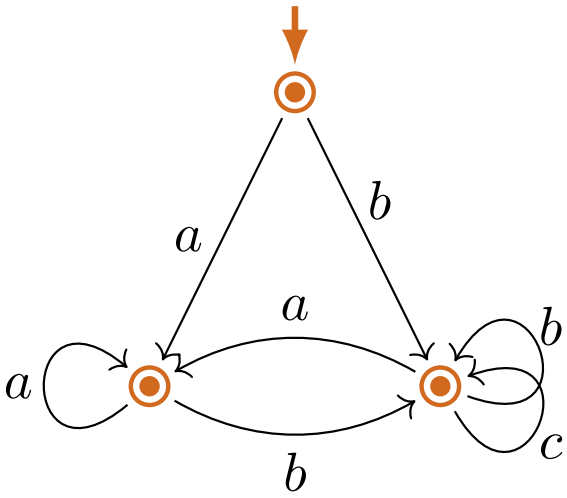
1-graph



LLEE

$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

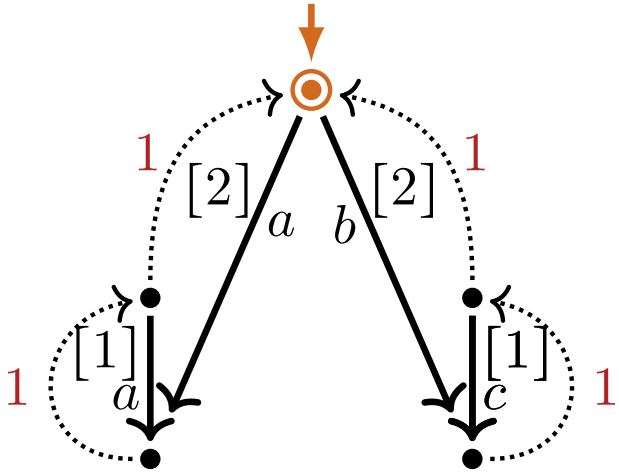
(induced) graph



not LLEE

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

1-graph

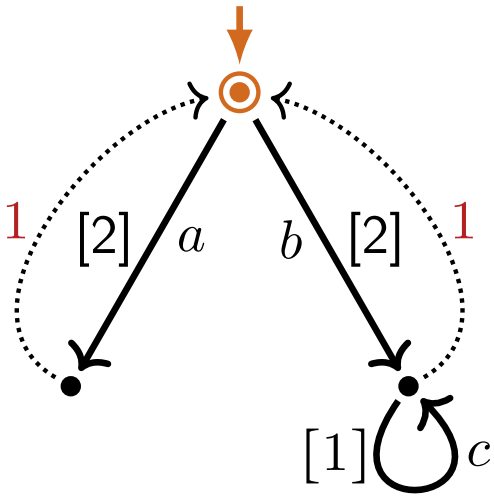


LLEE

$\underline{P}\big(\big(a \cdot (1 \cdot ((a \cdot 1)^* \cdot 1))$
$\cdot\, b \cdot ((1 \cdot (c \cdot 1)^*)) \cdot 1$
$+\, b \cdot ((1 \cdot (c \cdot 1)^*)) \cdot 1\big)^*\big)$

1-Graphs with LLEE $\hat{=}$ (induced) graphs with 1-LLEE

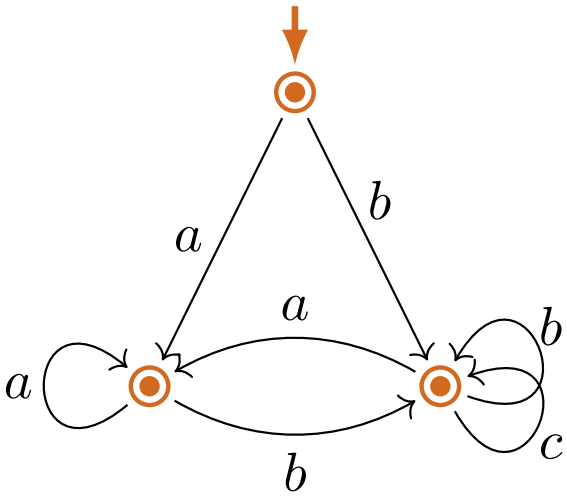
1-graph



LLEE

$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$

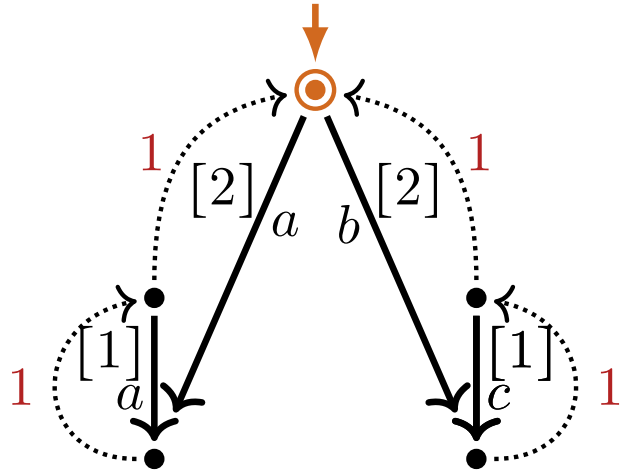
(induced) graph



1-LLEE

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

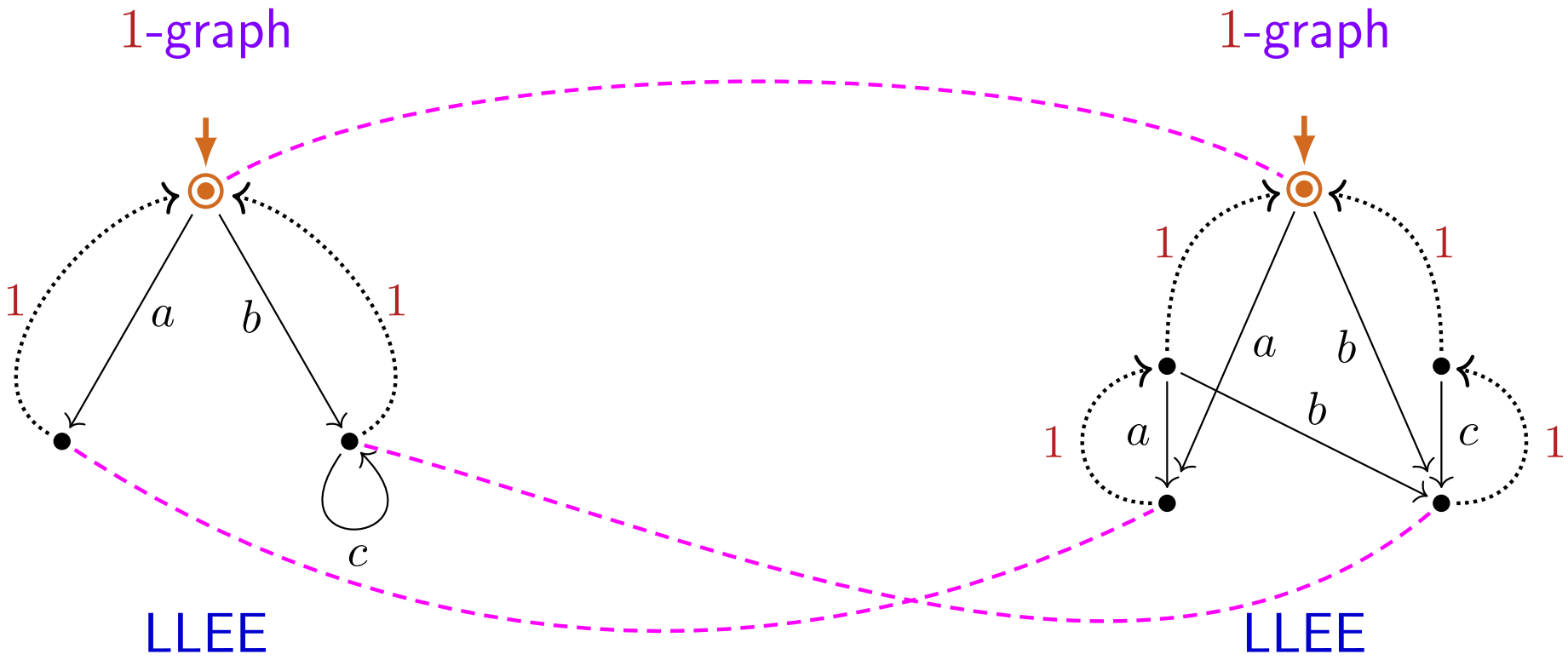
1-graph



LLEE

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

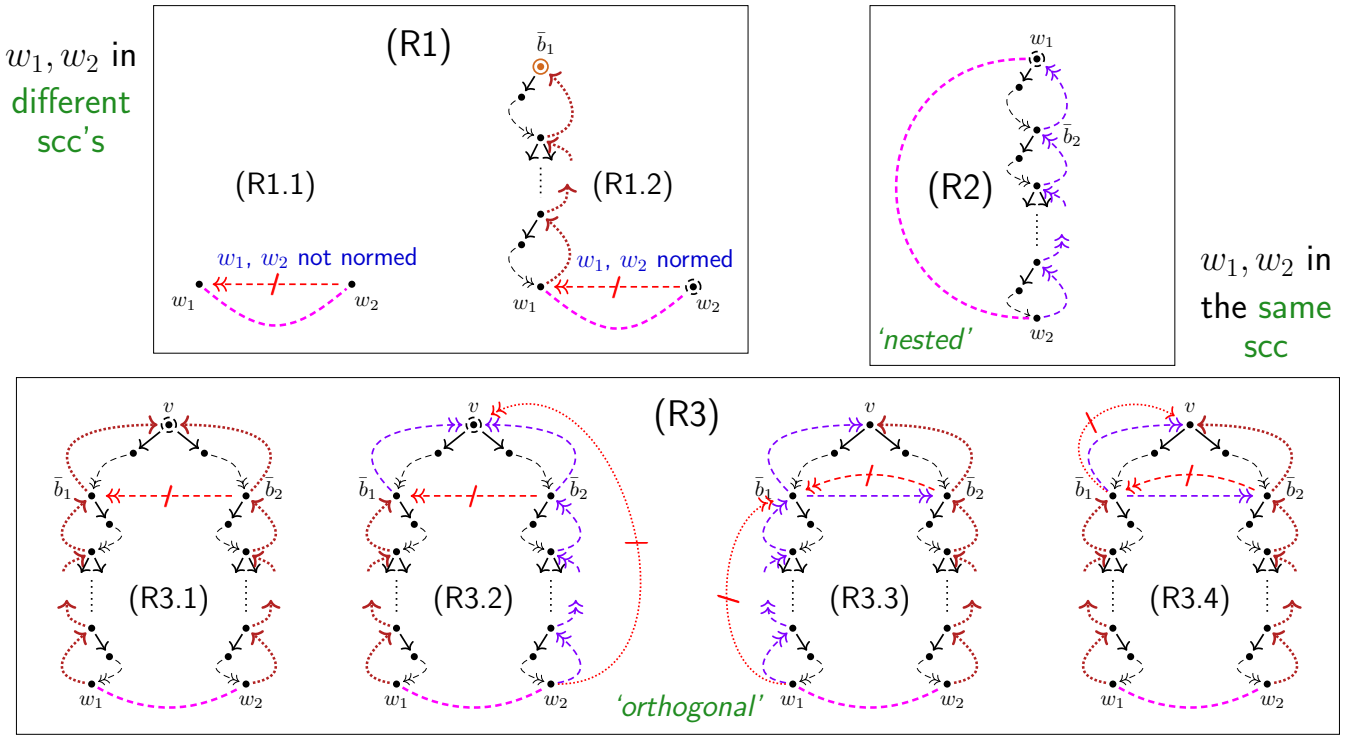
1-Graphs with LLEE $\triangleq$ (induced) graphs with 1-LLEE



$\underline{P}((a \cdot 1 + b \cdot (c^* \cdot 1))^*)$ $\leftrightarrow$

$\underline{P}\big(\big(a \cdot (1 \cdot ((a \cdot 1)^* \cdot 1))$
$\cdot\, b \cdot ((1 \cdot (c \cdot 1)^*)) \cdot 1$
$+\, b \cdot ((1 \cdot (c \cdot 1)^*)) \cdot 1\big)^*\big)$

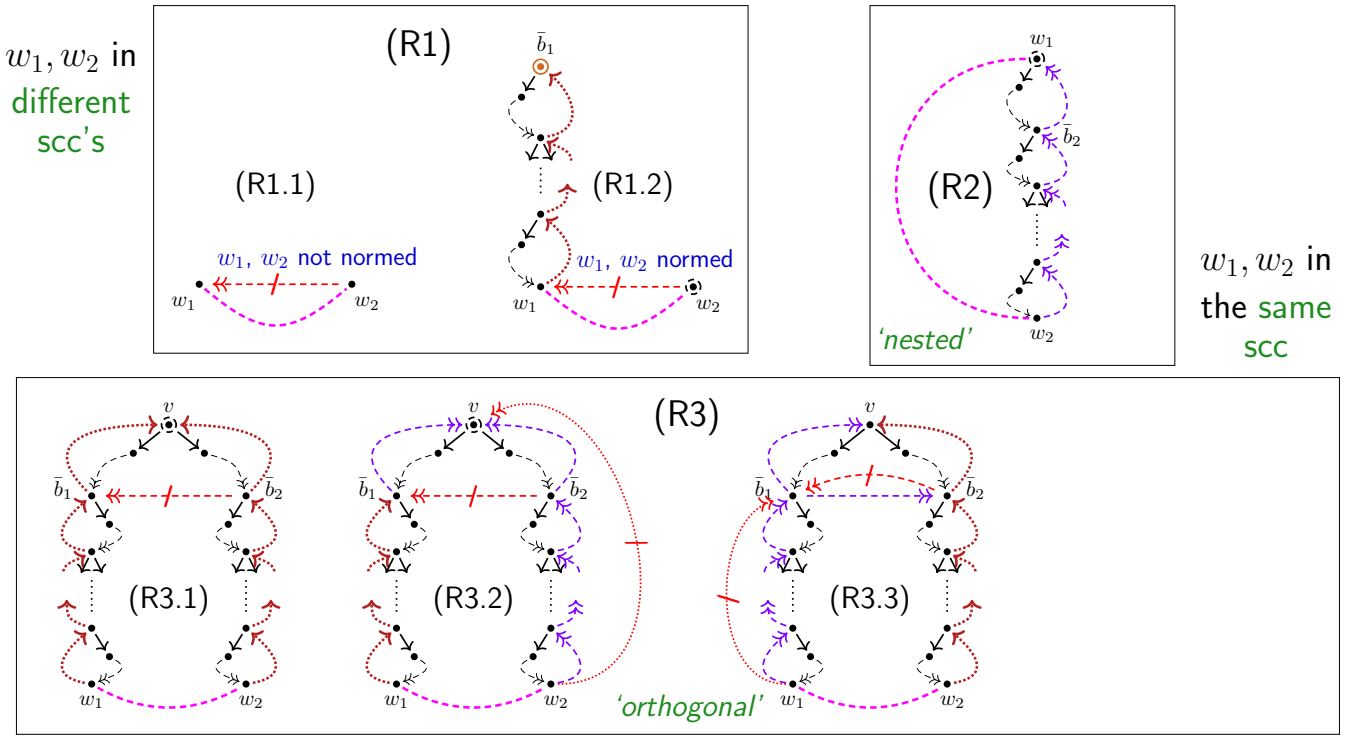
Reduced 1-bisimilarity redundancies in LLEE-1-graphs



Lemma

Every not collapsed LLEE-1-graph contains 1-bisimilar vertices $w_1 \neq w_2$ (a reduced 1-bisimilarity redundancy $\langle w_1, w_2 \rangle$) of kind (R1), (R2), (R3).

Reduced 1-bisimilarity redundancies in LLEE-1-graphs



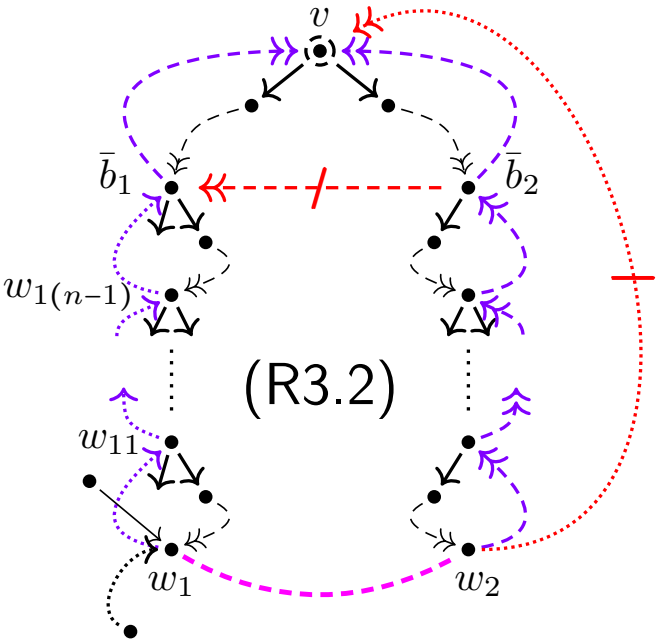
Lemma

Every not collapsed LLEE-1-graph contains 1-bisimilar vertices $w_1 \neq w_2$ (a reduced 1-bisimilarity redundancy $\langle w_1, w_2 \rangle$) of kind (R1), (R2), (R3).

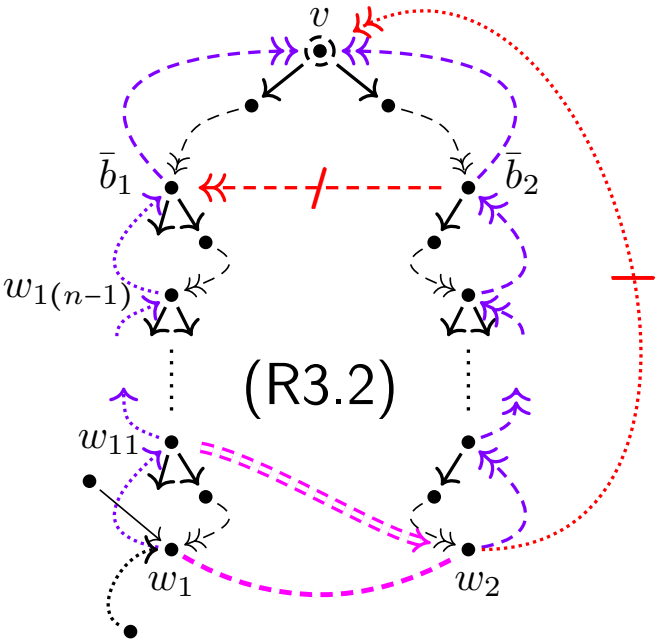
Lemma

Every simple reduced 1-bisimilarity redundancies in a LLEE-1-graph can be eliminated LLEE-preservingly.

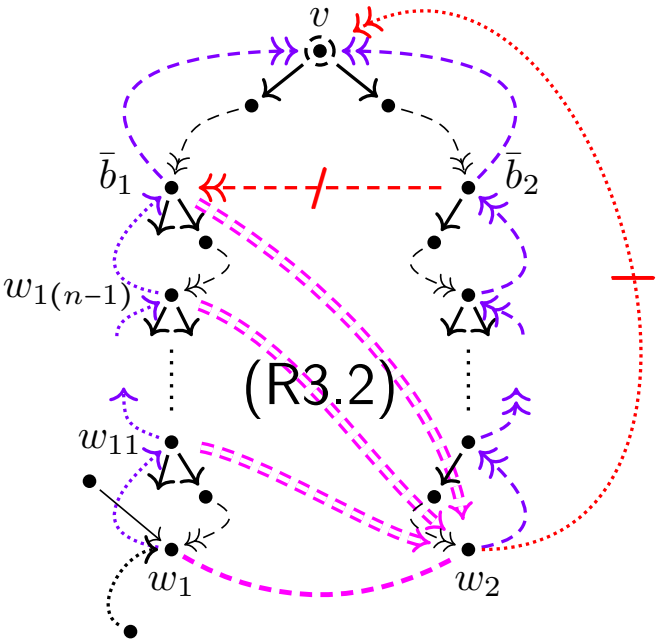
Elimination step of bisimilarity redundancies (R3.2)



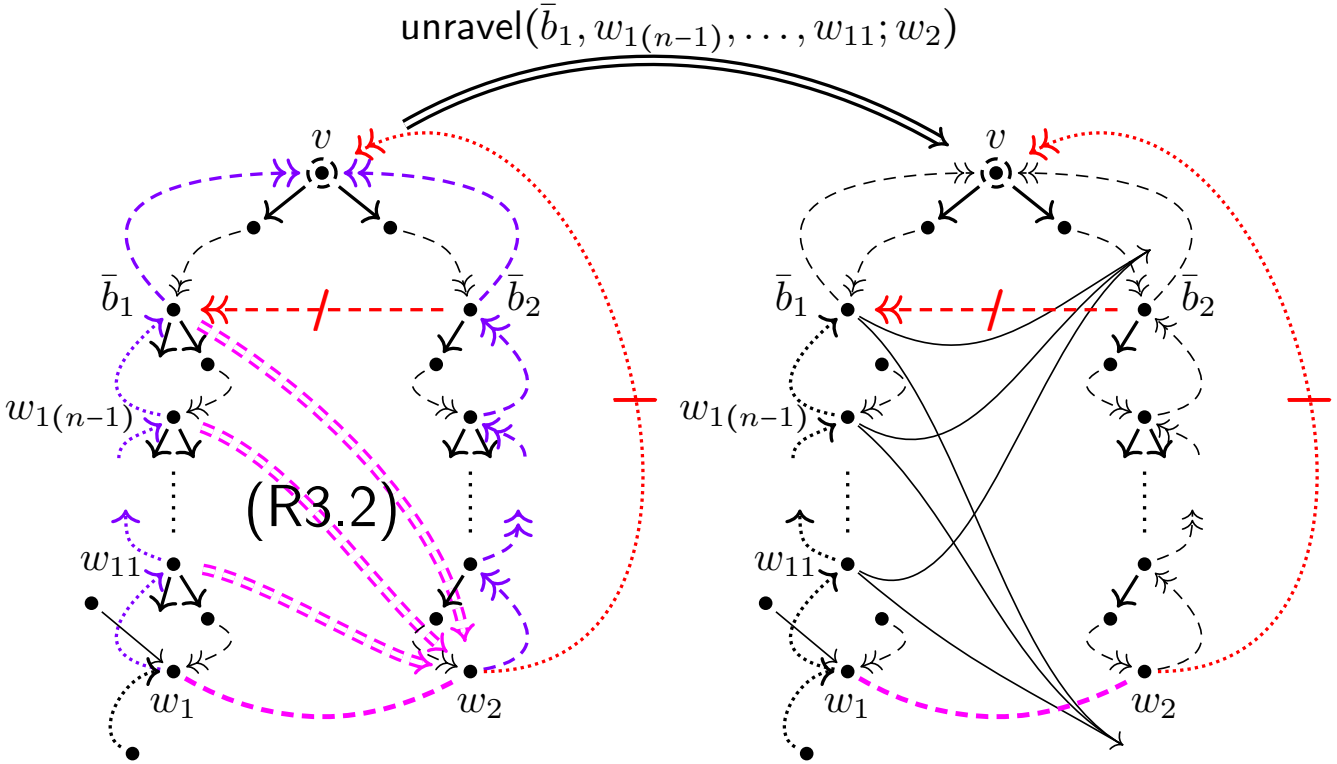
Elimination step of bisimilarity redundancies (R3.2)



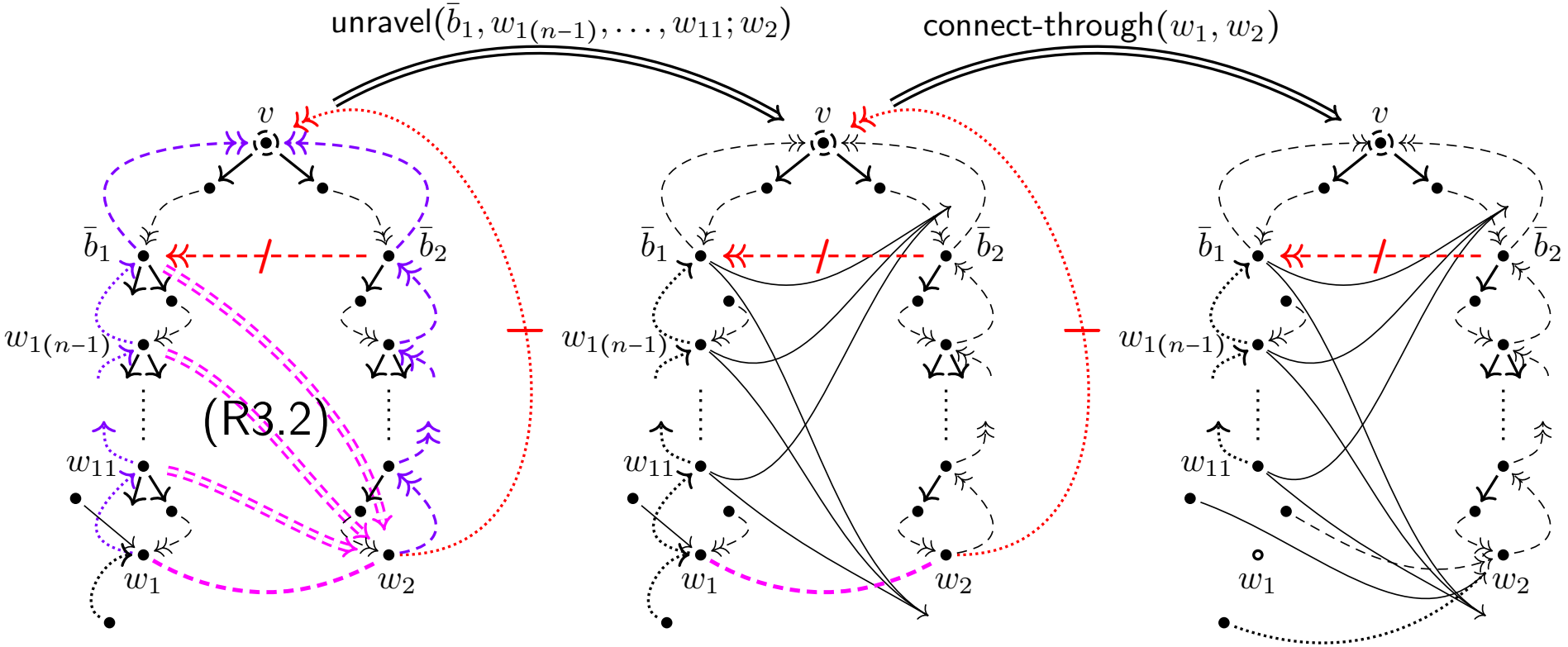
Elimination step of bisimilarity redundancies (R3.2)



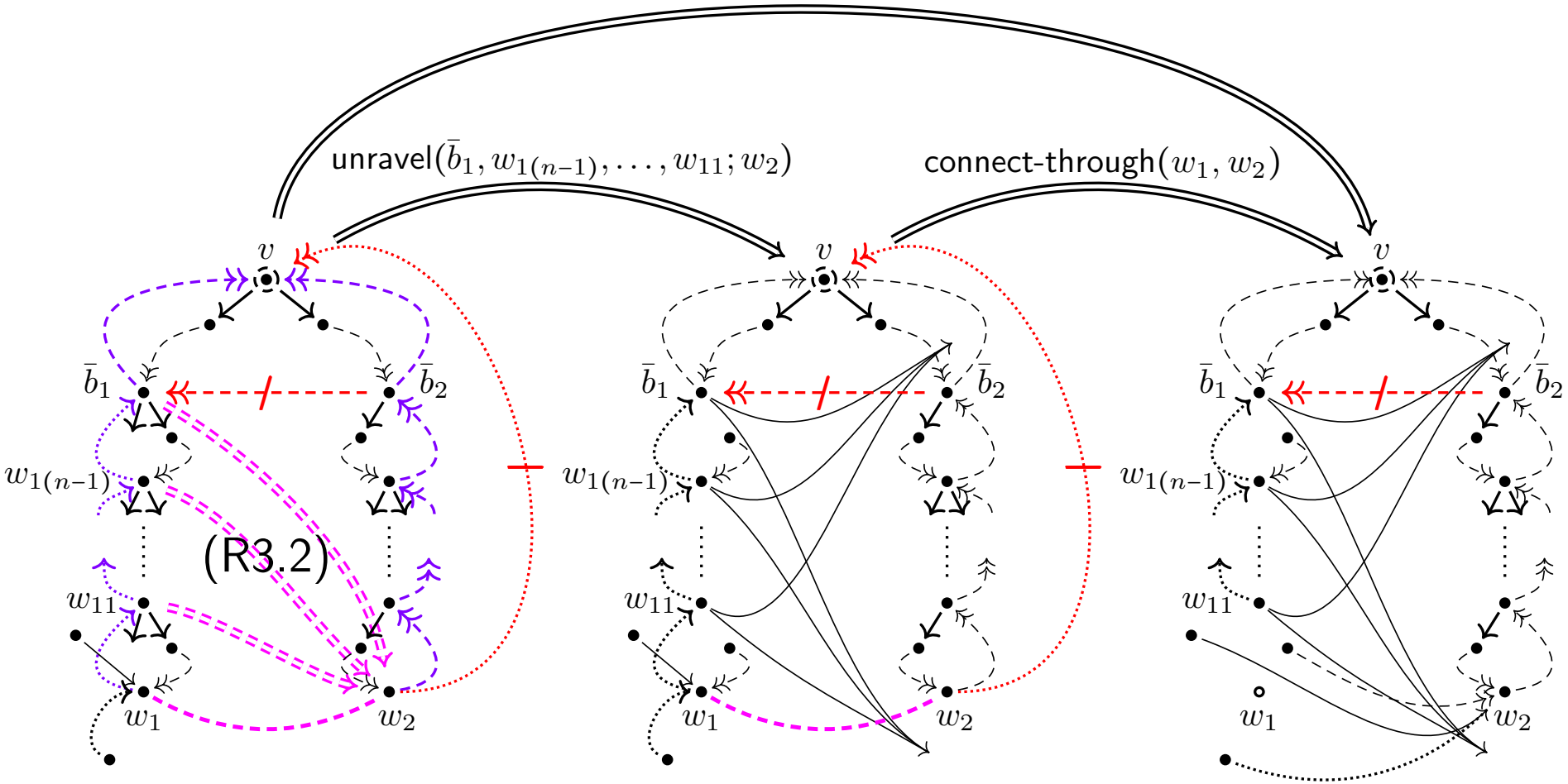
Elimination step of bisimilarity redundancies (R3.2)



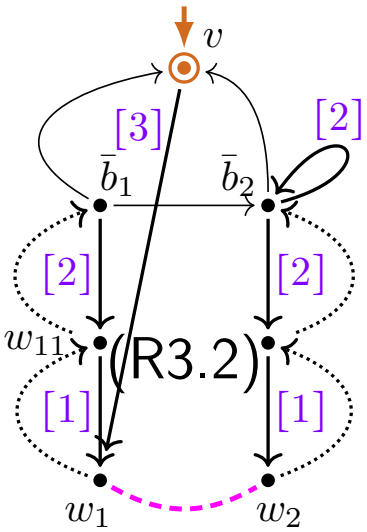
Elimination step of bisimilarity redundancies (R3.2)



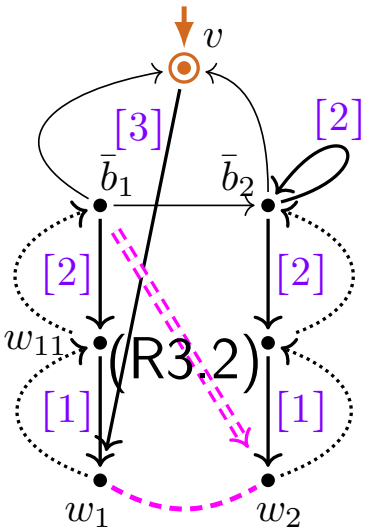
Elimination step of bisimilarity redundancies (R3.2)



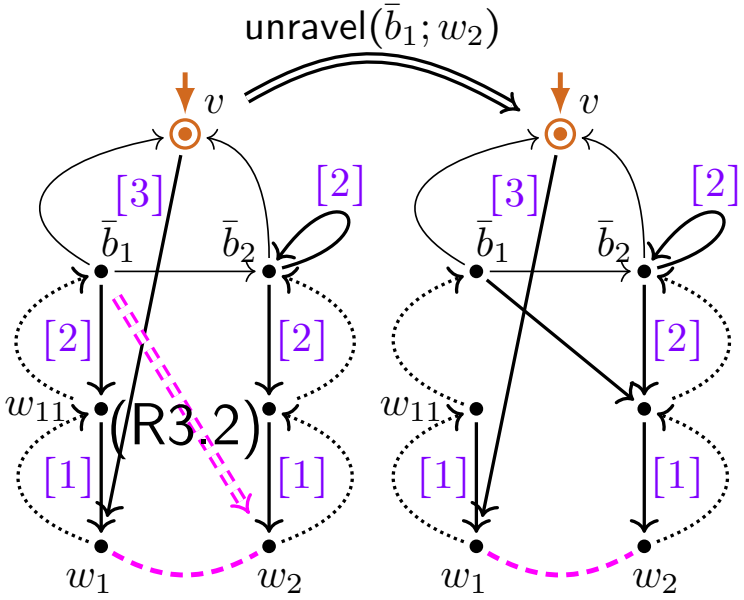
Elimination step of bisim. redundancy (R3.2) (example)



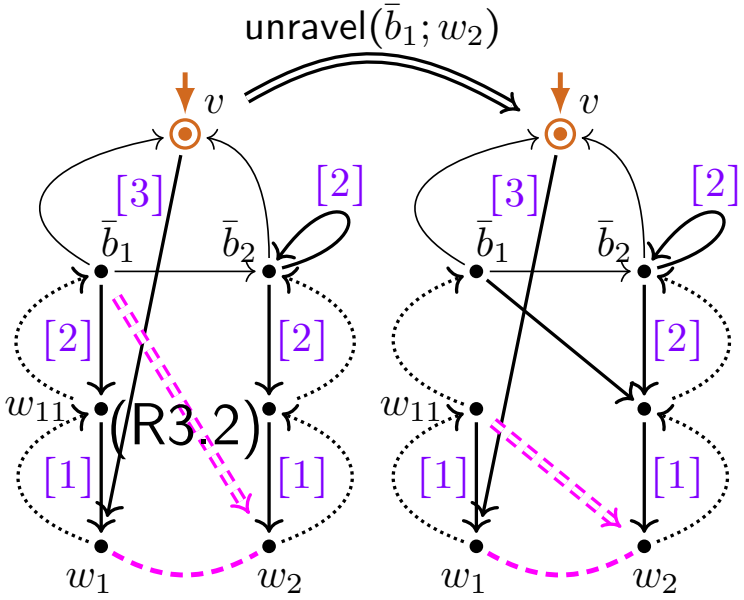
Elimination step of bisim. redundancy (R3.2) (example)



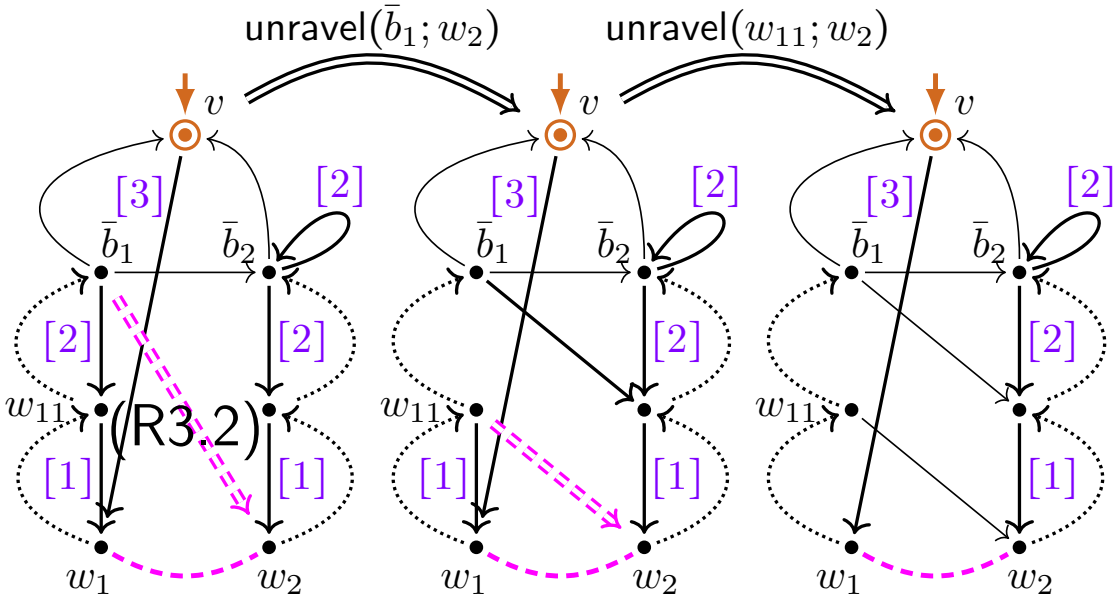
Elimination step of bisim. redundancy (R3.2) (example)



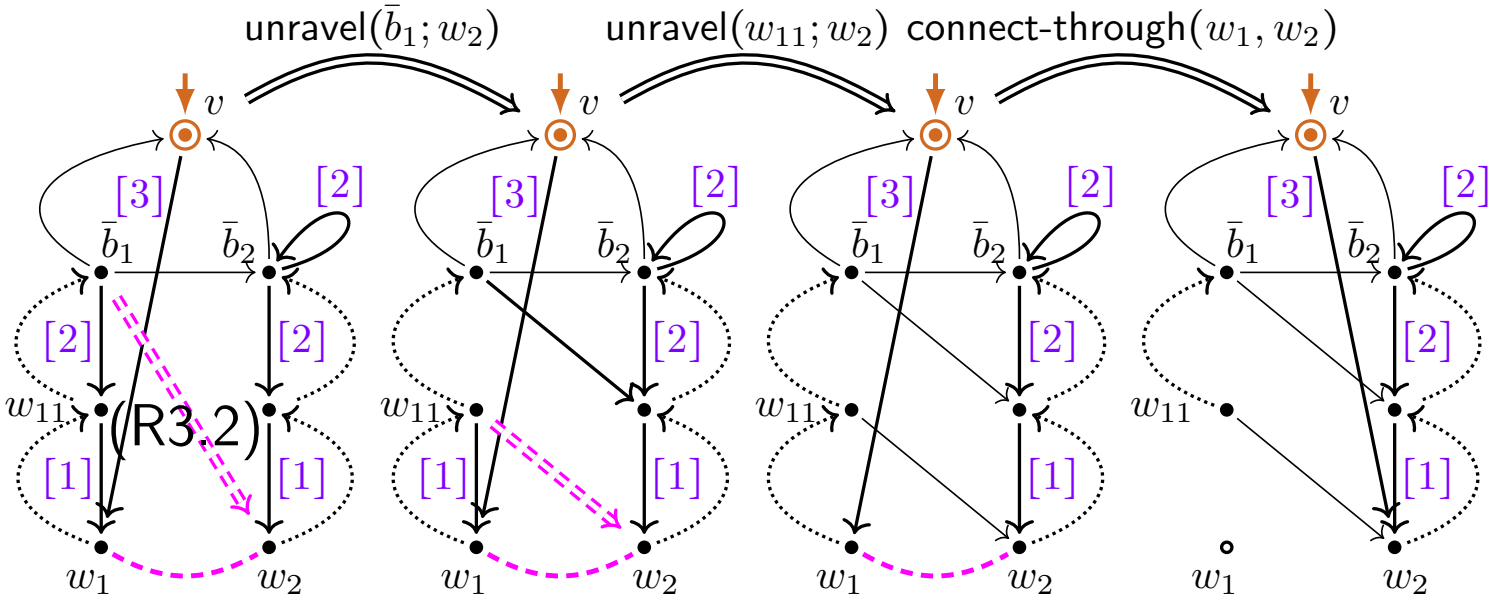
Elimination step of bisim. redundancy (R3.2) (example)



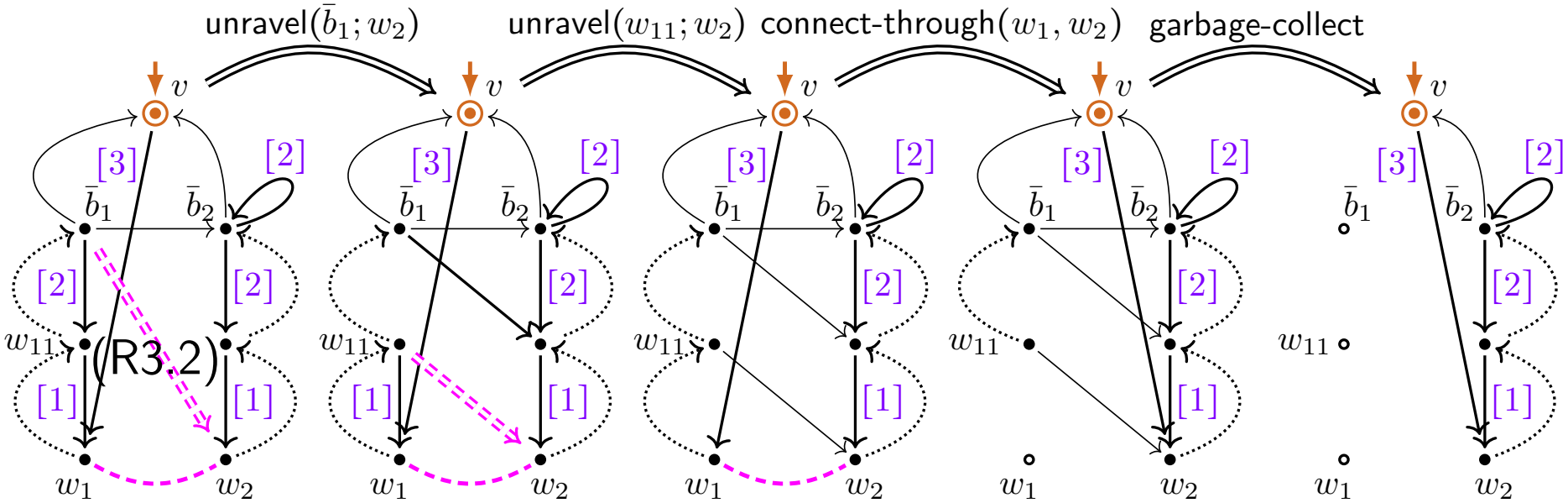
Elimination step of bisim. redundancy (R3.2) (example)



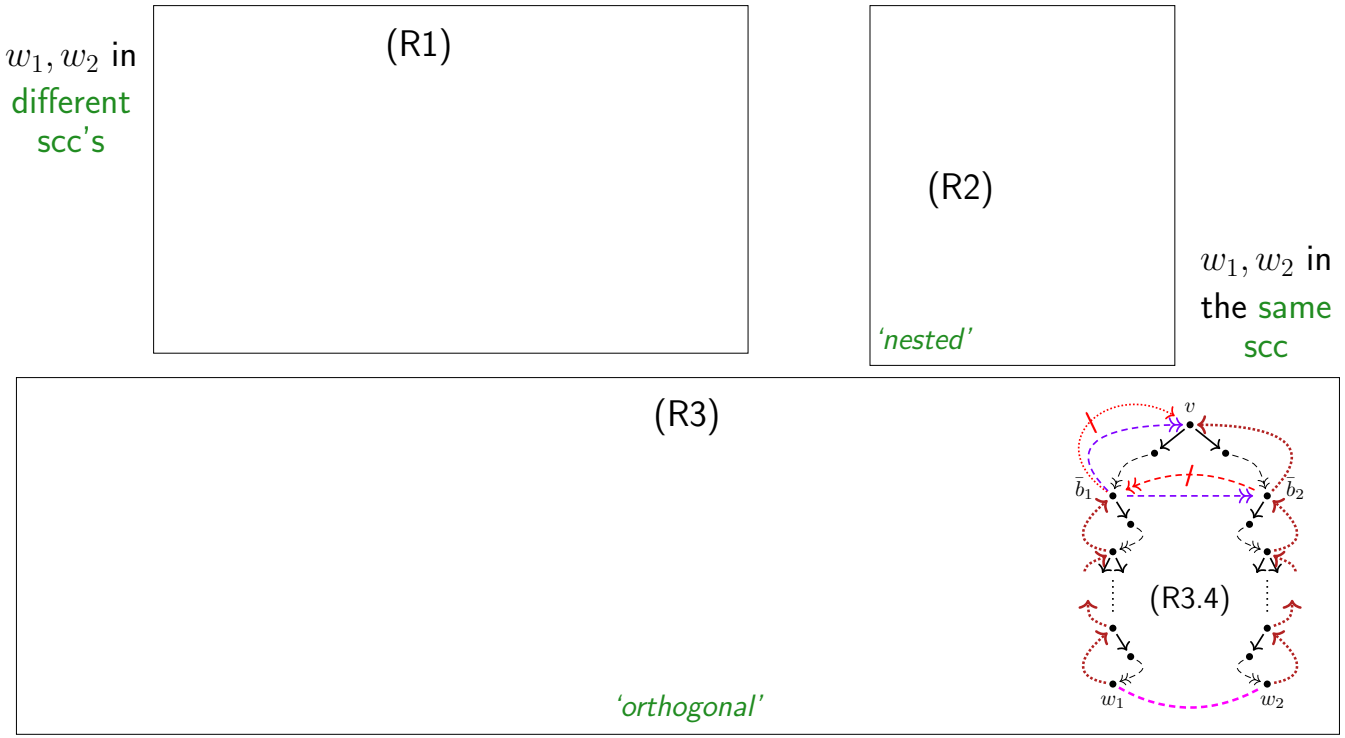
Elimination step of bisim. redundancy (R3.2) (example)



Elimination step of bisim. redundancy (R3.2) (example)



Reduced 1-bisimilarity redundancies in LLEE-1-graphs



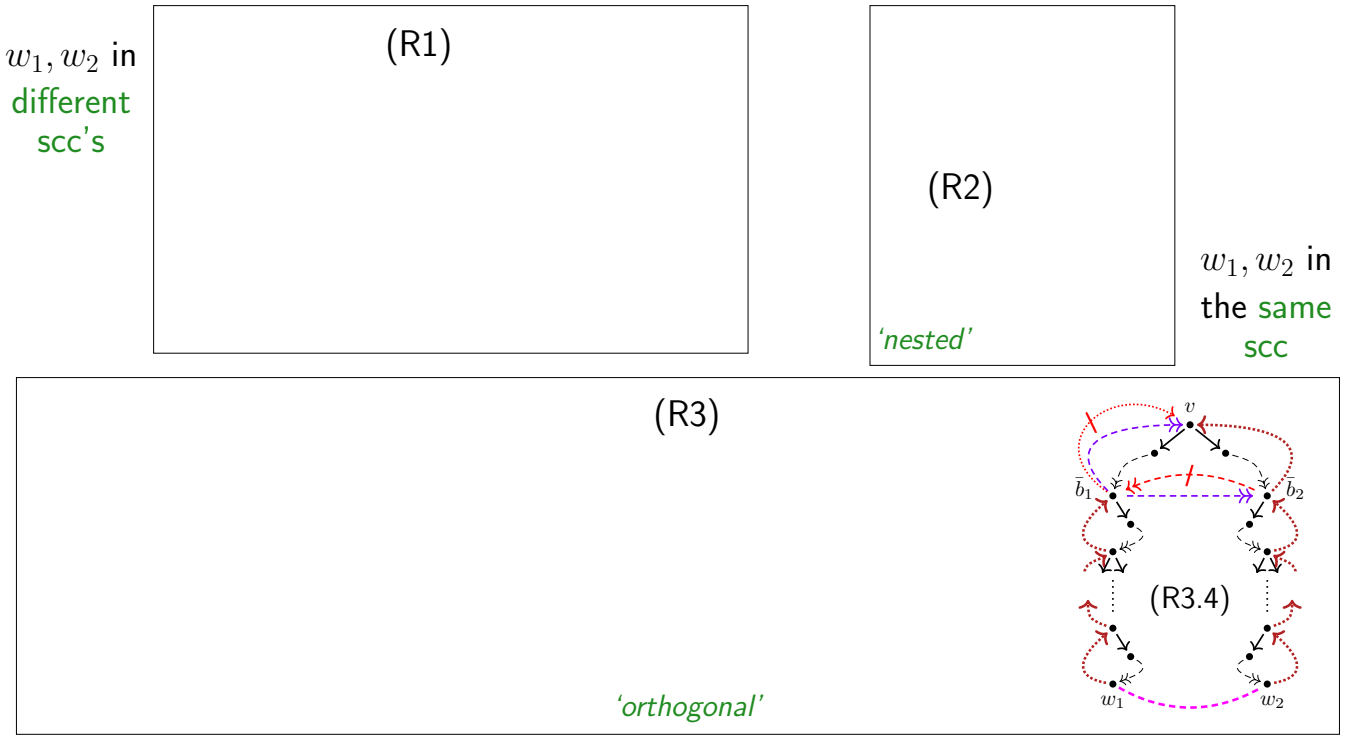
Lemma

Every not collapsed LLEE-1-graph contains 1-bisimilar vertices $w_1 \neq w_2$ (a reduced 1-bisimilarity redundancy $\langle w_1, w_2 \rangle$) of kind (R1), (R2), (R3).

Stumbling Block

How to LLEE-preservingly eliminate
reduced 1-bisimilarity redundancies of kind (R3.4)?

Reduced 1-bisimilarity redundancies in LLEE-1-graphs



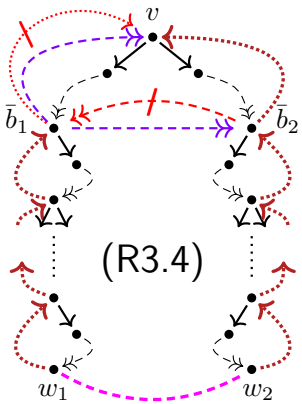
Lemma

Every not collapsed LLEE-1-graph contains 1-bisimilar vertices $w_1 \neq w_2$ (a reduced 1-bisimilarity redundancy $\langle w_1, w_2 \rangle$) of kind (R1), (R2), (R3).

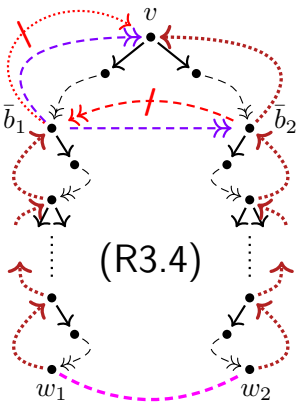
Stumbling Block

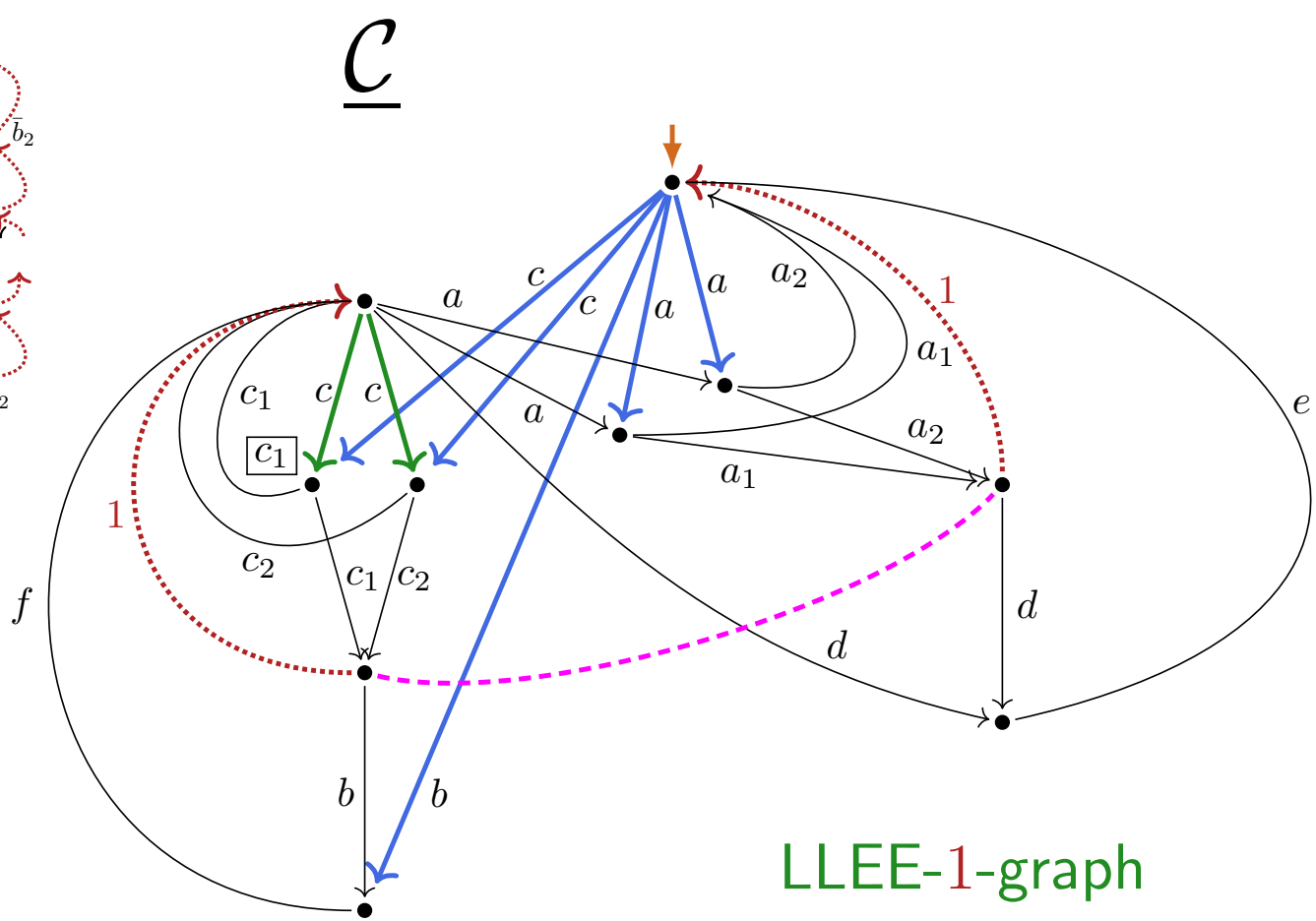
How to LLEE-preservingly eliminate
precrystalline reduced 1-bisimilarity redundancies?

Counterexample LLEE-preserving collapse



Counterexample LLEE-preserving collapse

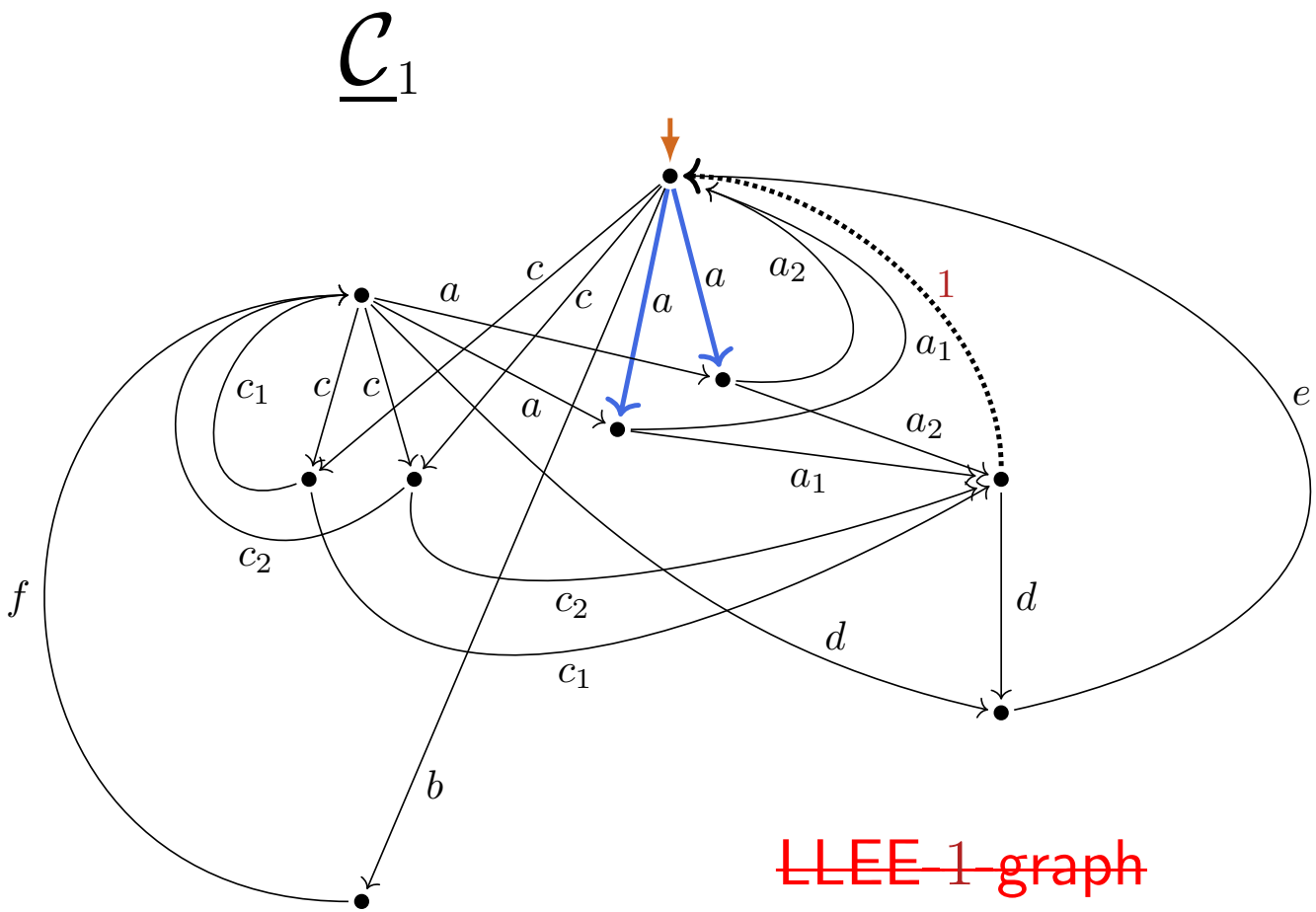




LLEE-1-graph

$\langle w_1, w_2 \rangle$ is a reduced
1-bisimilarity redundancy of kind (R3.4)

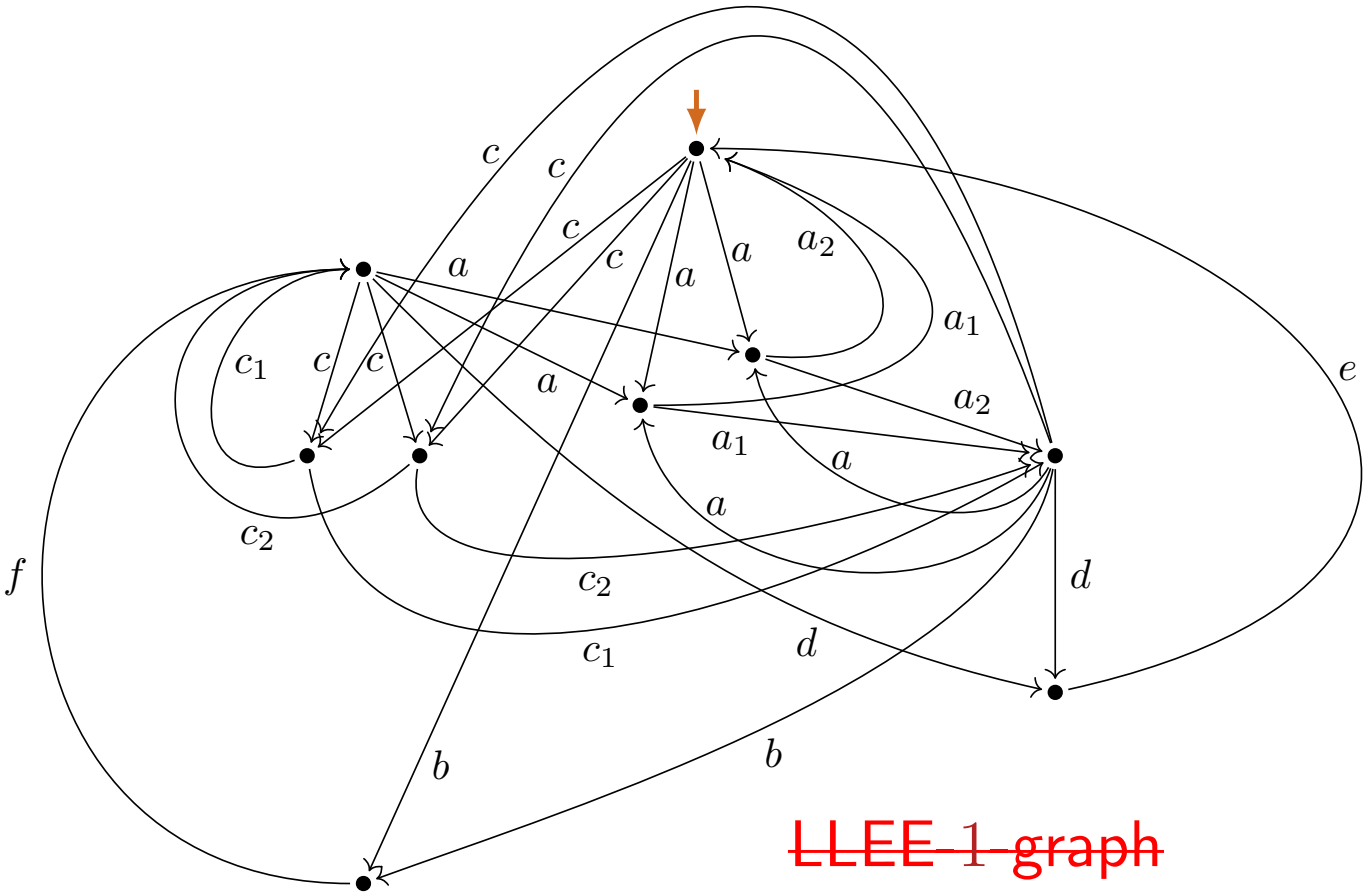
Counterexample LLEE-preserving collapse



~~LLEE 1-graph~~

1-bisimulation collapse

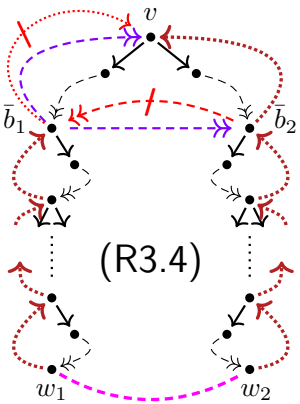
Counterexample LLEE-preserving collapse

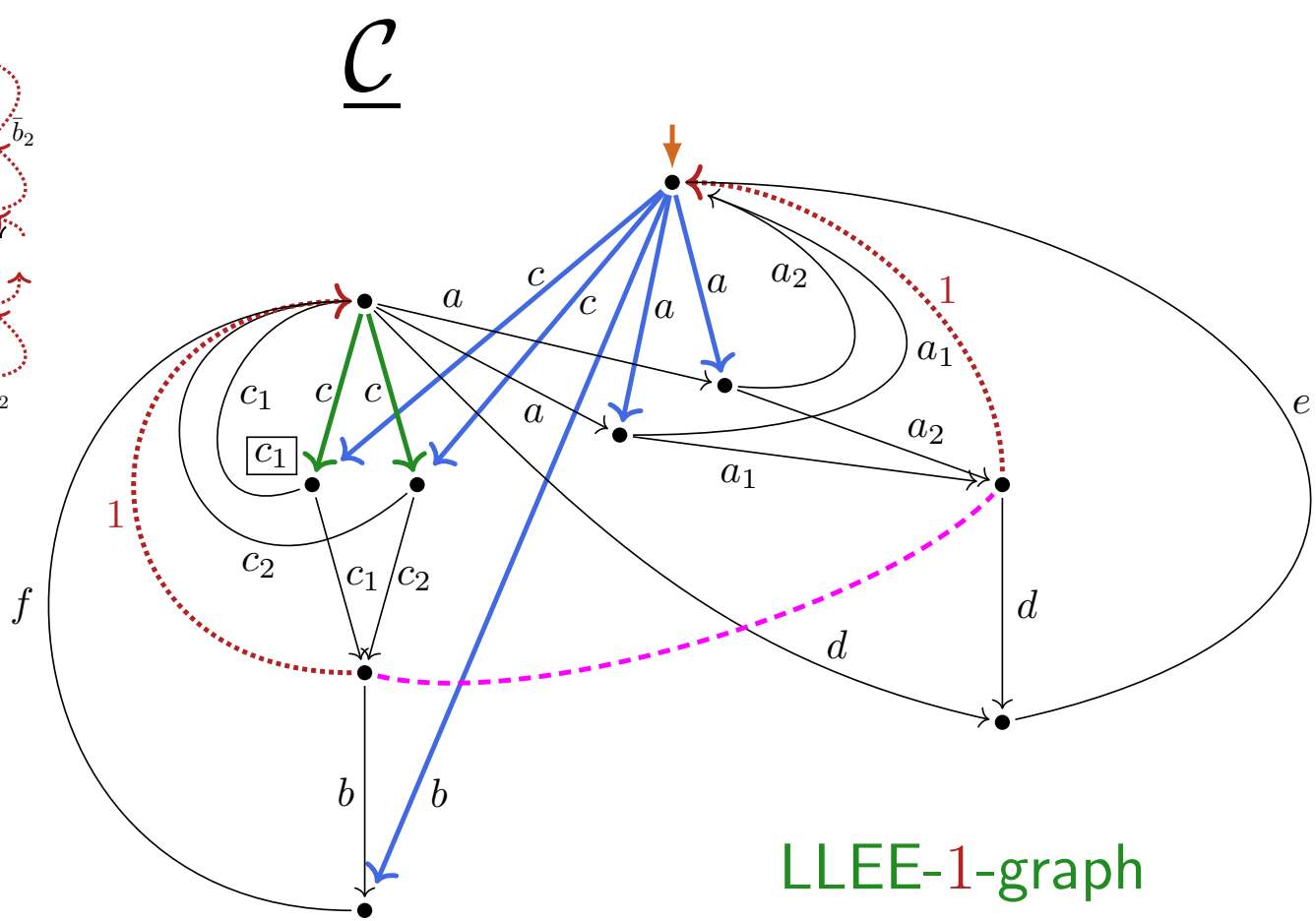


~~LLEE 1-graph~~

bisimulation collapse

Counterexample LLEE-preserving collapse

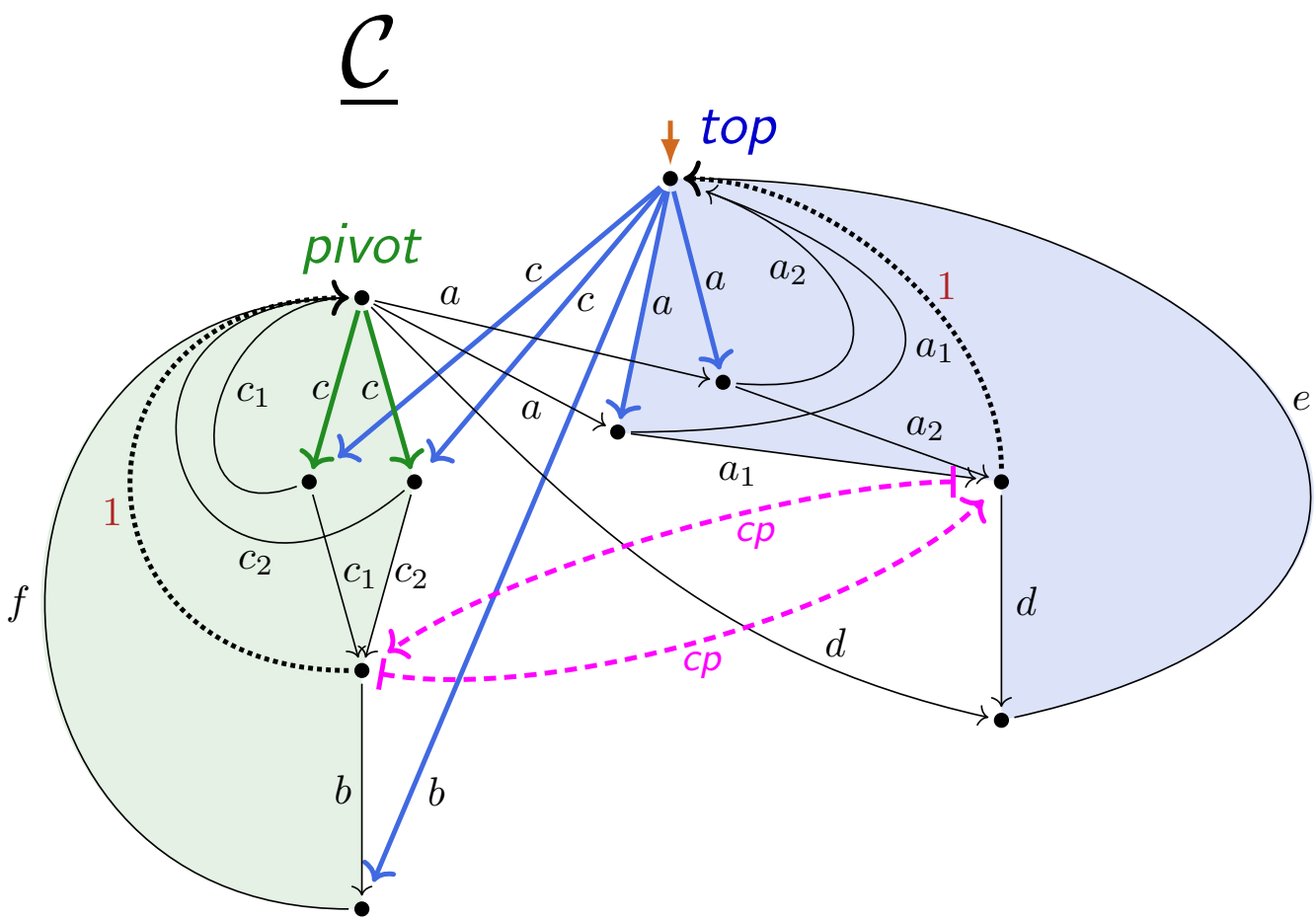




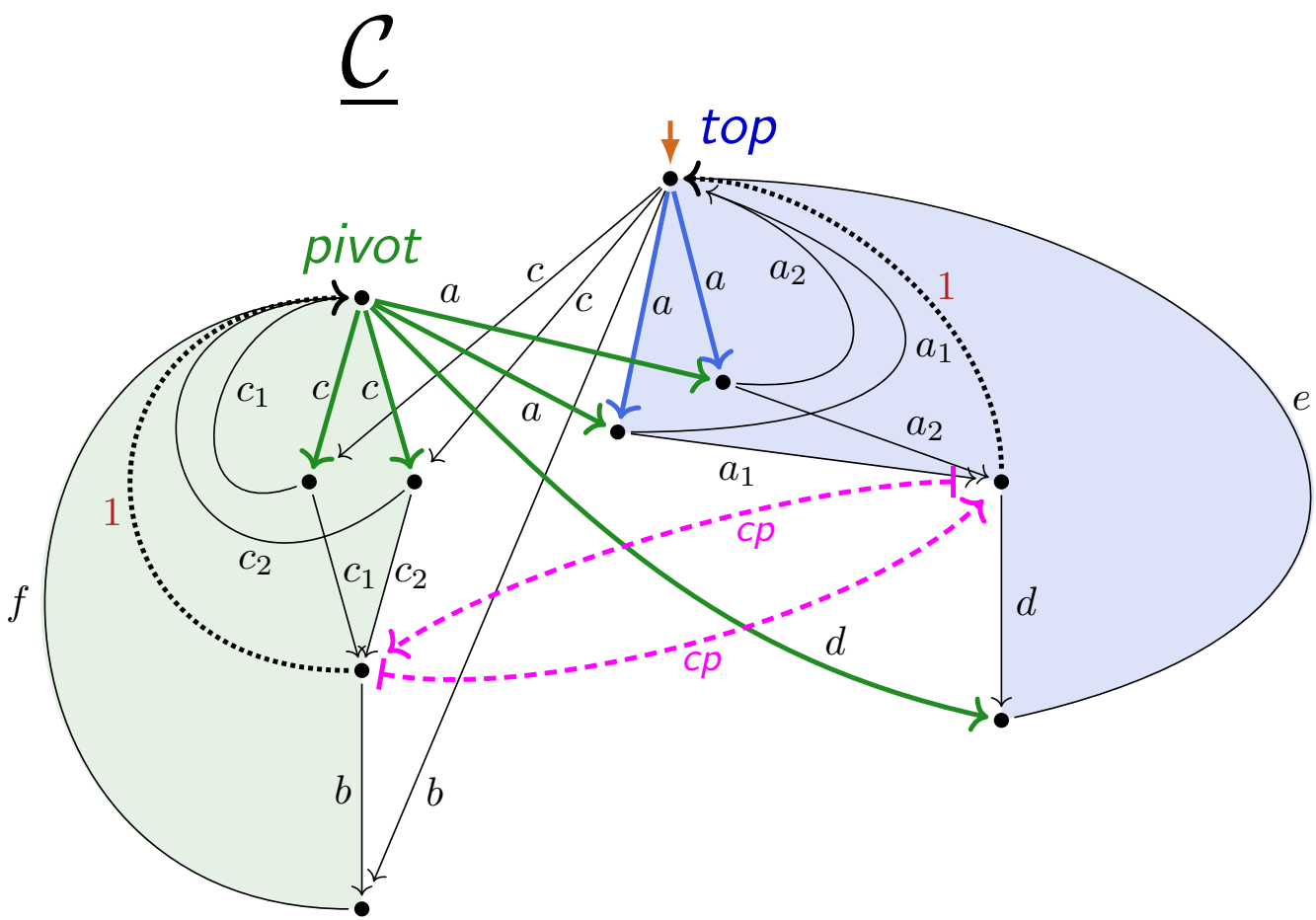
LLEE-1-graph

$\langle w_1, w_2 \rangle$ is a reduced
1-bisimilarity redundancy of kind (R3.4)

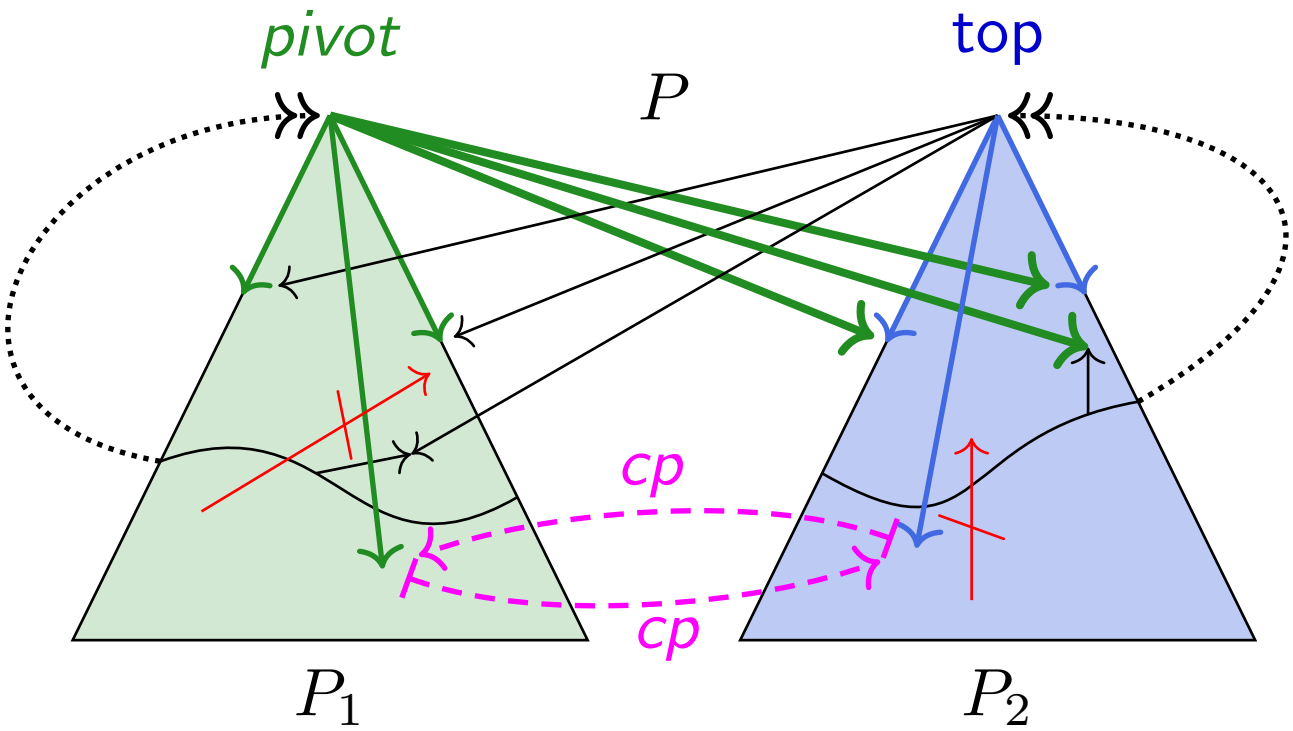
Twin-Crystal



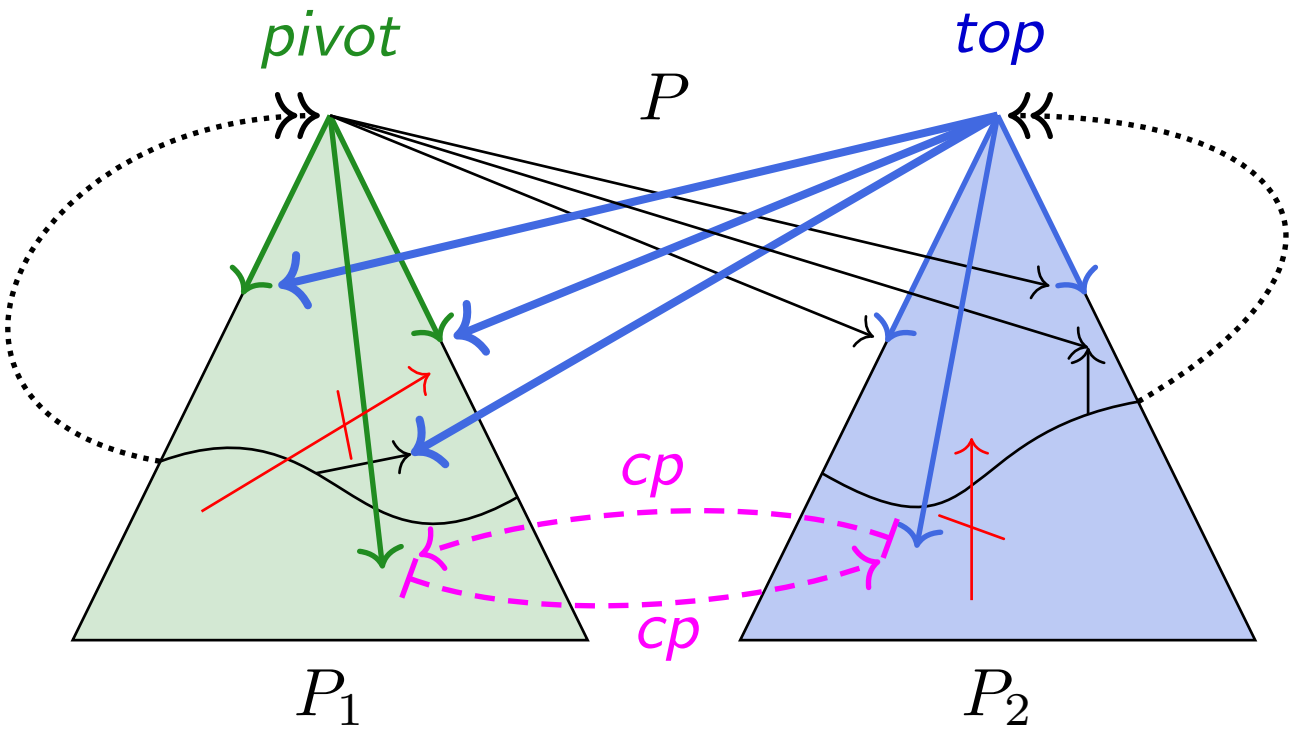
Twin-Crystal



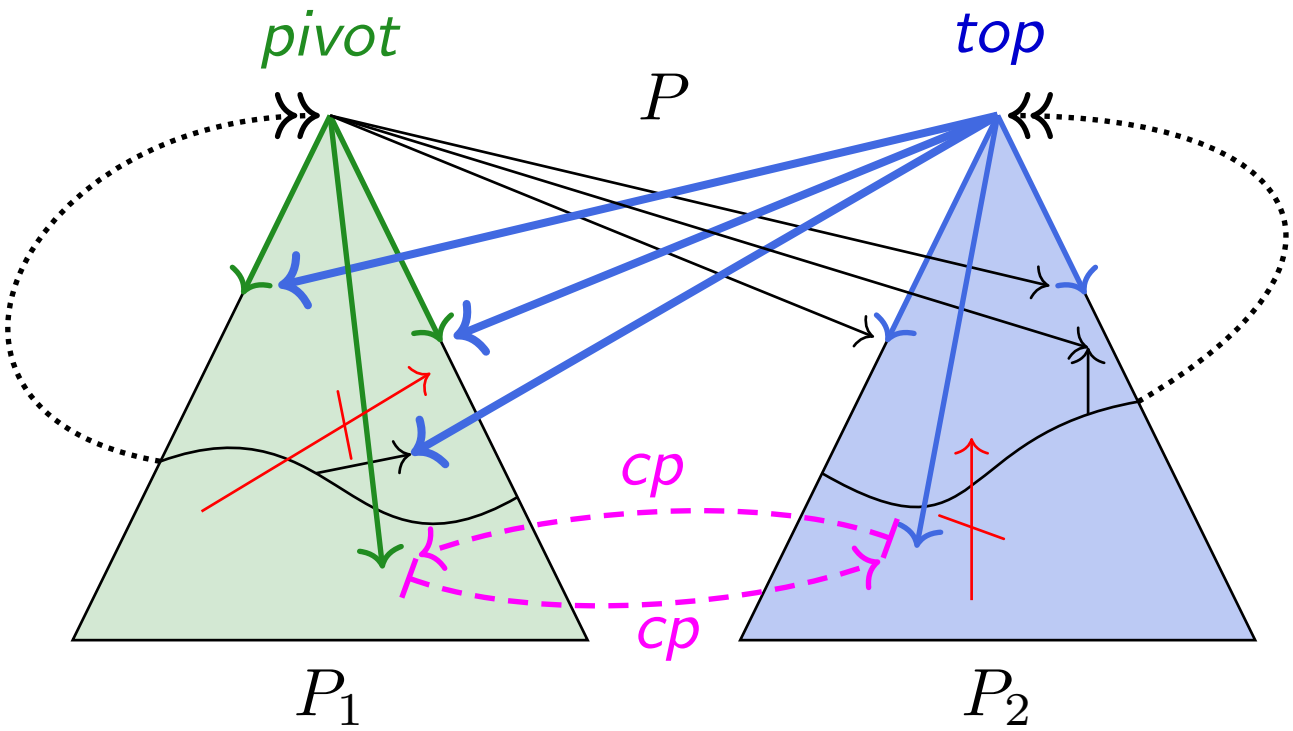
Twin-Crystal



Twin-Crystal

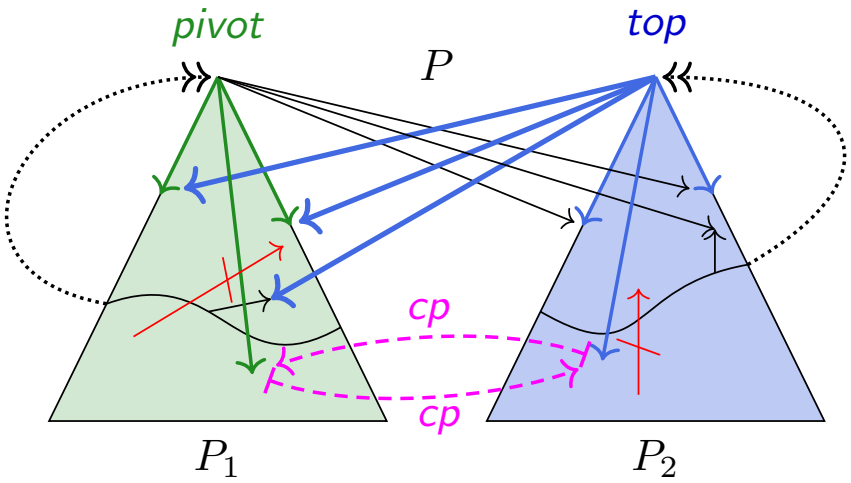


Twin-Crystal



- Provable solutions of twin-crystals are complete:
  they can be transferred to their bisimulation collapses

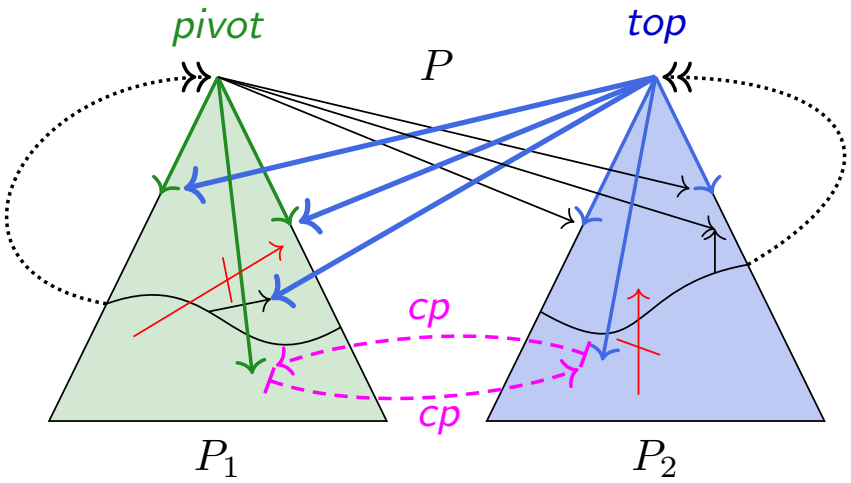
Crystallization



twin-crystal

Crystallized 1-graphs = LLEE-1-graphs that are collapsed apart from some scc's that are twin-crystals.

Crystallization

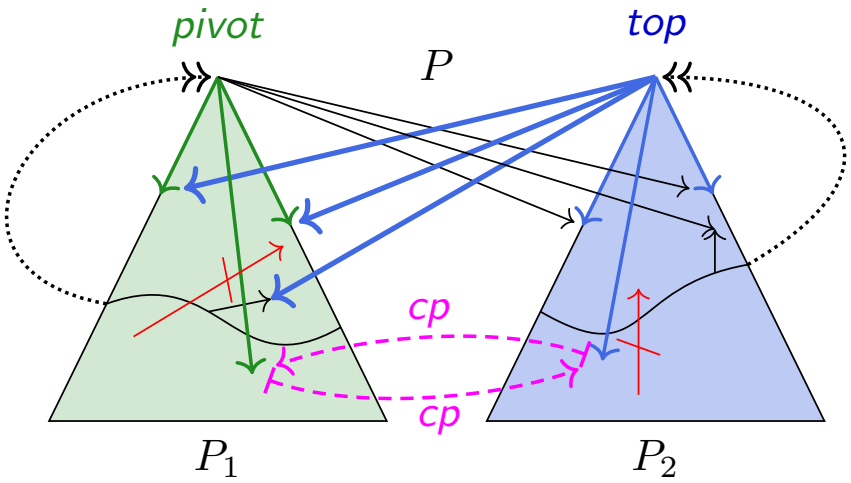


twin-crystal

Crystallized 1-graphs = LLEE-1-graphs that are collapsed apart from some scc's that are twin-crystals.

(CR) Crystallization: Every LLEE-1-graph can be reduced under bisimilarity to a bisimilar crystallized 1-graph.

Crystallization



twin-crystal

Crystallized 1-graphs = LLEE-1-graphs that are collapsed apart from some scc's that are twin-crystals.

(CR) Crystallization: Every LLEE-1-graph can be reduced under bisimilarity to a bisimilar crystallized 1-graph.

(CC) Every provable solution of a crystallized 1-graph gives rise to provable solution on the bisimulation collapse.

Results facilitated by steps (st3)

Proposition

The property 1-LLEE is not preserved under bisimulation collapse.

Results facilitated by steps (st3)

Proposition

The property 1-LLEE is not preserved under bisimulation collapse.

Crystallisation Lemma for LLEE-1-graphs

Every 1-graph $\underline{G}$ with LLEE can be pseudo-collapsed/crystallized by elimination steps for bisimilarity redundancies of kinds (R1), (R2), (R3.1), (R3.2), (R3.3) (and related steps) to a 1-graph $\underline{G_0}$ such that:

- $\underline{G_0} \leftrightarrows \underline{G}$,
- $\underline{G_0}$ is crystallised: all strongly connected components are collapsed or twin-crystals.

Results facilitated by steps (st3)

Proposition

The property 1-LLEE is not preserved under bisimulation collapse.

Crystallisation Lemma for LLEE-1-graphs

Every 1-graph $\underline{G}$ with LLEE can be pseudo-collapsed/crystallized by elimination steps for bisimilarity redundancies of kinds (R1), (R2), (R3.1), (R3.2), (R3.3) (and related steps) to a 1-graph $\underline{G}_0$ such that:

- $\underline{G}_0 \leftrightarrows \underline{G}$,
- $\underline{G}_0$ is crystallised: all strongly connected components are collapsed or twin-crystals.

Crystallisation facilitates to adapt the completeness proof for BBP (that uses preservation of LLEE under collapse) via a crystallisation detour.

Results facilitated by steps (st3)

Proposition

The property 1-LLEE is not preserved under bisimulation collapse.

Crystallisation Lemma for LLEE-1-graphs

Every 1-graph $\underline{G}$ with LLEE can be pseudo-collapsed/crystallized by elimination steps for bisimilarity redundancies of kinds (R1), (R2), (R3.1), (R3.2), (R3.3) (and related steps) to a 1-graph $\underline{G}_0$ such that:

- $\underline{G}_0 \leftrightarrows \underline{G}$,
- $\underline{G}_0$ is crystallised: all strongly connected components are collapsed or twin-crystals.

Crystallisation facilitates to adapt the completeness proof for BBP (that uses preservation of LLEE under collapse) via a crystallisation detour.

Theorem *(LICS 2022)*

Milner's proof system Mil is complete.

Interpretation-readback correspondences

- Regular expressions (also: 1-free/under-star-1-free $(*/\not{1})$ expr.s)
- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility questions)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Small steps for hard questions

  (st1) reduction steps for well-behaved 1-graphs under bisimilarity
  - $[\![\cdot]\!]_P$-expressibility is decidable

  (st2) collapse steps for LLEE graphs under bisimilarity
  - completeness for $(*/\not{1})$ expressions

  (st3) collapse steps for LLEE 1-graphs under bisimilarity, as far as possible
  - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences

  (irc) LLEE/1-LLEE graphs $\hat{=}$ image of $P^\bullet$ restricted to $(*/\not{1})$/all expr.s
  - where $P^\bullet$ a compact refinement of P
  - crucial for P-/$[\![\cdot]\!]_P$-expressiblity, and completeness questions
- Outlook

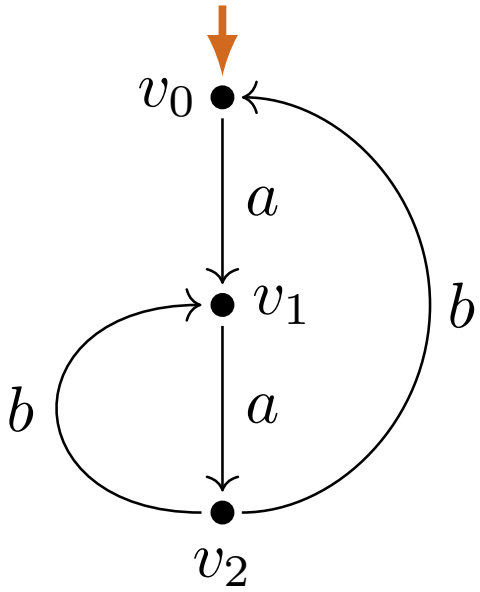
Readback

Lemma (~ *G–Fokkink, LICS 2020*)

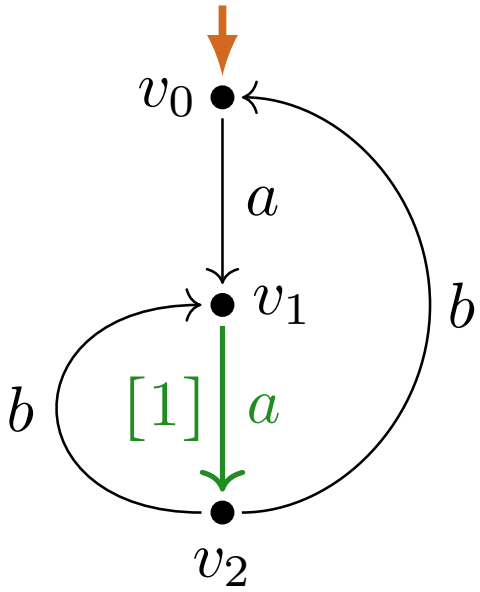
Process graphs G *with* LLEE *are* $[\![\cdot]\!]_P$*-expressible:*

$$\mathsf{LLEE}(G) \implies \exists\, e \in \mathit{RExp}\big(\, G \leftrightarrows P(e) \,\big)\,.$$

Readback from LLEE-witness (example)

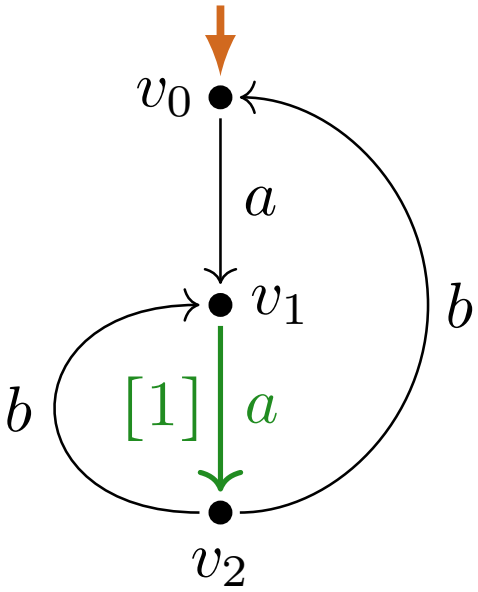


Readback from LLEE-witness (example)



layered
LEE-witness

Readback from LLEE-witness (example)

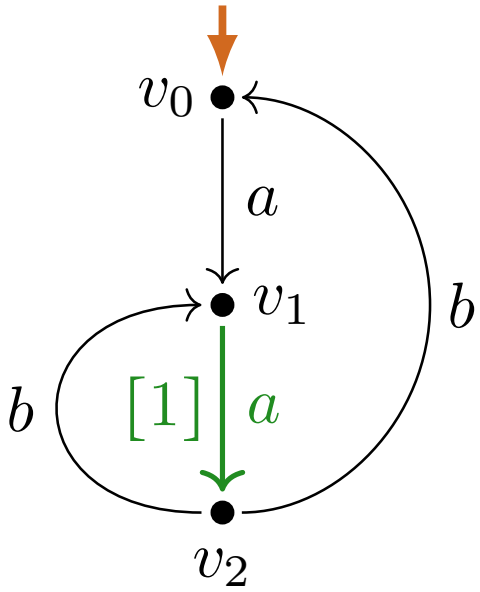


layered
LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

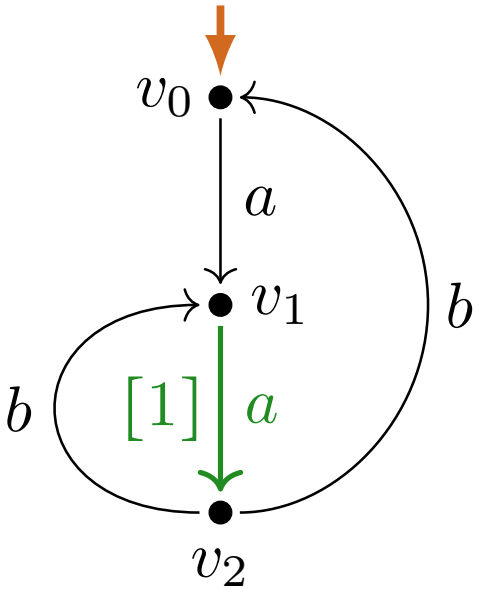
Readback from layered LEE-witness (example)

$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$



layered
LEE-witness

Readback from layered LEE-witness (example)

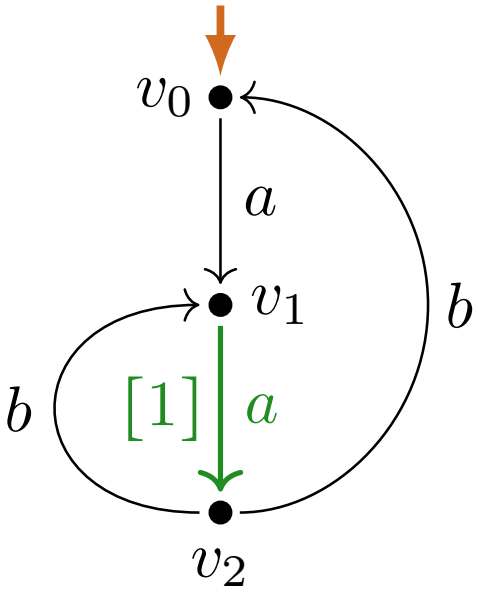


layered
LEE-witness

$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$

$$s(v_1) = \quad \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

Readback from layered LEE-witness (example)



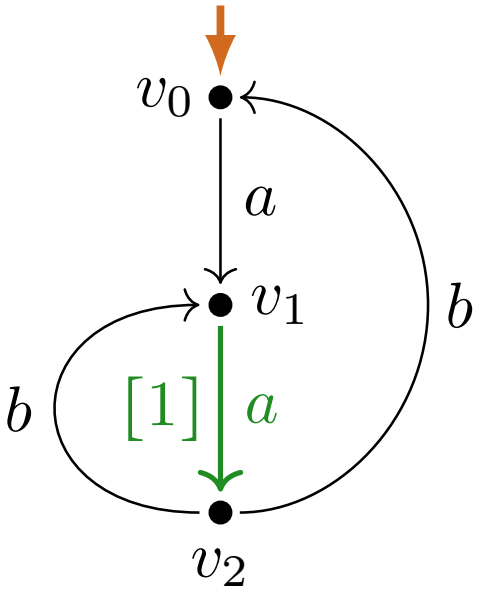
layered
LEE-witness

$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$

$$s(v_1) = \quad \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

$$s(v_2, v_1) = \quad 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big)$$

Readback from layered LEE-witness (example)



layered
LEE-witness

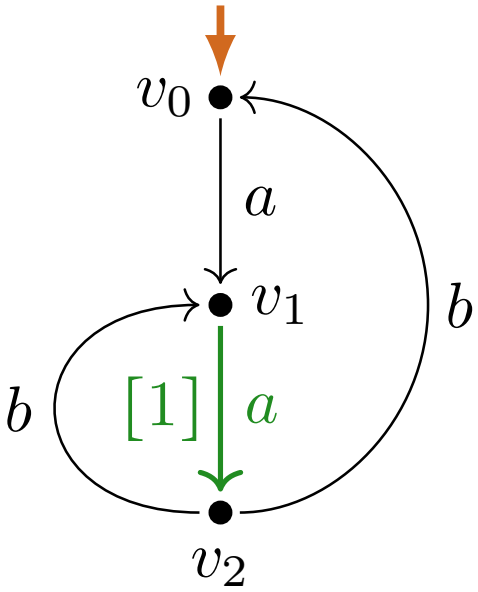
$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$

$$s(v_1) = \quad \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

$$s(v_2, v_1) = \quad 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big)$$

$$s(v_1, v_1) = \quad 1$$

Readback from layered LEE-witness (example)



layered
LEE-witness

$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$

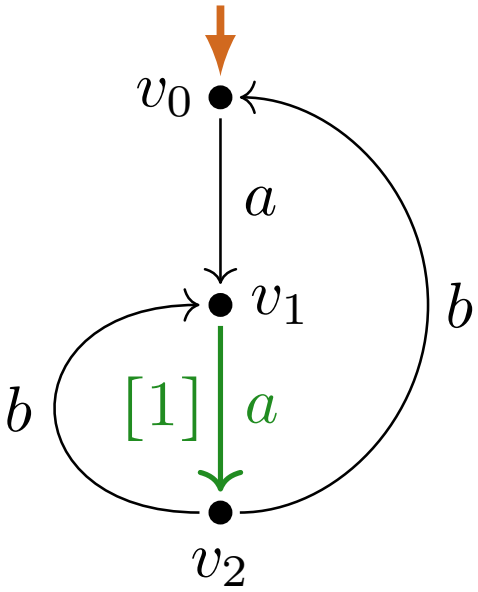
$$s(v_1) = \quad \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

$$s(v_2, v_1) = \quad 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big)$$

$$s(v_1, v_1) = \quad 1$$

$$s(v_0, v_1) = \quad 0^* \cdot a \cdot s(v_1, v_1)$$

Readback from layered LEE-witness (example)



layered
LEE-witness

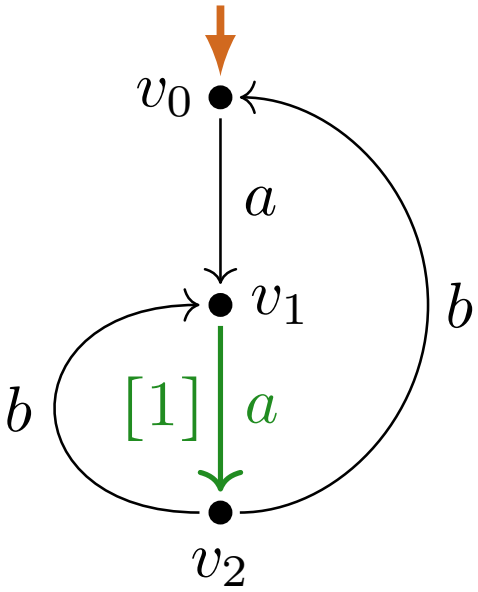
$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$

$$s(v_1) = \quad \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

$$s(v_2, v_1) = \quad 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big)$$

$$\begin{aligned} s(v_1, v_1) &= \quad 1 \\ s(v_0, v_1) &= \quad 0^* \cdot a \cdot s(v_1, v_1) \\ &= \quad 0^* \cdot a \cdot 1 \end{aligned}$$

Readback from layered LEE-witness (example)



layered LEE-witness

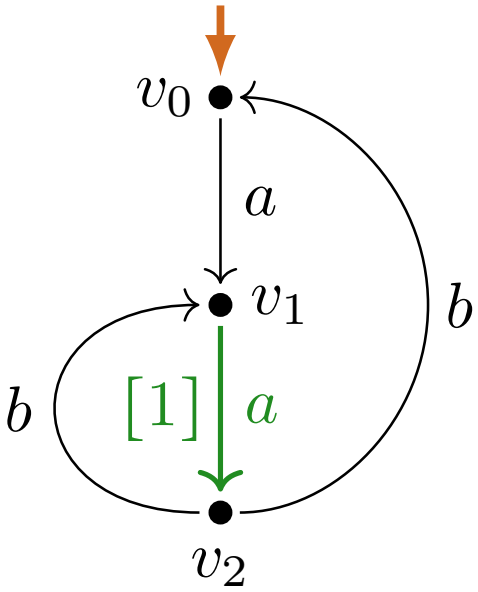
$$s(v_0) = \quad 0^* \cdot a \cdot s(v_1)$$

$$s(v_1) = \quad \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

$$s(v_2, v_1) = \quad 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big)$$

$$\begin{aligned} s(v_1, v_1) &= \quad 1 \\ s(v_0, v_1) &= \quad 0^* \cdot a \cdot s(v_1, v_1) \\ &= \quad 0^* \cdot a \cdot 1 \\ &=_{\text{Mil}^-} \quad a \end{aligned}$$

Readback from layered LEE-witness (example)



layered LEE-witness

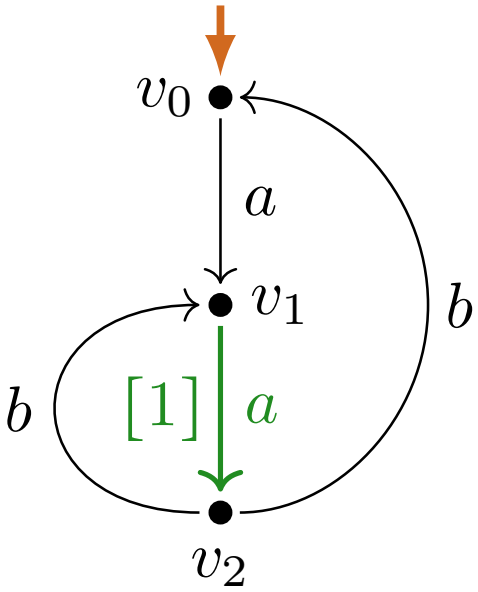
$$s(v_0) = 0^* \cdot a \cdot s(v_1)$$

$$s(v_1) = \big(a \cdot s(v_2, v_1)\big)^* \cdot 0$$

$$\begin{aligned} s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\ &=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \end{aligned}$$

$$\begin{aligned} s(v_1, v_1) &= 1 \\ s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\ &= 0^* \cdot a \cdot 1 \\ &=_{\text{Mil}^-} a \end{aligned}$$

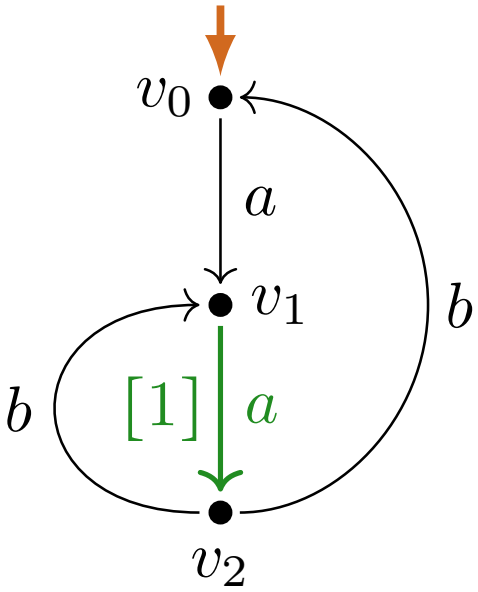
Readback from layered LEE-witness (example)



layered LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

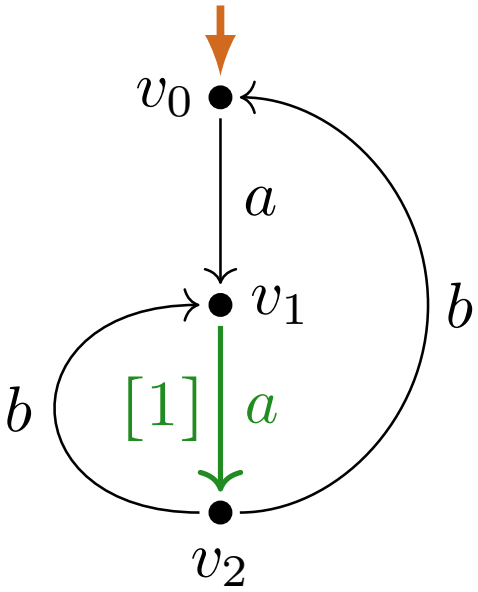
Readback from layered LEE-witness (example)



layered LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
\\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

Readback from layered LEE-witness (example)

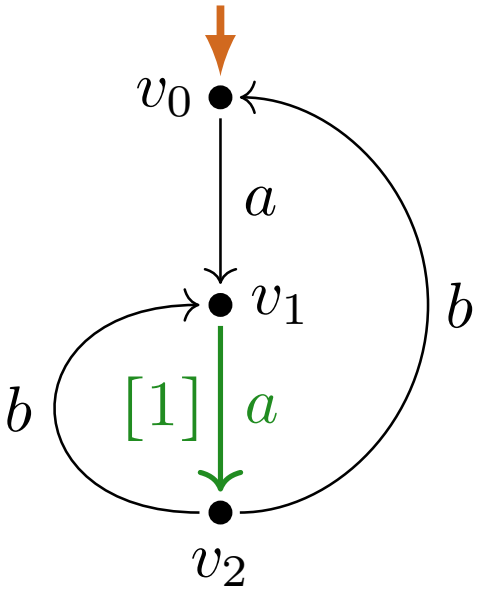


layered
LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot s(v_1)
\end{aligned}
$$

$$
\begin{aligned}
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

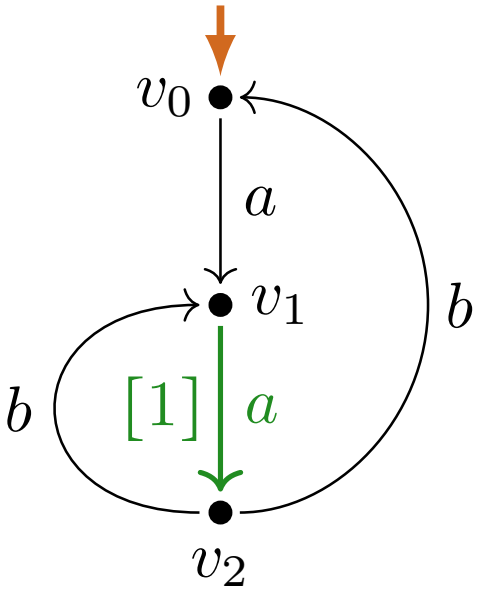
Readback from layered LEE-witness (example)



layered LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

Readback from layered LEE-witness (example)

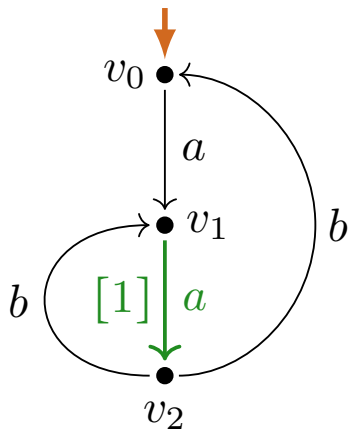


layered LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \in \mathit{RExp}^{(*/1)} \\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

Readback

Readback steps reverse P-defined transitions up to $=_{\text{Mil}^-}$ and $\leftrightarrows$.



layered
LEE-witness

$$
\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \in \mathit{RExp}^{(*/\perp)} \\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}
$$

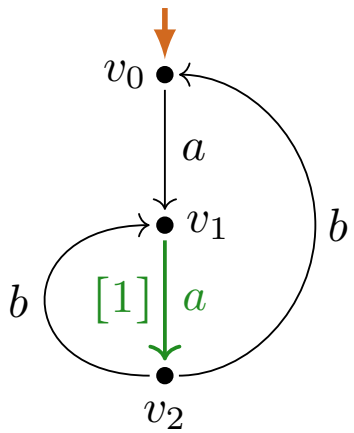
Readback

Lemma (~ *G–Fokkink, 2020*)

Process graphs with LLEE *are* $[\![\cdot]\!]_P$*-expressible by* $(*/\perp)$ *reg. expressions:*

$$\text{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\perp)} \big(G \leftrightarrows P(uf) \big).$$

Proof: Readback steps reverse P-defined transitions up to $=_{\text{Mil}^-}$ and $\leftrightarrows$.



layered
LEE-witness

$$\begin{aligned}
s(v_0) &= 0^* \cdot a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot s(v_1) \\
&=_{\text{Mil}^-} a \cdot \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \in \mathit{RExp}^{(*/\perp)} \\
s(v_1) &= \big(a \cdot s(v_2, v_1)\big)^* \cdot 0 \\
&=_{\text{Mil}^-} \big(a \cdot (b + b \cdot a)\big)^* \cdot 0 \\
s(v_2, v_1) &= 0^* \cdot \big(b \cdot s(v_1, v_1) + b \cdot s(v_0, v_1)\big) \\
&=_{\text{Mil}^-} 0^* \cdot \big(b \cdot 1 + b \cdot a\big) \\
&=_{\text{Mil}^-} b + b \cdot a \\
s(v_1, v_1) &= 1 \\
s(v_0, v_1) &= 0^* \cdot a \cdot s(v_1, v_1) \\
&= 0^* \cdot a \cdot 1 \\
&=_{\text{Mil}^-} a
\end{aligned}$$

Readback (refined)

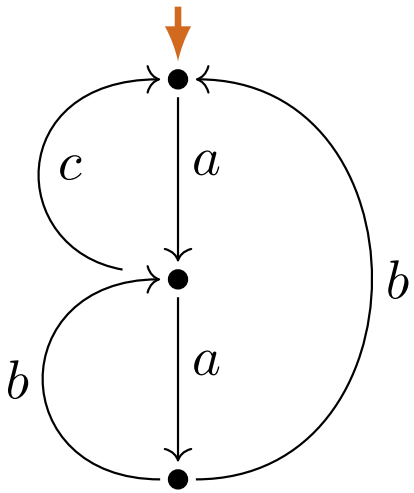
Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!

G_2



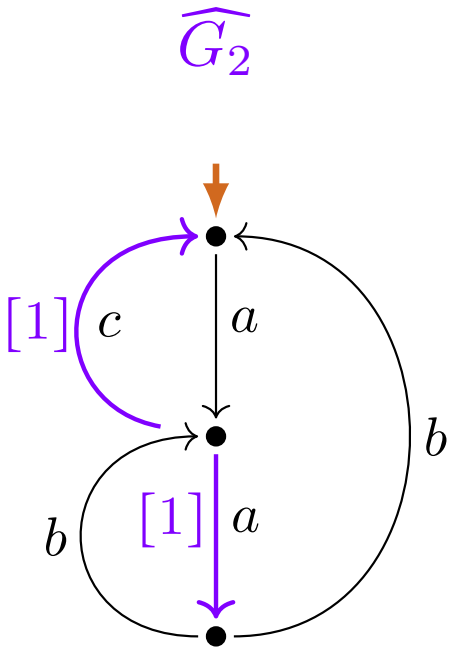
Readback (refined)

Statement

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

$$\text{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\not{1})} \left(G \sim P(uf) \right).$$

Proof: Refine readback steps to precisely reverse P-defined transitions!



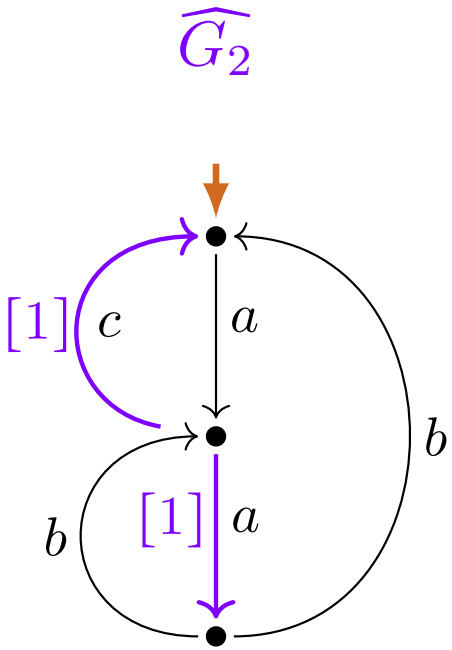
Readback (refined)

Statement

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

$$\text{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\not{1})} \left(G \sim P(uf) \right).$$

Proof: Refine readback steps to precisely reverse P-defined transitions!



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

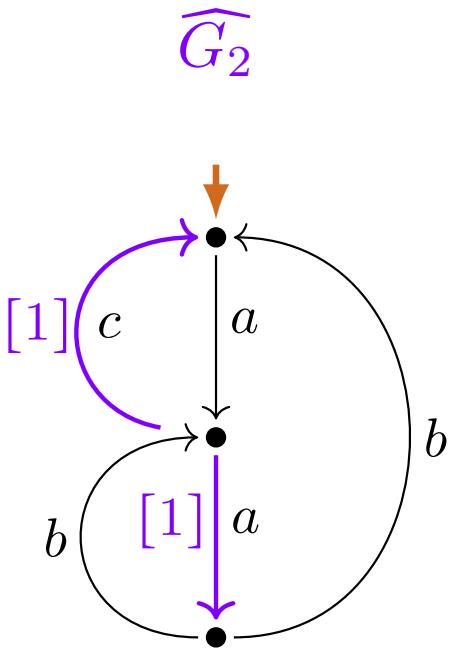
Readback (refined)

Statement

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

$$\text{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\not{1})} \left(G \sim P(uf) \right).$$

Proof: Refine readback steps to precisely reverse P-defined transitions!



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

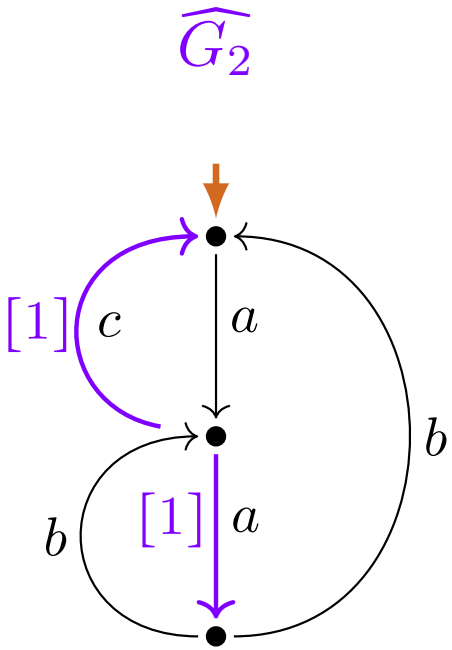
Readback (refined)

Statement

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

$$\text{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\not{1})} \left(G \sim P(uf) \right).$$

Proof: Refine readback steps to precisely reverse P-defined transitions!



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

$$\downarrow a$$

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

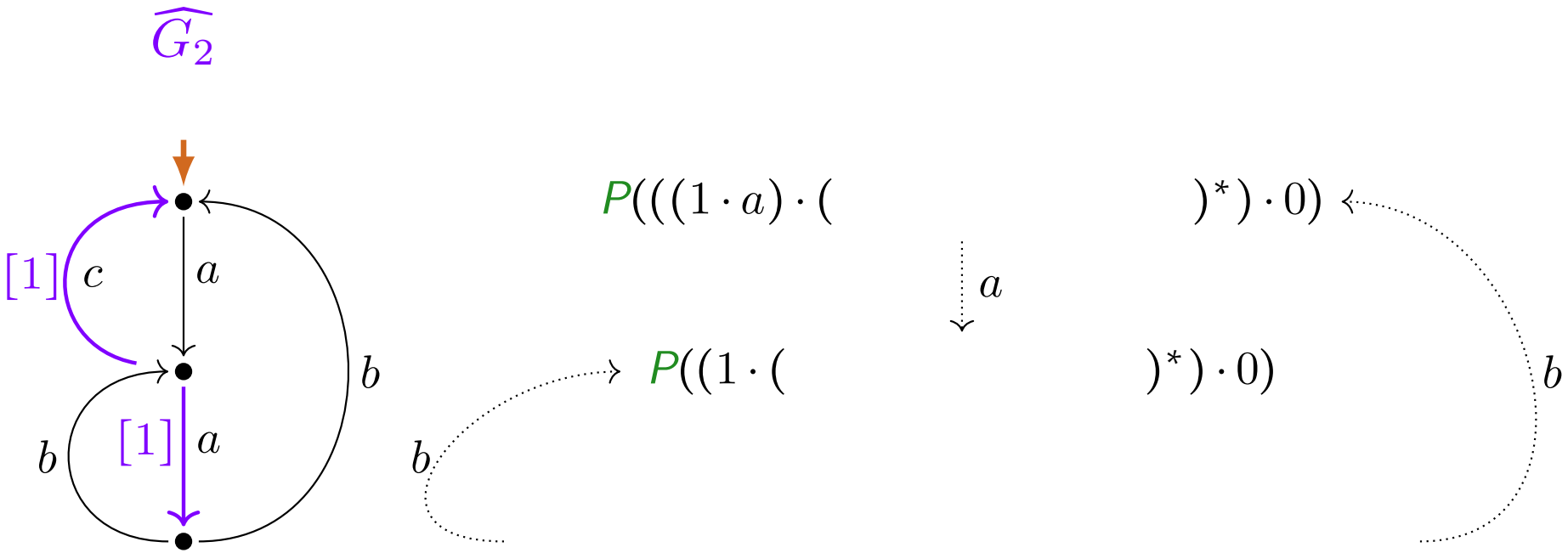
Readback (refined)

Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!



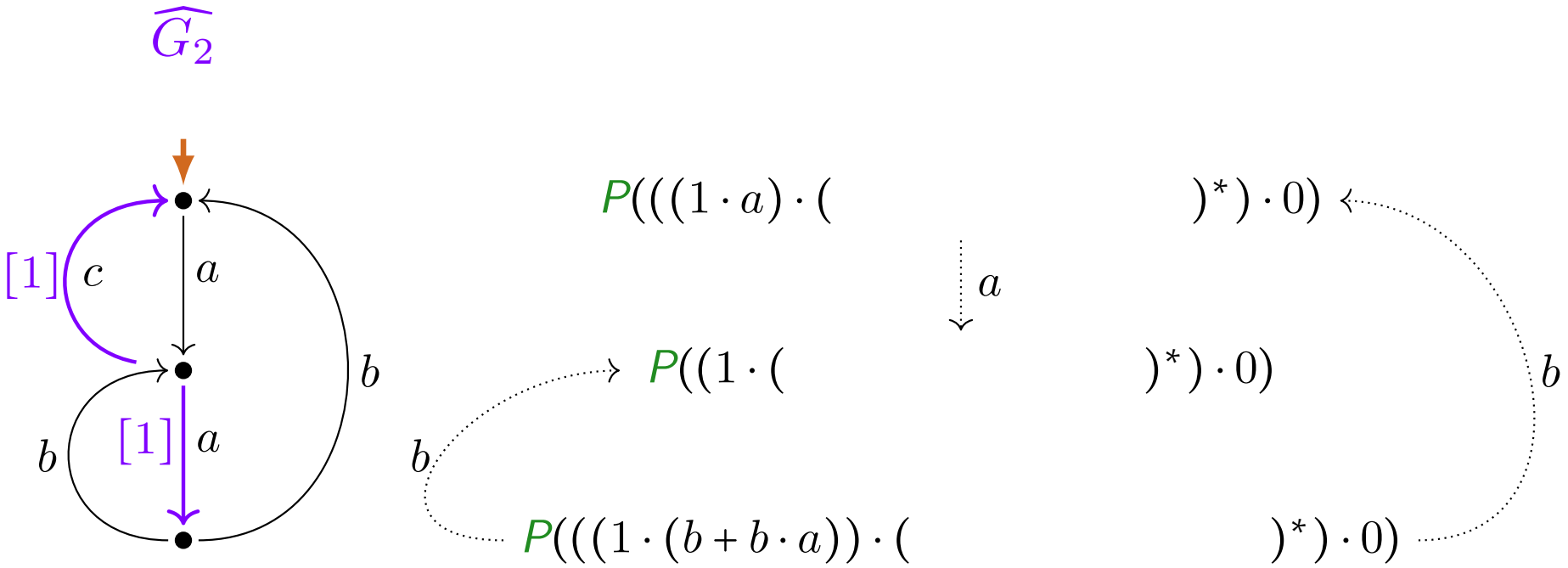
Readback (refined)

Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!



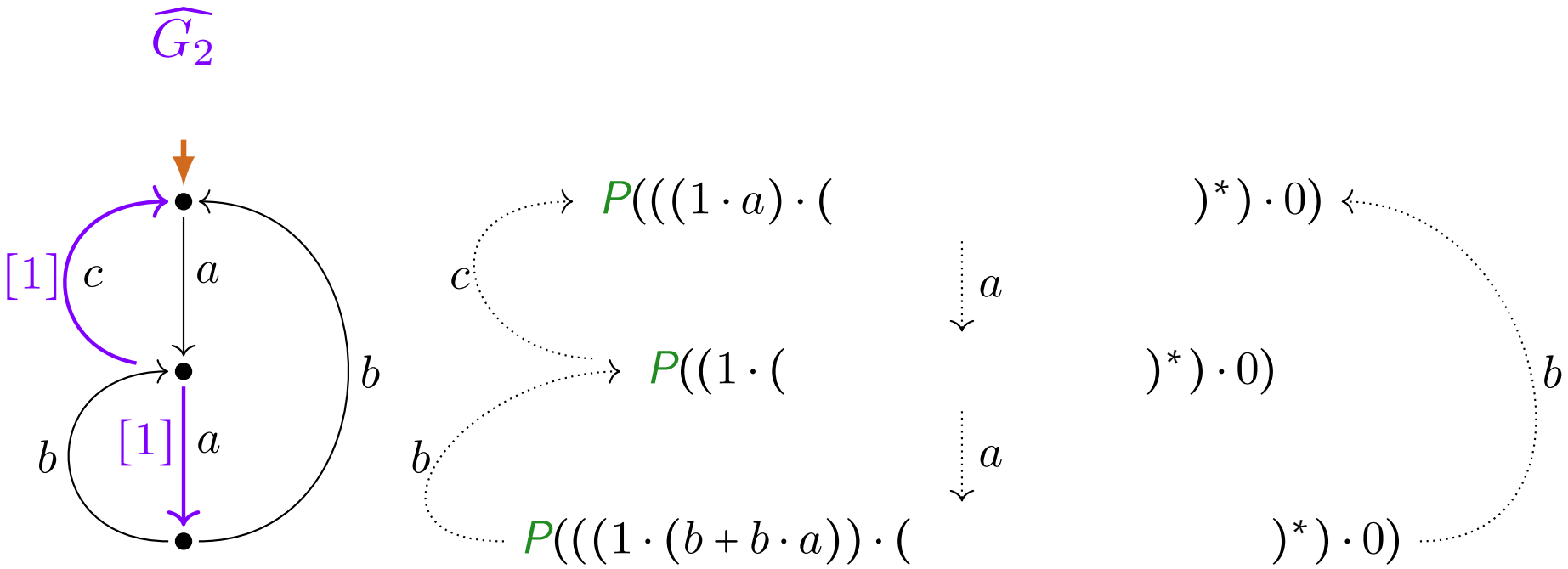
Readback (refined)

Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!



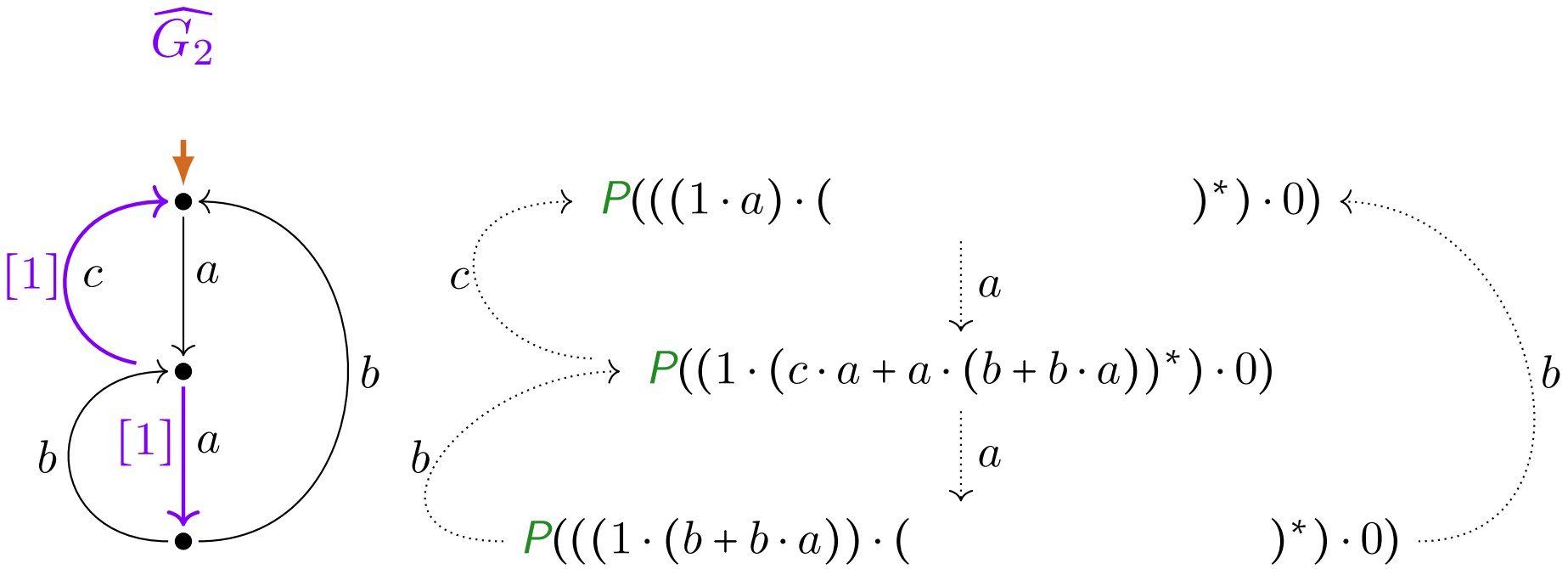
Readback (refined)

Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!



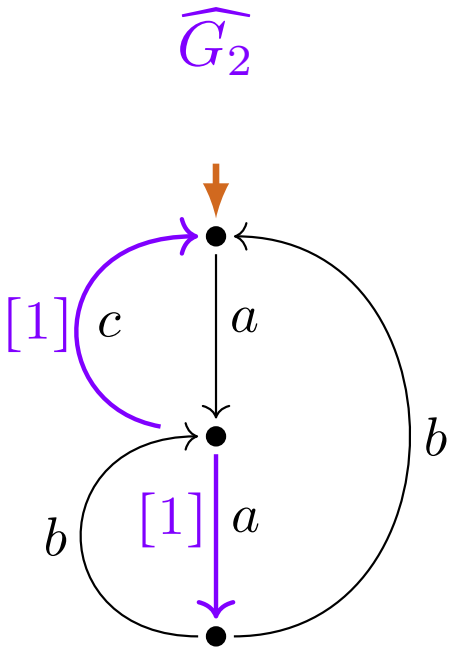
Readback (refined)

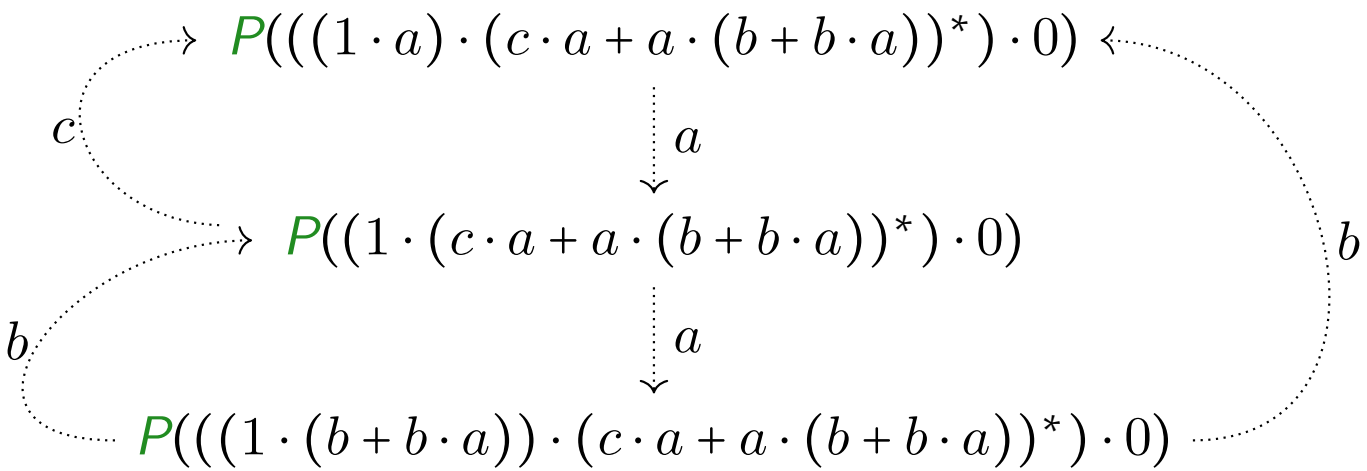
Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!





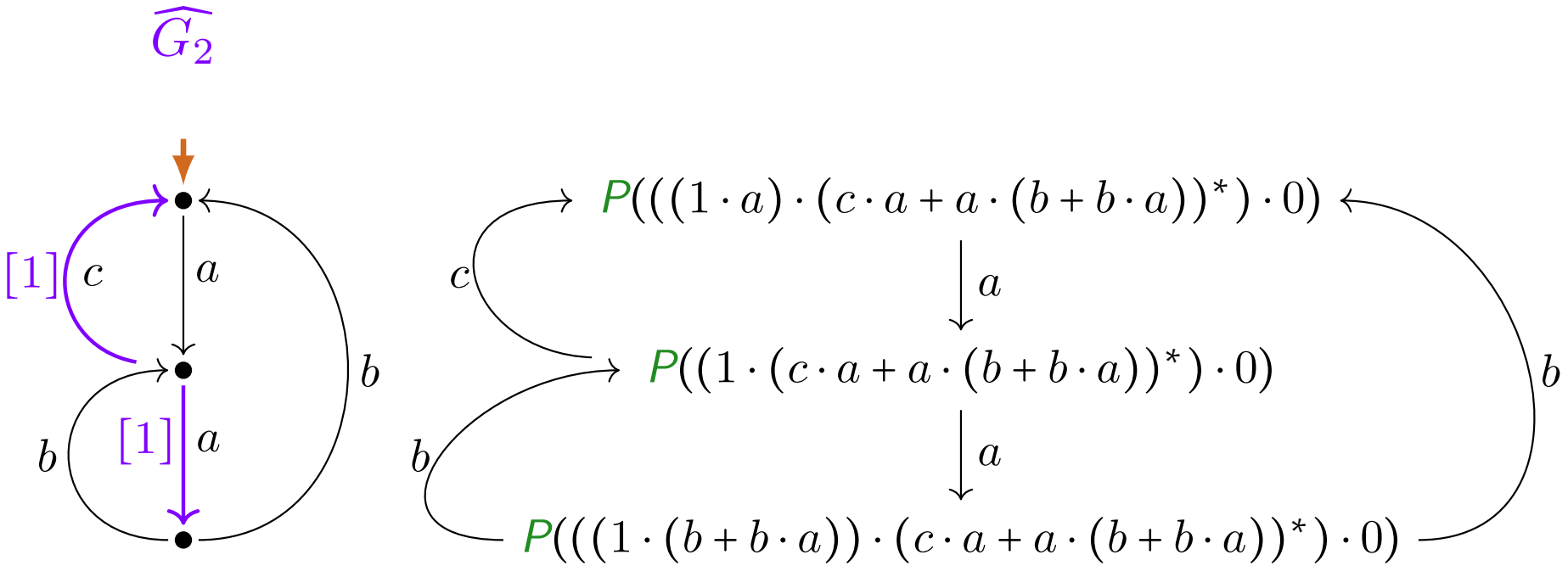
Readback (refined)

Statement

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

$$\text{LLEE}(G) \implies \exists\, uf \in \mathit{RExp}^{(*/\not{1})} \left(G \sim P(uf) \right).$$

Proof: Refine readback steps to precisely reverse P-defined transitions!



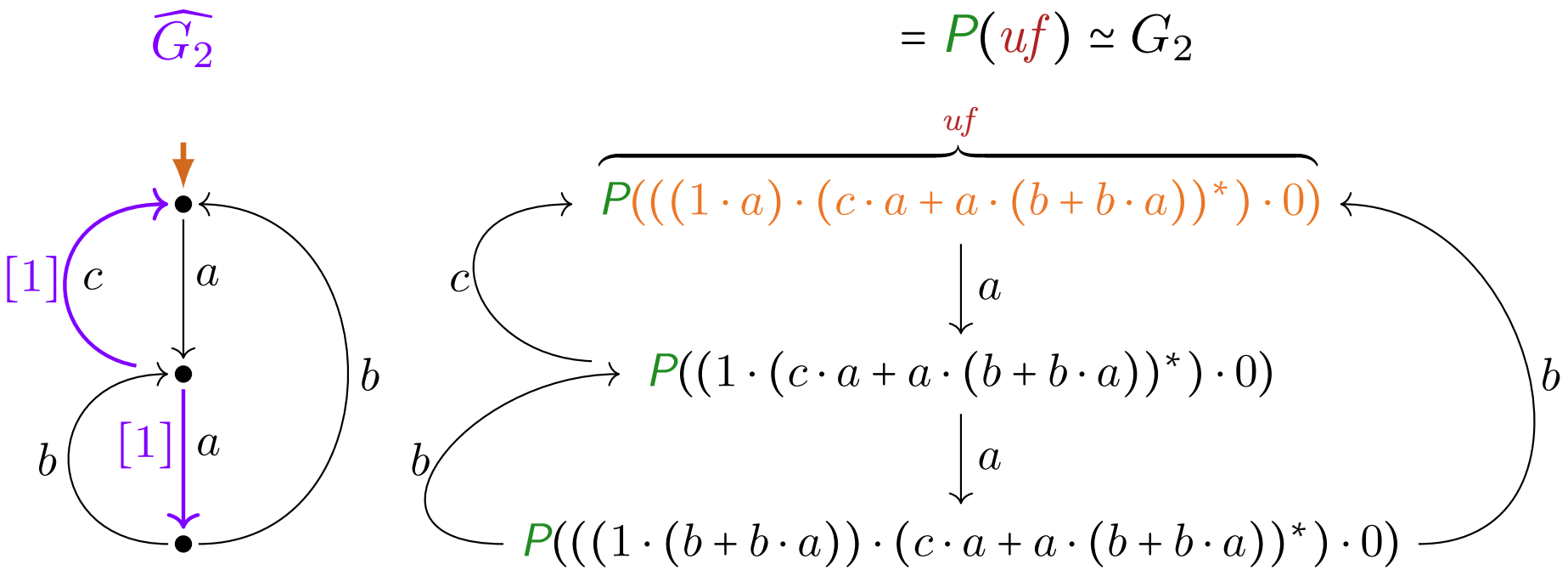
Readback (refined)

Statement

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

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

Proof: Refine readback steps to precisely reverse P-defined transitions!



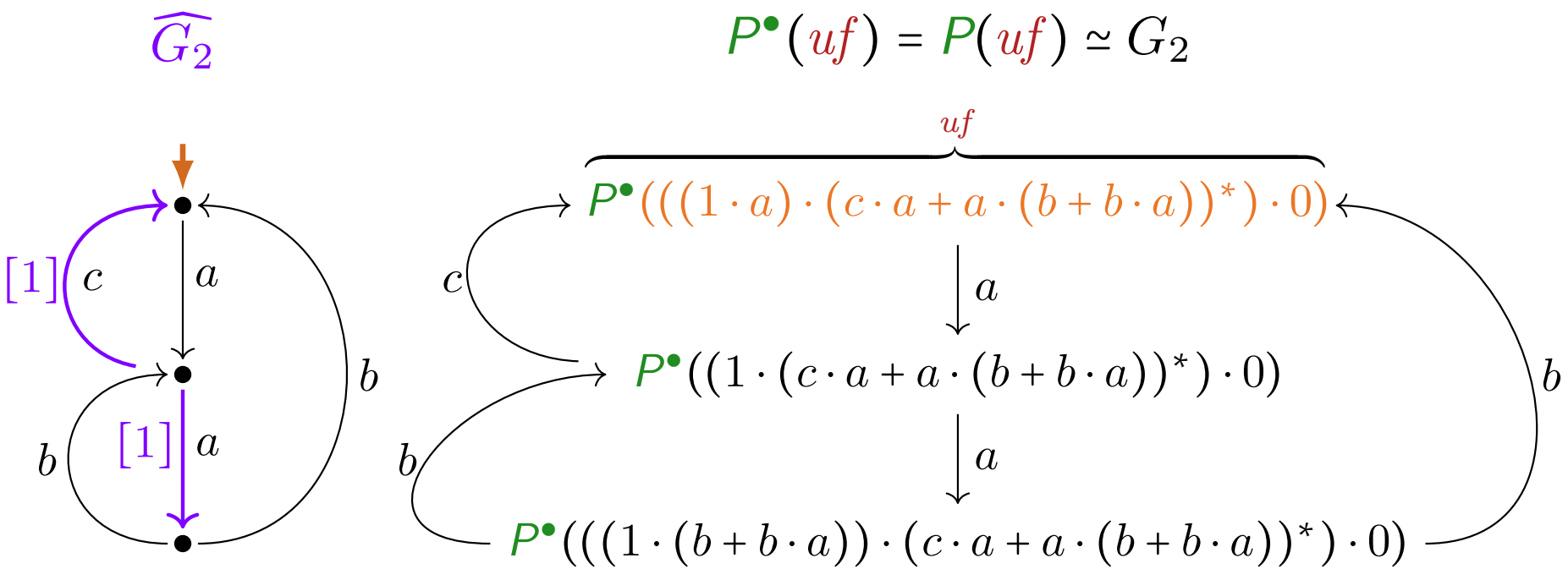
Readback (refined)

Lemma (*TERMGRAPH '24/'26*)

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

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

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



Characterising $P^\bullet$-expressibility

Theorem (LLEE $\triangleq$ image of $P^\bullet|_{RExp^{(*/\perp)}}$)

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

Characterising $P^\bullet$-expressibility

Theorem (LLEE $\triangleq$ image of $P^\bullet|_{RExp^{(*/\perp)}}$)

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

Corollary (P-expressibility by $(*/\perp)$ expr's)

For every process graph G, TFAE:

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

Characterising $P^\bullet$-expressibility

Theorem (LLEE $\triangleq$ image of $P^\bullet|_{RExp^{(*/\perp)}}$)

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

Corollary (P-expressibility by $(*/\perp)$ expr's)

For every process graph G, TFAE:

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

Theorem (1-LLEE $\triangleq$ image of $P^\bullet$)

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

Summary

- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Loop existence and elimination (LEE), layered form (LLEE)
- Small steps for hard questions
  - reduction steps for well-behaved specifications under bisimilarity
    - expressibility is decidable
  - collapse steps for LLEE graphs under bisimilarity
    - completeness for $(*/\not{1})$ expressions
  - collapse steps, as far as possible, for 1-LLEE graphs under bisimilarity
    - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences
  - LEE/1-LEE characterise restricted to $(*/\not{1})$/unrestricted image of $P^\bullet$
    - where $P^\bullet$ a sharing-increased compact refinement of P

Summary

- Milner's process interpretation P/semantics $[\![\cdot]\!]_P$
  - P-/$[\![\cdot]\!]_P$-expressible graphs ($\rightsquigarrow$ expressibility question)
  - axioms for $[\![\cdot]\!]_P$-identity ($\rightsquigarrow$ completeness question)
- Loop existence and elimination (LEE), layered form (LLEE)
- Small steps for hard questions
  - reduction steps for well-behaved specifications under bisimilarity
    - expressibility is decidable
  - collapse steps for LLEE graphs under bisimilarity
    - completeness for $(*/\not{1})$ expressions
  - collapse steps, as far as possible, for 1-LLEE graphs under bisimilarity
    - counterexample to collapse, but leads to completeness
- Interpretation–readback correspondences
  - LEE/1-LEE characterise restricted to $(*/\not{1})$/unrestricted image of $P^\bullet$
    - where $P^\bullet$ a sharing-increased compact refinement of P

Next aims, and refined questions

Expressibility problems (for $P^\bullet$ and $[\![\cdot]\!]_P$)

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

- LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)((*/\bot))$ 1-LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)$ (TERMGRAPH'26)
- LLEE $\hat{=}$ co-reducible$^+$ graphs ($\Leftarrow$ Laarakker/Kappè, TERMGRAPH'26)

Next aims, and refined questions

Expressibility problems (for $P^\bullet$ and $[\![\cdot]\!]_P$)

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

- LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)((*/\!\!\perp))$ 1-LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)$ (TERMGRAPH'26)
- LLEE $\hat{=}$ co-reducible$^+$ graphs ($\Leftarrow$ Laarakker/Kappè, TERMGRAPH'26)

A2: LEE/$P^\bullet$-Expressibility is decidable in $\leq$ cubic time.

$\Leftarrow$ confluence of loop elimination with pruning steps (IWC'26), which also entails LLEE = LEE.

Next aims, and refined questions

Expressibility problems (for $P^\bullet$ and $[\![\cdot]\!]_P$)

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

- LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)((*/\!\!\perp))$ 1-LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)$ (TERMGRAPH'26)
- LLEE $\hat{=}$ co-reducible$^+$ graphs ($\Leftarrow$ Laarakker/Kappè, TERMGRAPH'26)

A2: LEE/$P^\bullet$-Expressibility is decidable in $\leq$ cubic time.

$\Leftarrow$ confluence of loop elimination with pruning steps (IWC'26), which also entails LLEE $=$ LEE.

Q1: Is 1-LLEE/$P^\bullet$-expressibility decidable in polynomial time**?**

Q2: Is $[\![\cdot]\!]_P$-expressibility by a regular expression, for a finite process graph, decidable in polynomial time/fixed-parameter tractable time**?**

Next aims, and refined questions

Expressibility problems (for $P^\bullet$ and $[\![\cdot]\!]_P$)

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

- LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)((*/\!\!\perp))$ 1-LLEE $\hat{=}$ $\mathrm{im}(P^\bullet)$ (TERMGRAPH'26)
- LLEE $\hat{=}$ co-reducible$^+$ graphs ($\Leftarrow$ Laarakker/Kappè, TERMGRAPH'26)

A2: LEE/$P^\bullet$-Expressibility is decidable in $\leq$ cubic time.

$\Leftarrow$ confluence of loop elimination with pruning steps (IWC'26), which also entails LLEE = LEE.

Q1: Is 1-LLEE/$P^\bullet$-expressibility decidable in polynomial time?

Q2: Is $[\![\cdot]\!]_P$-expressibility by a regular expression, for a finite process graph, decidable in polynomial time/fixed-parameter tractable time?

Completeness problem, solution:

A3: motivation of crystallization

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

TERMGRAPH / IWC Talks

TERMGRAPH (tomorrow, 19 July)

CG: *Towards a Characterisation of the Graph Structure of Process Interpretations of Regular Expressions*

Ext. Abstract `https://clegra.github.io/lf/pi-struc-TG-26.pdf`

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

TERMGRAPH / IWC Talks

TERMGRAPH (tomorrow, 19 July)

CG: *Towards a Characterisation of the Graph Structure of Process Interpretations of Regular Expressions*

Ext. Abstract https://clegra.github.io/lf/pi-struc-TG-26.pdf

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

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

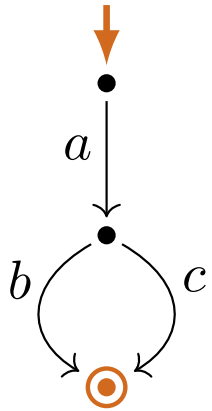
- critical-pair like analysis of loop elimination steps $\longrightarrow_{\mathsf{elim}}$
- $\longrightarrow_{\mathsf{elim}}$ forks are joinable by $\longrightarrow_{\mathsf{elim}}$ steps and prune steps $\longrightarrow_{\mathsf{prune}}$

Resources

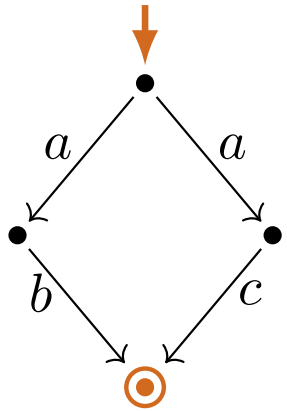
- Slides on clegra.github.io
  - slides: .../lf/IFIP-1_6-2026.pdf
- CG: Closing the Image of the Process Interpretation of 1-Free Regular Expressions Under Bisimulation Collapse
  - TERMGRAPH 2024, extended abstract
- CG: The Image of the Process Interpretation of Regular Expressions is Not Closed under 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

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

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



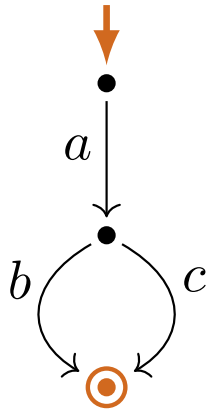
$P(a \cdot (b + c))$



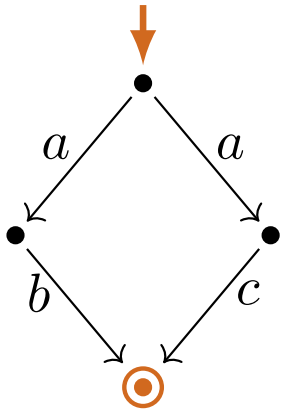
$P(a \cdot b + a \cdot c)$

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

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



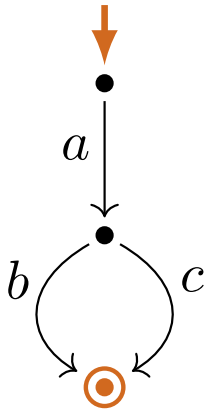
$\not\leftrightarrow$



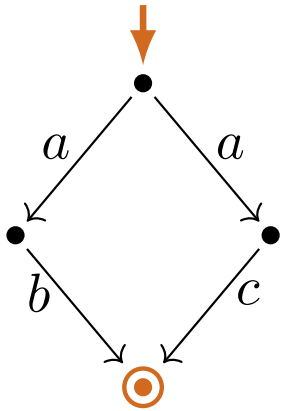
$P(a \cdot (b + c))$ $\not\leftrightarrow$ $P(a \cdot b + a \cdot c)$

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

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



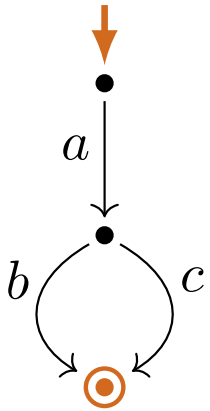
$\neq_{[\![\cdot]\!]_P}$



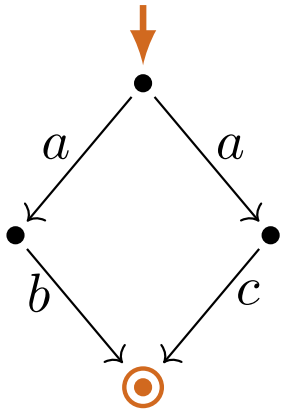
$a \cdot (b + c)$ $\neq_{[\![\cdot]\!]_P}$ $a \cdot b + a \cdot c$

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

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



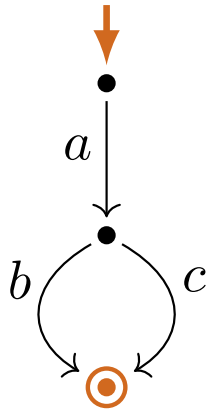
$=_L$



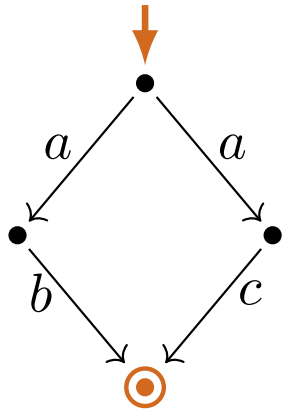
$$a \cdot (b + c) \quad =_L \quad a \cdot b + a \cdot c$$

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

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



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



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

Milner's proof system Mil

Axioms:

(A1) $e + (f + g) = (e + f) + g$

(A2) $e + 0 = e$

(A3) $e + f = f + e$

(A4) $e + e = e$

(A5) $e \cdot (f \cdot g) = (e \cdot f) \cdot g$

(A6) $(e + f) \cdot g = e \cdot g + f \cdot g$

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

(A7) $e = 1 \cdot e$

(A8) $e = e \cdot 1$

(A9) $0 = 0 \cdot e$

(A10) $e^* = 1 + e \cdot e^*$

(A11) $e^* = (1 + e)^*$

But: $e \cdot 0 \neq 0$

Inference rules: rules of equational logic *plus*

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

Milner's Completeness Question

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

Milner's questions

(Q1) Completeness Question:

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

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

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

Milner's questions

(Q1) Completeness Question:

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

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

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

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

Milner's questions

(Q1) Completeness Question:

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

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

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

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

Milner's questions

(Q1) Completeness Question:

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

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

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

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

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

Milner's questions

(Q1) Completeness Question:

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

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

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

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

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

Question (Q2) specialized

$(Q2)_0$ *P*-Expressibility and *P*-$(*/\dot{-})$-Expressibility:

What structural property characterizes:

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

Milner's Proof System for Regular Expressions Modulo Bisimilarity is Complete

Crystallization: Near-Collapsing Process Graph Interpretations of Regular Expressions

Clemens Grabmayer (Department of Computer Science, Gran Sasso Science Institute, Viale F. Crispi, 7, 67100 L'Aquila AQ, Italy)

Abstract

We report on a lengthy completeness proof for Robin Milner's proof system Mil (1984) for bisimilarity of regular expressions in the process semantics. Central for our proof are the recognitions:

1. Process graphs with 1-transitions (1-charts) and the loop existence/elimination property LLEE are **not** closed under bisimilation collapse,
2. Such process graphs can be '**crystallized**' to 'near-collapsed' 1-charts with some strongly connected components of 'twin-crystal' form.

The Process Semantics of Regular Expressions

Milner (1984) introduced a process semantics for regular expressions: the interpretation of 0 is *deadlock*, of 1 is an *empty step to termination*, letters a are *atomic actions*, the operators $+$ and $\cdot$ stand for *choice* and *concatenation* of processes, and unary Kleene star $(\cdot)^*$ represents *(unbounded) iteration*. Formally, Milner defined chart (finite process graph) interpretations $\mathcal{C}(e)$ of regular expressions e.

Milner's Proof System

As axiomatization of the relation $e_1 =_{\boldsymbol{P}} e_2$ on regular expressions e_1 and e_2 defined by $\mathcal{C}(e_1) \leftrightarroweq \mathcal{C}(e_2)$ (as bisimilarity $\leftrightarroweq$ of chart interpretations), Milner asked whether the following system Mil is complete:

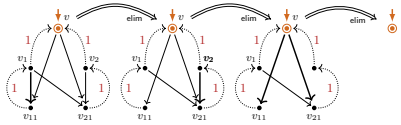
(A1)	$e+(f+g)=(e+f)+g$	(A7)	$e=1\cdot e$
(A2)	$e+0=e$	(A8)	$e=e\cdot 1$
(A3)	$e+f=f+e$	(A9)	$0=0\cdot e$
(A4)	$e+e=e$	(A10)	$e^*=1+e\cdot e^*$
(A5)	$e\cdot(f\cdot g)=(e\cdot f)\cdot g$	(A11)	$e^*=(1+e)^*$
(A6)	$(e+f)\cdot g=e\cdot g+f\cdot g$		

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

This system is a variation of Salomaa's complete axiom system (1966) for language equality of regular expressions, missing left-distributivity $e\cdot(f+g)=e\cdot f+e\cdot g$ and $e\cdot 0=0$, which are unsound here.

Loop Existence and Elimination

The process semantics is incomplete: not every finite process graph is *expressible by* (=bisimilar to the interpretation of) a regular expression. A sufficient condition for expressibility is the *(layered) loop existence and elimination property* LLEE. It is defined via elimination of 'loops' (loop subcharts):



LLEE holds if a graph without infinite behavior can be obtained. Important features of LLEE:

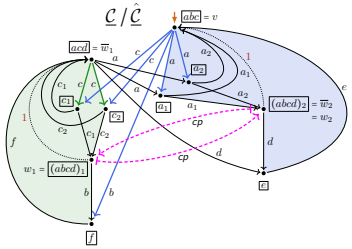
(US) Every guarded LLEE-1-chart (chart, maybe 1-transitions, with LLEE) is uniquely Mil-provably solvable modulo provability in Mil (CALCO 2021).

(IV) The chart interpretation $\mathcal{C}(e)$ of a regular expression e can always be expanded under bisimilarity to a LLEE-1-chart $\mathcal{C}(e)$ (TERMGRAPH 2020).

($C_\pm$) LLEE-charts (without 1-transitions) are preserved by bisimulation collapse (G/Fokkink, LICS'20).

LLEE-preserving Collapse Fails

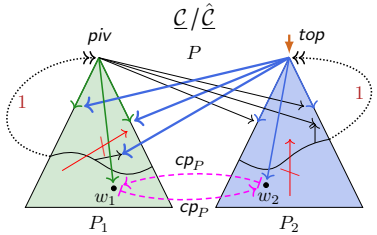
LLEE-1-charts with 1-transitions, however, are **not** preserved under bisimulation collapse. A counterexample is provided by the following LLEE-1-chart $\mathcal{C}$:



Identifying the bisimilar vertices w_1 and w_2 yields a chart for which LLEE fails. Also, the subcharts of $\mathcal{C}$ that are rooted at w_1 and w_2 are **not** *LLEE-preservingly jointly minimizable under bisimilarity.*

Twin-Crystals

The counterexample to LLEE-preserving collapse is symmetric, and its structure can be abstracted as:



It is a LLEE-1-chart with a single scc (strongly connected component) P that consists of a *pivot part* P_1 below *pivot vertex* ***piv***, and a *top part* P_2 below *top vertex* ***top***. P_1 and P_2 are connected only via transitions from ***piv*** and from ***top***. While both P_1 and P_2 are collapsed, P contains *bisimilarity redundancies* (= distinct bisimilar vertices) such as $\{w_1, w_2\}$ that are linked by a self-inverse counterpart function cp_P. We call such an scc a *twin-crystal*. We have:

(CC) Every Mil-provable solution of a twin-crystal gives rise to a Mil-provable solution of its bisimulation collapse (which often is not a LLEE-1-chart).

Crystallization of LLEE-1-charts

By *crystallization* of a LLEE-1-chart $\mathcal{C}$ we mean:

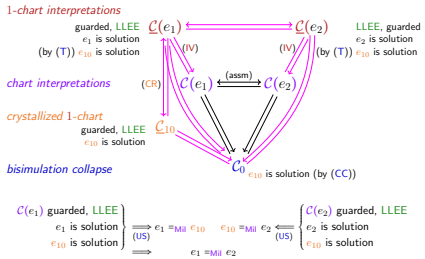
- a process of minimization of $\mathcal{C}$ under bisimilarity by steps that **eliminate most** (all but crystalline) bisimilarity redundancies $\{w_1, w_2\}$, roughly by redirecting transitions that target w_1 over to w_2;
- hereby only '**reduced**' bisimilarity redundancies can be eliminated LLEE-preservingly, which exist whenever a LLEE-1-chart is **not** collapsed;
- the result is a *crystallized* LLEE-1-chart that is bisimilar to $\mathcal{C}$, and collapsed **apart from** within some its scc's that are twin-crystals.

The crystallization process facilitates to show:

(CR) From every LLEE-1-chart a bisimilar crystallized LLEE-1-chart can be obtained.

Completeness Proof

Let $\mathcal{C}(e_1) \leftrightarroweq \mathcal{C}(e_2)$ be bisimilar chart interpretations of regular expressions e_1 and e_2. To secure LLEE, $\mathcal{C}(e_1)$ and $\mathcal{C}(e_2)$ are expanded to their 1-chart interpretations $\mathcal{C}(e_1)$ and $\mathcal{C}(e_2)$. One of them, say $\mathcal{C}(e_1)$, is crystallized to $\mathcal{C}_{10}$. All (1-)charts are linked by (1-)bisimulations to their bisimulation collapse $\mathcal{C}_0$.



From $\mathcal{C}_{10}$ a provable solution e_{10} can be extracted due to LLEE, transferred (T) to the collapse $\mathcal{C}_0$, and then to $\mathcal{C}(e_1)$ and $\mathcal{C}(e_2)$. On the LLEE-1-charts $\mathcal{C}(e_1)$ and $\mathcal{C}(e_2)$, e_{10} can be proved equal to the solutions e_1 and e_2 there, respectively. By transitivity, $e_1 =_{\text{Mil}} e_2$ (provability of $e_1 = e_2$ in Mil) follows.

Theorem. *Milner's system* Mil *is complete:* $e_1 =_{\boldsymbol{P}} e_2$ *implies* $e_1 =_{\text{Mil}} e_2$*, for reg. expr's* e_1, e_2.

Next Steps and Projects

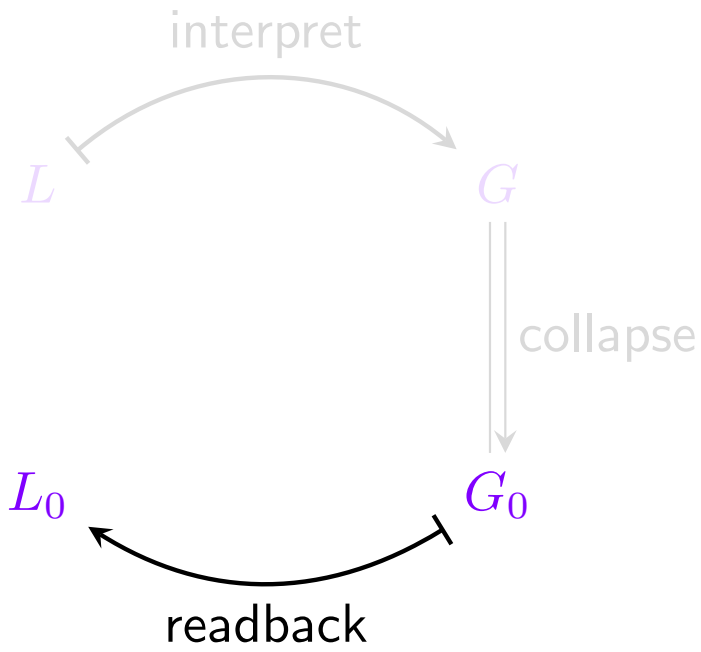
- Monograph project: proof in fine-grained details.
- Build an animation tool for crystallization.
- Apply crystallization to find an efficient algorithm for expressibility of finite process graphs by a regular expression modulo bisimilarity.

Contact

clemens.grabmayer@gssi.it

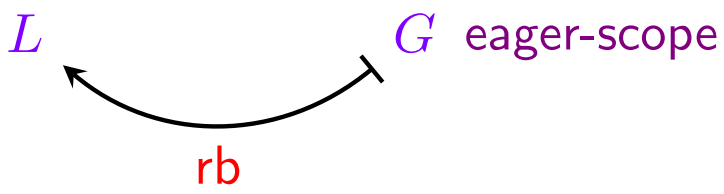
GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Readback



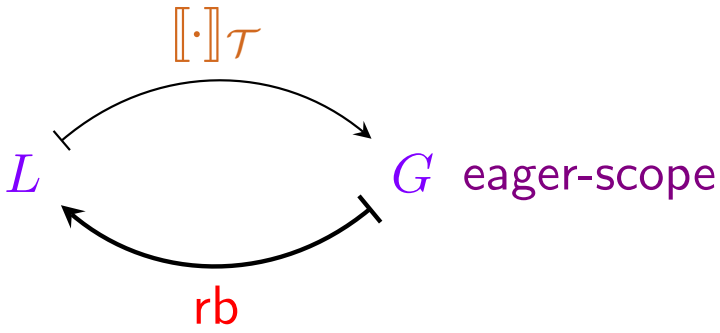
Readback

defined with property:



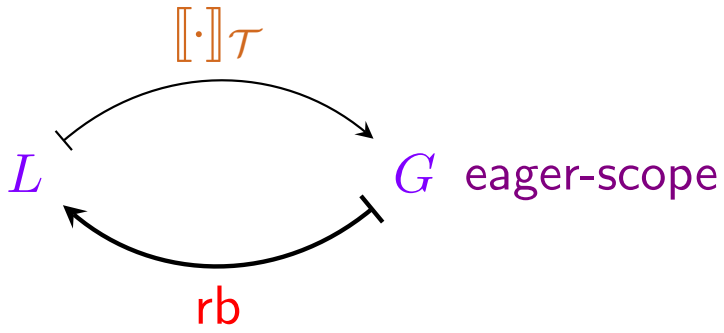
Readback

defined with property:



Readback

defined with property:



Theorem

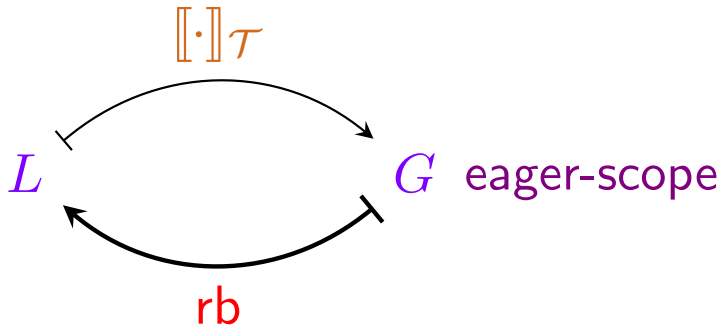
For all eager-scope λ-term-graphs G:

$$([\![\cdot]\!]_\tau \circ \mathsf{rb})(G) \simeq G$$

The readback rb is a right-inverse of $[\![\cdot]\!]_\tau$ modulo isomorphism $\simeq$.

Readback

defined with property:



Theorem

For all eager-scope λ-term-graphs G:

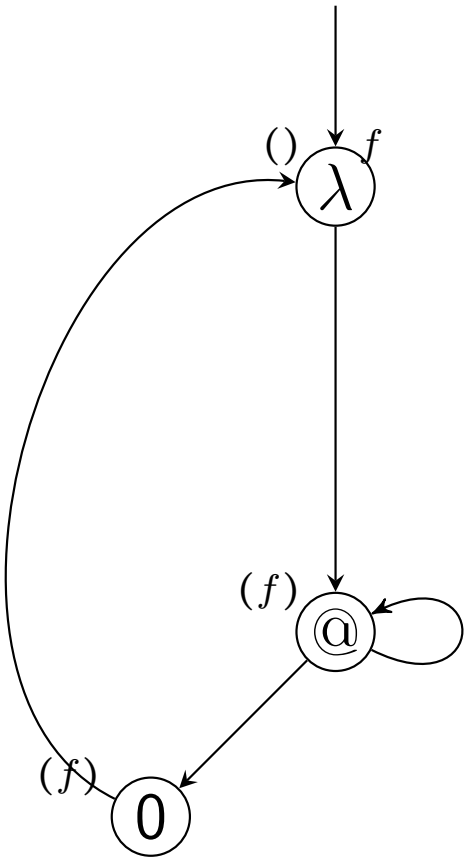
$$([\![\cdot]\!]_\mathcal{T} \circ \mathsf{rb})(G) \simeq G$$

The readback rb is a right-inverse of $[\![\cdot]\!]_\mathcal{T}$ modulo isomorphism $\simeq$.

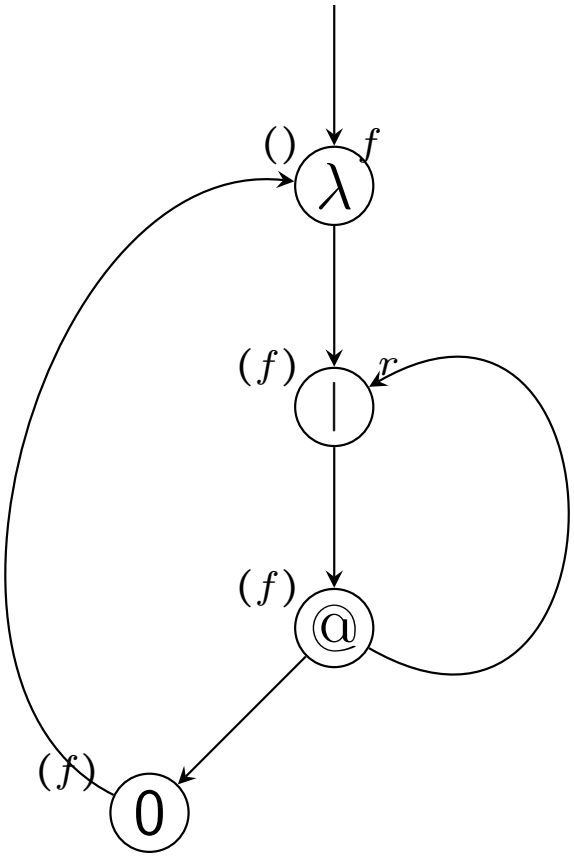
idea:

1. construct a spanning tree T of G
2. using local rules, in a bottom-up traversal of T synthesize $L = \mathsf{rb}(G)$

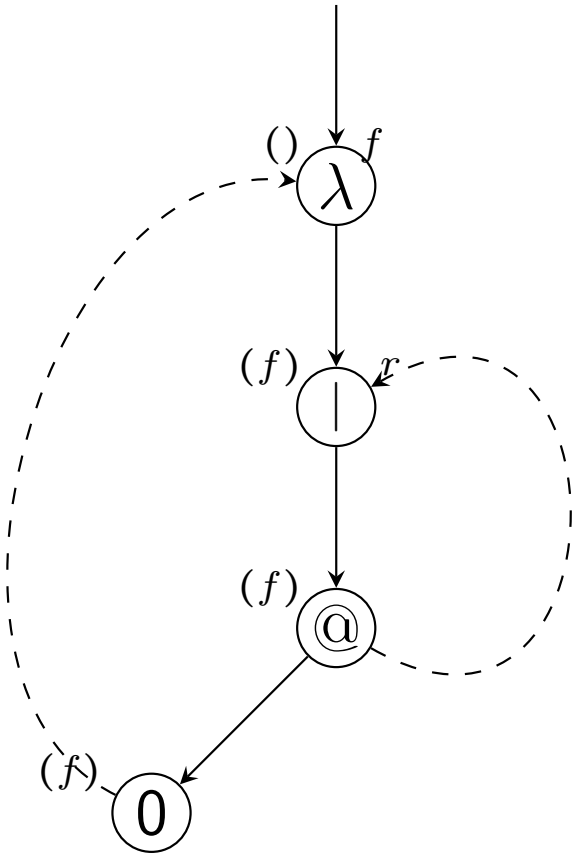
Readback: example (fix)



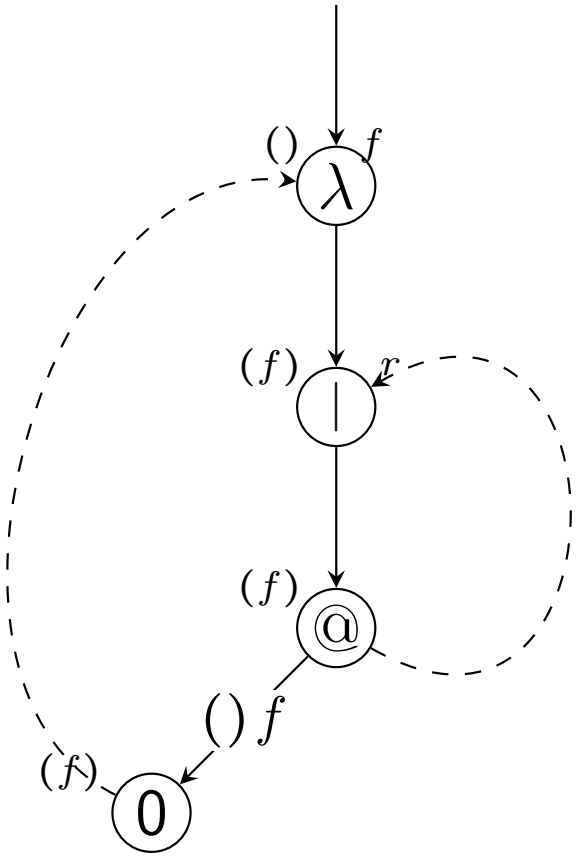
Readback: example (fix)



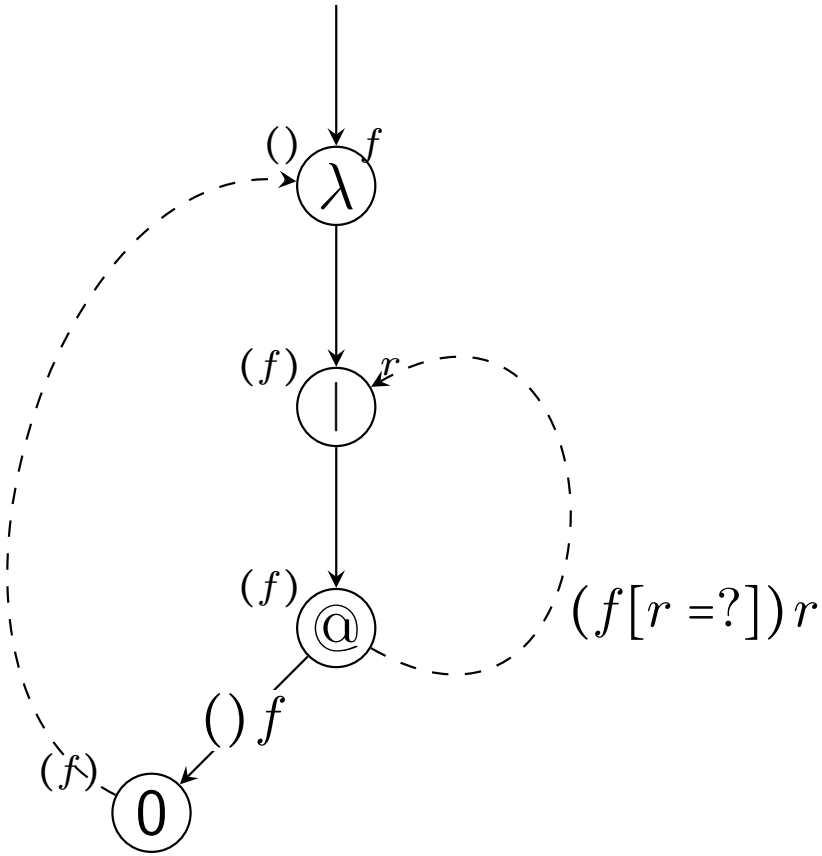
Readback: example (fix)



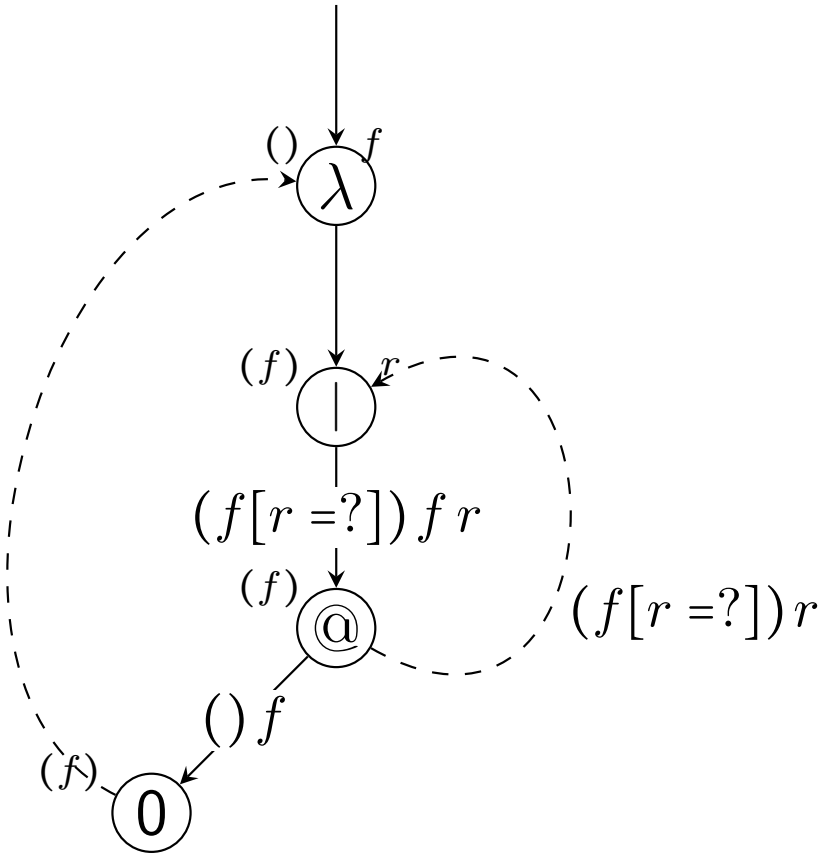
Readback: example (fix)



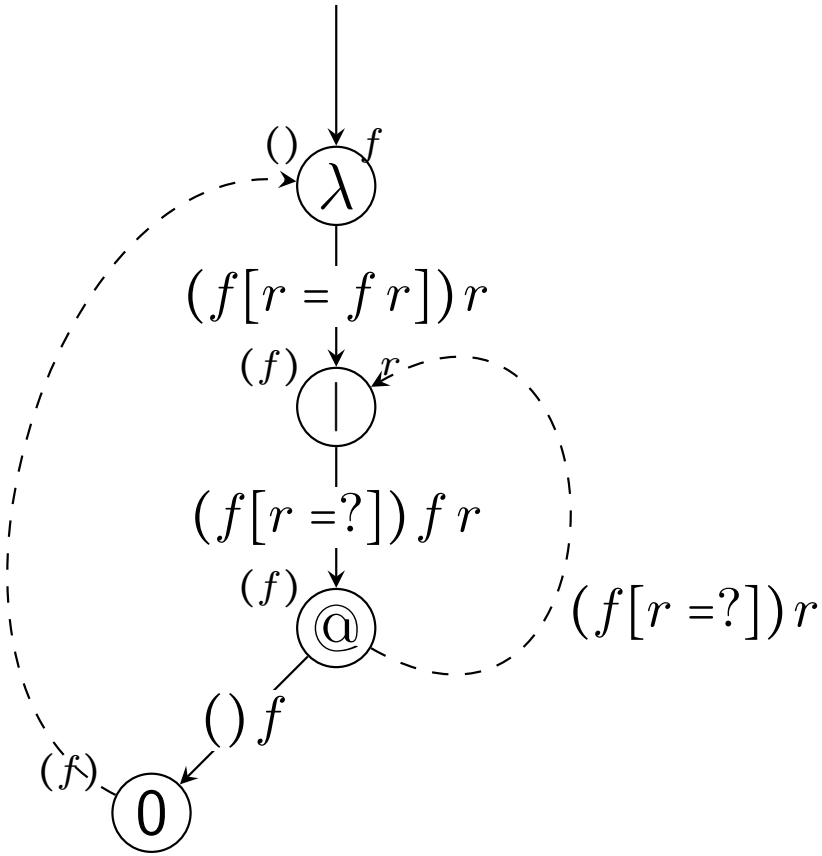
Readback: example (fix)



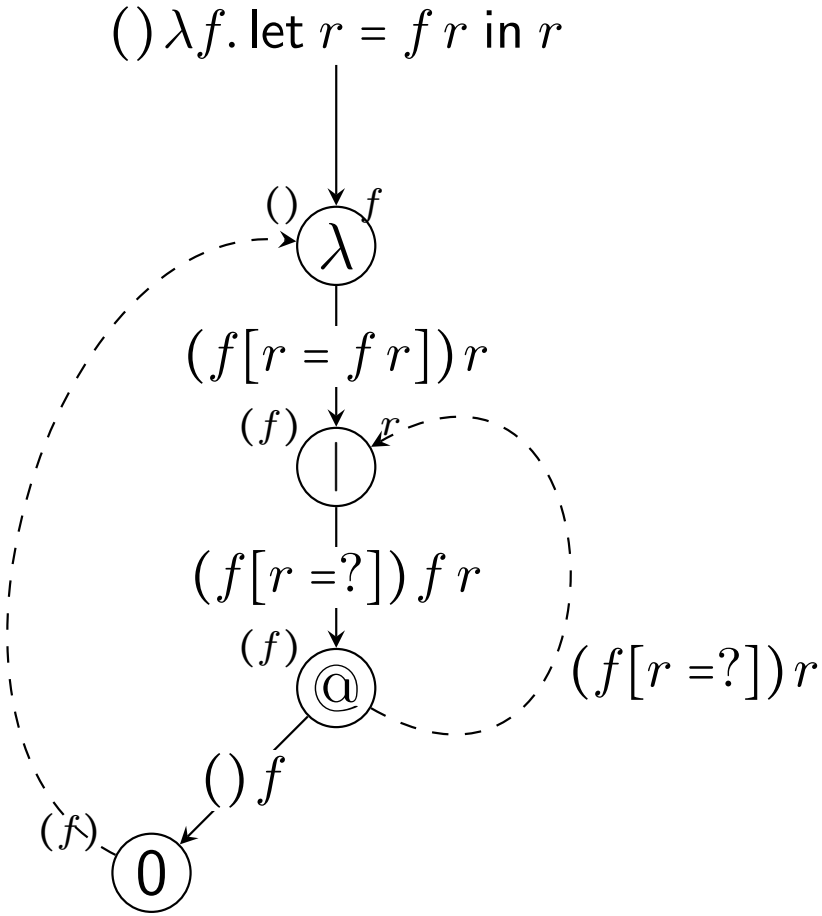
Readback: example (fix)



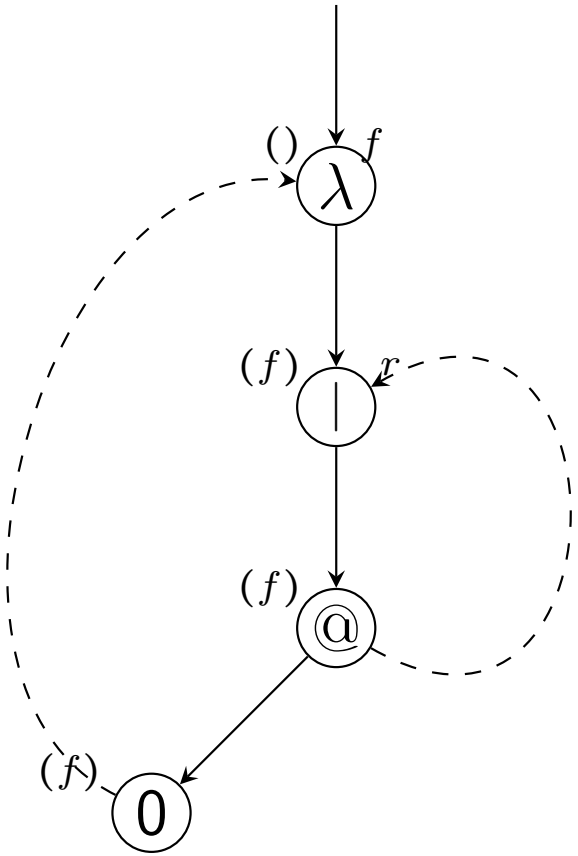
Readback: example (fix)



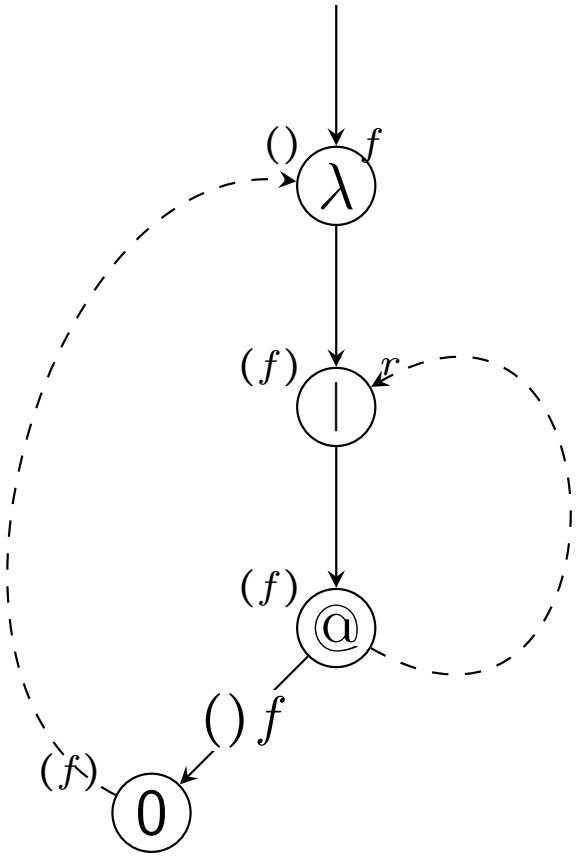
Readback: example (fix)

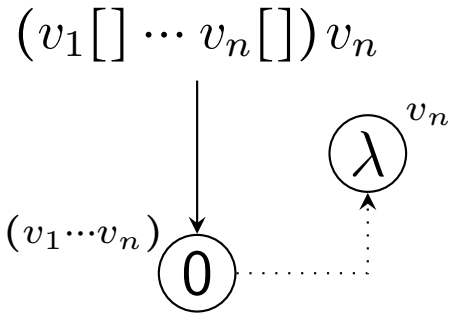


Readback: example (fix)

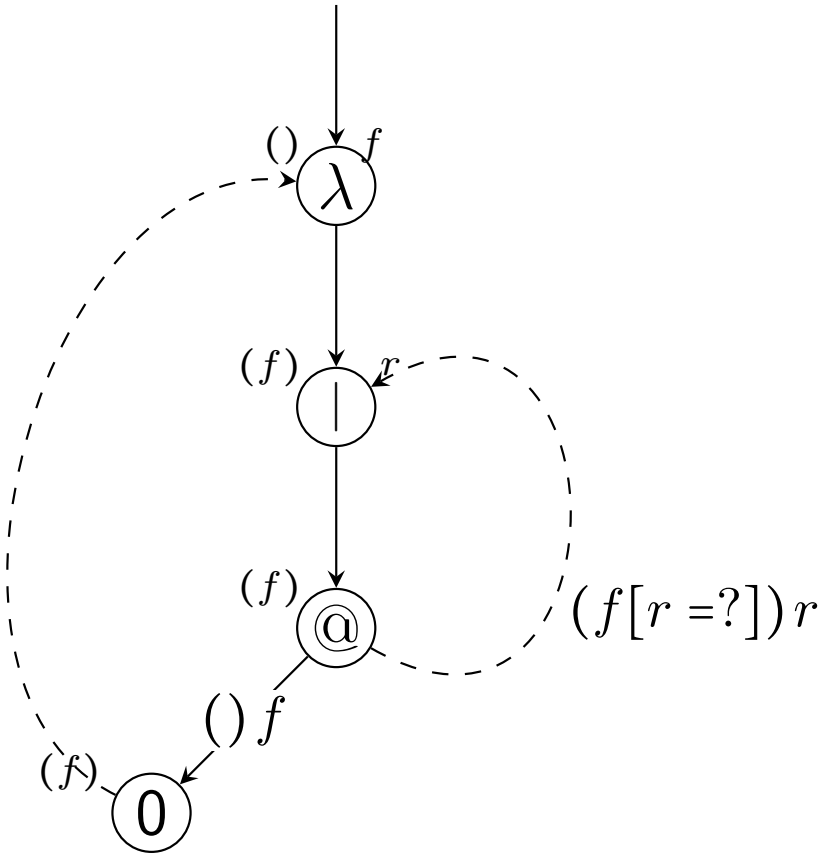


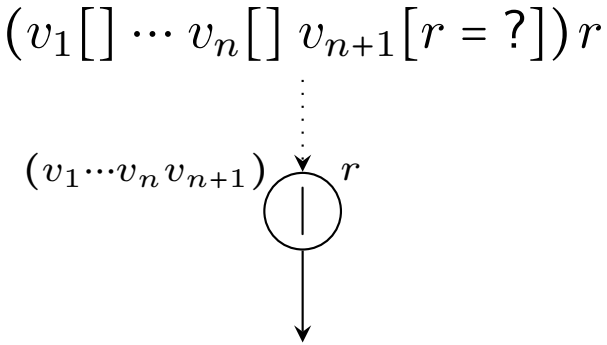
Readback: example (fix)



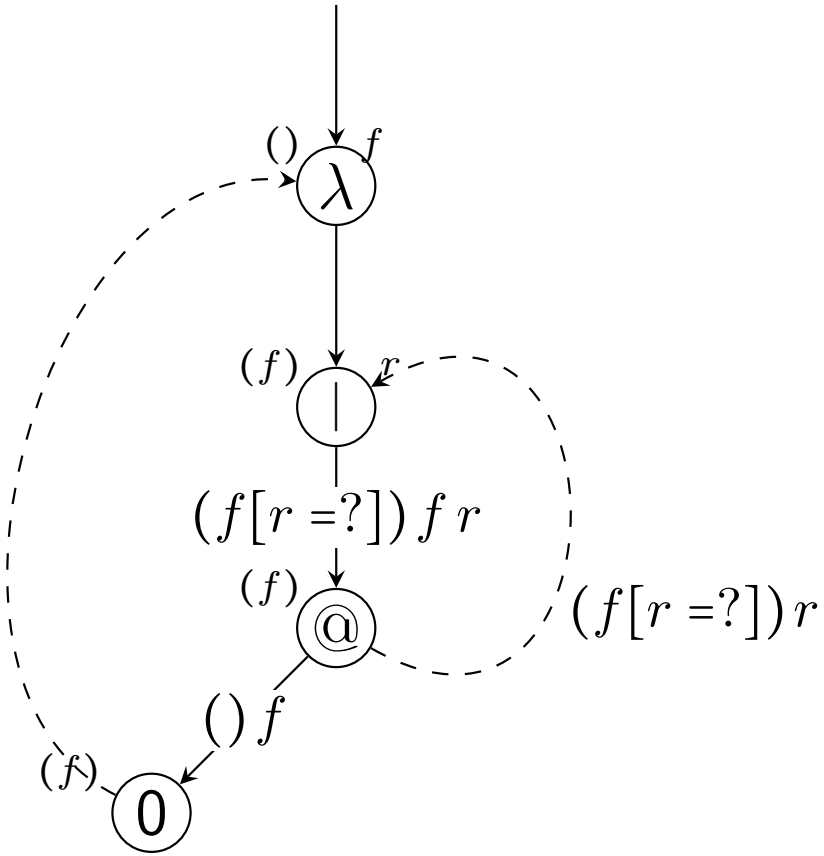


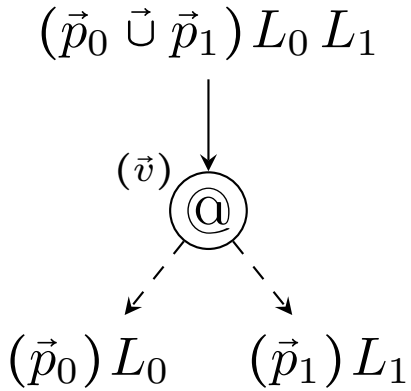
Readback: example (fix)



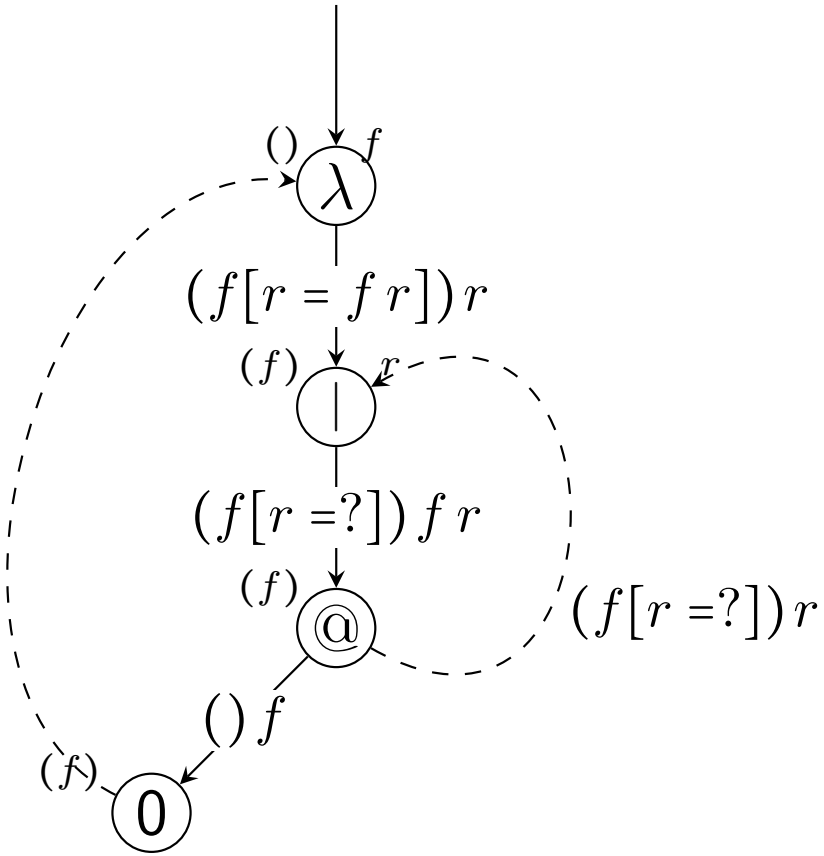


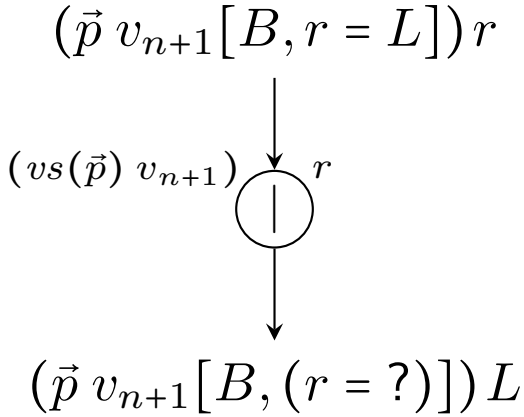
Readback: example (fix)





Readback: example (fix)





Readback: example (fix)