

# Closing the Image of the Process Interpretation of 1-Free Regular Expressions Under Bisim. Collapse

Clemens Grabmayer

<https://clegra.github.io>

Department of Computer Science



L'Aquila, Italy

TERMGRAPH 2024

Luxembourg

April 7, 2024

# Overview

- ▶ 1-free regular expressions (with unary/binary star)
- ▶ process interpretation/semantics of regular expressions
  - ▶ expressible/not expressible process graphs

# Overview

- ▶ 1-free regular expressions (with unary/binary star)
- ▶ process interpretation/semantics of regular expressions
  - ▶ expressible/not expressible process graphs
- ▶ loop existence and elimination property (LEE) (G/Fokkink, 2020)
  - ▶ interpretation/extraction correspondences with 1-free reg. expr's
  - ▶ LEE is preserved under bisimulation collapse
- ▶ Q: Image of process interpretation of 1-free regular expressions closed under bisimulation collapse?

# Overview

- ▶ 1-free regular expressions (with **unary**/binary star)
- ▶ process interpretation/semantics of regular expressions
  - ▶ expressible/not expressible process graphs
- ▶ **loop existence and elimination** property (**LEE**) (G/Fokkink, 2020)
  - ▶ interpretation/**extraction** correspondences with 1-free reg. expr's
  - ▶ **LEE** is preserved under bisimulation collapse
- ▶ Q: Image of process interpretation of 1-free regular expressions **closed under bisimulation collapse**?
  - ▶ A1: **No.** — But ...
- ▶ **compact** process interpretation
- ▶ **refined** expression **extraction**
  - ▶ A2: **compact** process interpretation is **image-closed under collapse**
- ▶ outlook: consequences

# Regular Expressions

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

Regular expressions over alphabet  $A$  with unary Kleene star:

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

# Regular Expressions

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

Regular expressions over alphabet  $A$  with unary Kleene star:

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

- ▶ symbol  $0$  instead of  $\emptyset$ , symbol  $1$  instead of  $\{\emptyset\}$  and  $\epsilon$
- ▶ with unary Kleene star  $^*$ :  $1$  is definable as  $0^*$

# Regular Expressions

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

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

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

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

- ▶ symbol **0** instead of  $\emptyset$ , symbol **1** instead of  $\{\emptyset\}$  and  $\epsilon$
- ▶ with unary Kleene star  $^*$ : **1** is definable as **0<sup>\*</sup>**

# Regular Expressions

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

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

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

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

- ▶ symbol **0** instead of  $\emptyset$ , symbol **1** instead of  $\{\emptyset\}$  and  $\epsilon$
- ▶ with unary Kleene star  $^*$ : **1** is definable as **0<sup>\*</sup>**
- ▶ with binary Kleene star  $^{\otimes}$ : **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:

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

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

- ▶ symbol **0** instead of  $\emptyset$ , symbol **1** instead of  $\{\emptyset\}$  and  $\epsilon$
- ▶ with unary Kleene star  $^*$ : **1** is definable as **0<sup>\*</sup>**
- ▶ with binary Kleene star  $^{\otimes}$ : **1** is **not** definable (in its absence)

# Regular Expressions

Definition ( $\sim$  Kleene, 1951,  $\sim$  Copi-Elgot-Wright, 1958)

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

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

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

- ▶ symbol **0** instead of  $\emptyset$ , symbol **1** instead of  $\{\emptyset\}$  and  $\epsilon$
- ▶ with unary Kleene star  $^*$ : **1** is definable as **0<sup>\*</sup>**
- ▶ with binary Kleene star  $^{\otimes}$ : **1** is **not** definable (in its absence)

Definition

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

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

# Regular Expressions

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

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

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

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

- ▶ symbol **0** instead of  $\emptyset$ , symbol **1** instead of  $\{\emptyset\}$  and  $\epsilon$
- ▶ with unary Kleene star  $^*$ : **1** is definable as **0<sup>\*</sup>**
- ▶ with binary Kleene star  $^{\otimes}$ : **1** is **not** definable (in its absence)

Definition

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

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

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

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

$0 \xrightarrow{P}$  deadlock  $\delta$ , no termination

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

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

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

$0 \xrightarrow{P}$  deadlock  $\delta$ , no termination

$1 \xrightarrow{P}$  empty-step process  $\epsilon$ , 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  $P(e)$ )

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

$0 \xrightarrow{P}$  deadlock  $\delta$ , no termination

$1 \xrightarrow{P}$  empty-step process  $\epsilon$ , 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  $P(e)$ )

$e_1^{\otimes} e_2 \xrightarrow{P}$  (*iteration-exit*) repeat (terminate or execute  $P(e_1)$ ),  
then  $P(e_2)$

# Process semantics $\llbracket \cdot \rrbracket_P$ of regular expressions *(Milner, 1984)*

$0 \xrightarrow{P}$  deadlock  $\delta$ , no termination

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

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

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

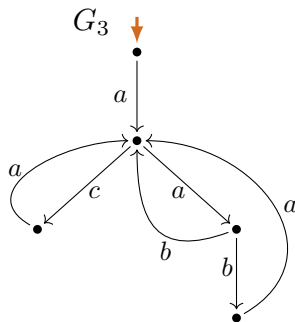
$e_1 \cdot e_2 \xrightarrow{P}$  (*sequentialization*) execute  $P(e_1)$ , then  $P(e_2)$

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

$e_1^{\otimes} e_2 \xrightarrow{P}$  (*iteration-exit*) repeat (terminate or execute  $P(e_1)$ ),  
then  $P(e_2)$

$\llbracket e \rrbracket_P := \llbracket P(e) \rrbracket_{\leftrightarrow}$  (*bisimilarity* equivalence class of process  $P(e)$ )

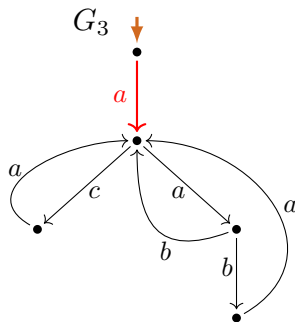
# $P$ -expressibility and $[[\cdot]]_P$ -expressibility (examples)



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

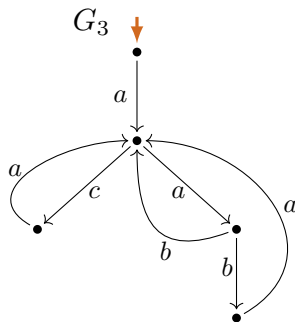


# $P$ -expressibility and $[[\cdot]]_P$ -expressibility (examples)



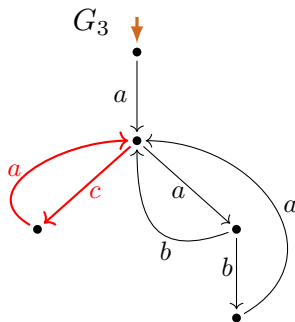
$$P\left( \overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^*}^f \cdot 0 \right)$$

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



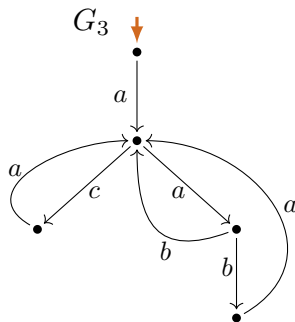
$$P\left( \overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^*}^f \cdot 0 \right)$$

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



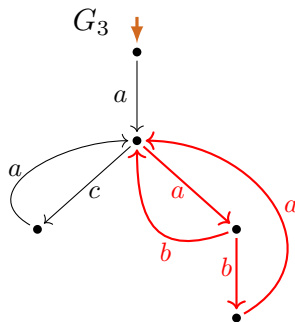
$$P\left(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^f\right)$$

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



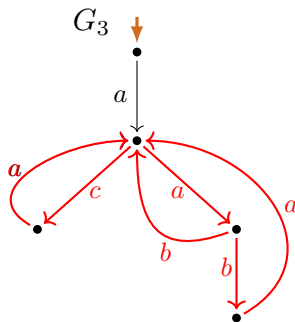
$$P\left( \overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^*}^f \cdot 0 \right)$$

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



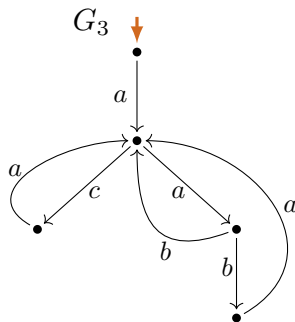
$$P\left(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^*}^f \cdot 0\right)$$

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



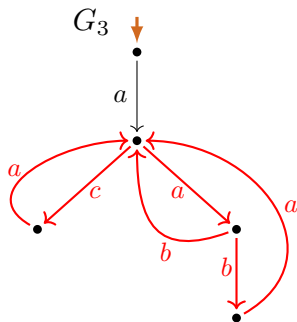
$$P\left(\overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^*}^f \cdot 0\right)$$

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



$$\begin{array}{l}
 \overbrace{P\left( (a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^* \cdot 0 \right)}^f \\
 P\left( a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\otimes} 0 \right)
 \end{array}$$

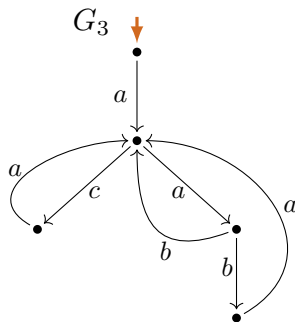
# $P$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)



$$\begin{array}{c}
 \overbrace{\phantom{P\left( (a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^* \cdot 0 \right)}}^f \\
 P\left( (a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^* \cdot 0 \right) \\
 P\left( a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\otimes 0} \right)
 \end{array}$$



# $P$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)

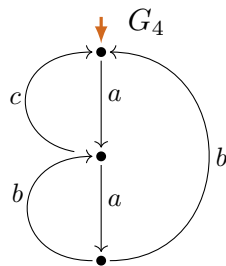
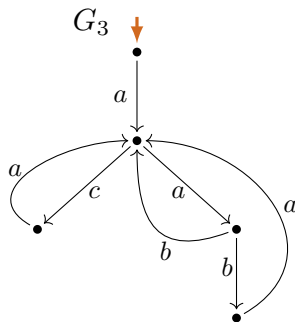


$$\overbrace{P\left( (a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^* \cdot 0 \right)}^f$$

$$P\left( a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\otimes 0} \right)$$

$$G_3 \in \llbracket f \rrbracket_P$$

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

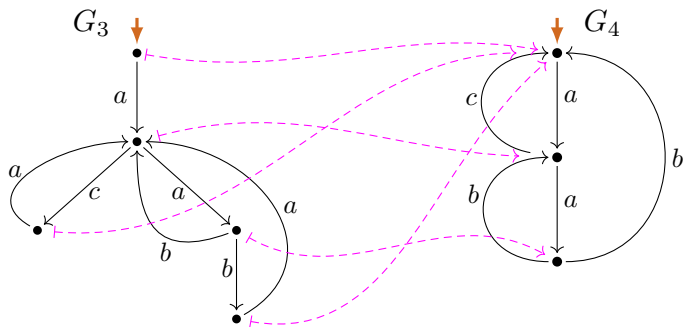


$$\overbrace{P\left( (a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^* \cdot 0 \right)}^f$$

$$P\left( a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\otimes} 0 \right)$$

$$G_3 \in [[f]]_P$$

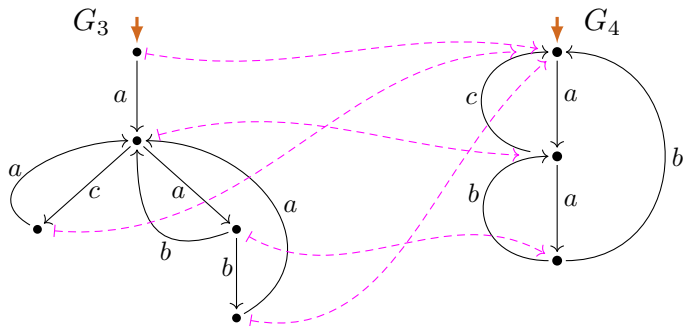
# $P$ -expressibility and $[[\cdot]]_P$ -expressibility (examples)



$$\begin{aligned}
 & \overbrace{P\left( (a \cdot (c \cdot a + a \cdot (b + b \cdot a)))^* \cdot 0 \right)}^f \\
 & P\left( a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\otimes 0} \right)
 \end{aligned}$$

$$G_3 \in [[f]]_P$$

# $P$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)



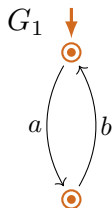
$$P\left( \overbrace{(a \cdot (c \cdot a + a \cdot (b + b \cdot a))^*) \cdot 0}^f \right)$$

$$P\left( a \cdot (c \cdot a + a \cdot (b + b \cdot a))^{\otimes 0} \right)$$

$$G_3 \in \llbracket f \rrbracket_P$$

$$G_4 \in \llbracket f \rrbracket_P$$

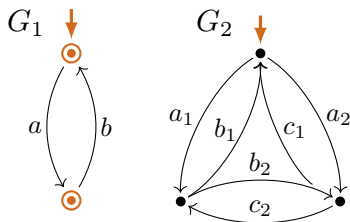
# $P$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)



**not**  $P$ -expressible

**not**  $\llbracket \cdot \rrbracket_P$ -expressible

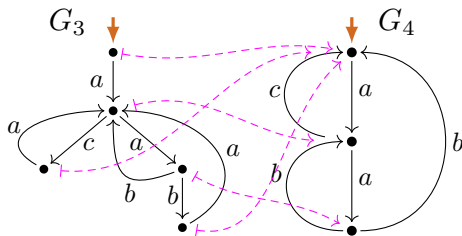
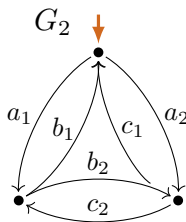
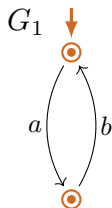
# $P$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)



**not**  $P$ -expressible

**not**  $\llbracket \cdot \rrbracket_P$ -expressible

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



**not**  $P$ -expressible

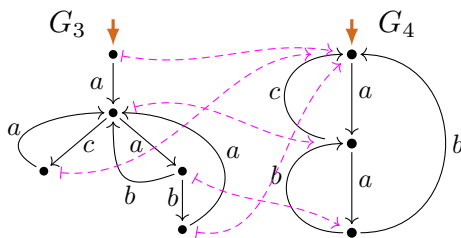
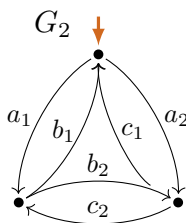
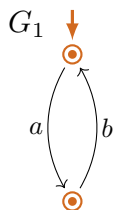
**not**  $[[\cdot]]_P$ -expressible

$P$ -expressible

$[[\cdot]]_P$ -expressible

$[[\cdot]]_P$ -expressible

# $P$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)



**not**  $P$ -expressible

**not**  $\llbracket \cdot \rrbracket_P$ -expressible

$P$ -expressible

$\llbracket \cdot \rrbracket_P$ -expressible

**?**

$\llbracket \cdot \rrbracket_P$ -expressible



# Process interpretation $P$ (formal definition)

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

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

# Process interpretation $P$ (formal definition)

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

$$\begin{array}{c}
 \frac{}{a \xrightarrow{a} 1} \quad \frac{e_i \xrightarrow{a} e'_i}{e_1 + e_2 \xrightarrow{a} e'_i} \quad (i \in \{1, 2\}) \\
 \\
 \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}
 \end{array}$$

# Process interpretation $P$ (formal definition)

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

$$\begin{array}{c}
 \overline{1 \Downarrow} \qquad \overline{e_i \Downarrow} \quad (i \in \{1, 2\}) \quad \overline{e_1 \Downarrow \quad e_2 \Downarrow} \quad \overline{(e^*) \Downarrow} \\
 \overline{(e_1 + e_2) \Downarrow} \qquad \overline{(e_1 \cdot e_2) \Downarrow} \\
 \\
 \overline{a \xrightarrow{a} 1} \qquad \overline{e_i \xrightarrow{a} e'_i} \quad (i \in \{1, 2\}) \\
 \overline{e_1 + e_2 \xrightarrow{a} e'_i} \\
 \\
 \overline{e \xrightarrow{a} e'} \\
 \overline{e^* \xrightarrow{a} e' \cdot e^*}
 \end{array}$$

# Process interpretation $P$ (formal definition)

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

$$\begin{array}{c}
 \overline{1 \Downarrow} \qquad \overline{e_i \Downarrow} \quad (i \in \{1, 2\}) \quad \overline{e_1 \Downarrow \quad e_2 \Downarrow} \quad \overline{(e^*) \Downarrow} \\
 \overline{(e_1 + e_2) \Downarrow} \qquad \overline{e_1 \Downarrow \quad e_2 \Downarrow} \quad \overline{(e_1 \cdot e_2) \Downarrow} \\
 \overline{a \xrightarrow{a} 1} \qquad \overline{e_i \xrightarrow{a} e'_i} \quad (i \in \{1, 2\}) \\
 \overline{e_1 \xrightarrow{a} e'_1} \quad \overline{e_1 \Downarrow \quad e_2 \xrightarrow{a} e'_2} \quad \overline{e \xrightarrow{a} e'} \\
 \overline{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \quad \overline{e_1 \cdot e_2 \xrightarrow{a} e'_2} \quad \overline{e^* \xrightarrow{a} e' \cdot e^*}
 \end{array}$$

# Process interpretation $P$ (formal definition)

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

$$\begin{array}{c}
 \overline{1 \Downarrow} \qquad \frac{e_i \Downarrow}{(e_1 + e_2) \Downarrow} \ (i \in \{1, 2\}) \qquad \frac{e_1 \Downarrow \quad e_2 \Downarrow}{(e_1 \cdot e_2) \Downarrow} \qquad \overline{(e^*) \Downarrow} \\
 \\
 \frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} e'_i}{e_1 + e_2 \xrightarrow{a} e'_i} \ (i \in \{1, 2\}) \\
 \\
 \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \qquad \frac{e_1 \Downarrow \quad e_2 \xrightarrow{a} e'_2}{e_1 \cdot e_2 \xrightarrow{a} e'_2} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}
 \end{array}$$

## Definition

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

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

# Interpretation/extraction correspondences with LEE

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

$(\text{Int})_P^{(+)}$ : *P-expressible graphs have the structural property LEE*

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

# Interpretation/extraction correspondences with LEE

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

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

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

**(Extr) $_P$** : **LEE** *implies*  $\llbracket \cdot \rrbracket_P$ -*expressibility*

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

# Interpretation/extraction correspondences with LEE

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

**(Int)<sub>P</sub><sup>(+)</sup>**: *P-expressible graphs have the structural property LEE*

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

**(Extr)<sub>P</sub>**: **LEE** *implies*  $\llbracket \cdot \rrbracket_P$ -*expressibility*

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

**(Coll)**: **LEE** *is preserved under collapse*

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



# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

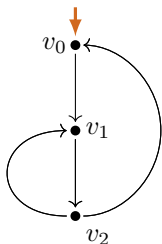
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to **it**.
- (L3) Termination is **only** possible at the **start vertex**.

# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to **it**.
- (L3) Termination is **only** possible at the **start vertex**.

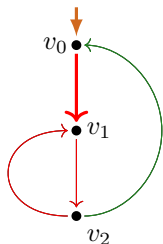


# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to **it**.
- (L3) Termination is **only** possible at the **start vertex**.



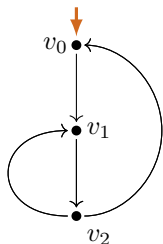
(L1), ~~(L2)~~

# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to **it**.
- (L3) Termination is **only** possible at the **start vertex**.



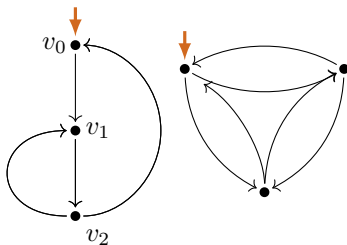
(L1), ~~(L2)~~

# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



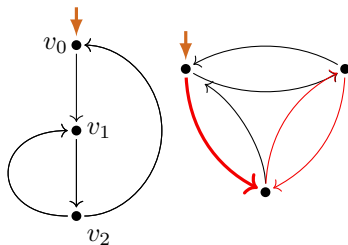
(L1), ~~(L2)~~

# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



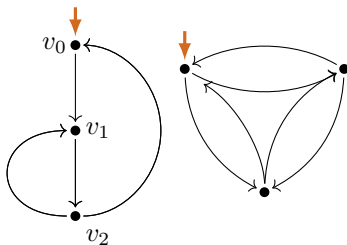
(L1), ~~(L2)~~

# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



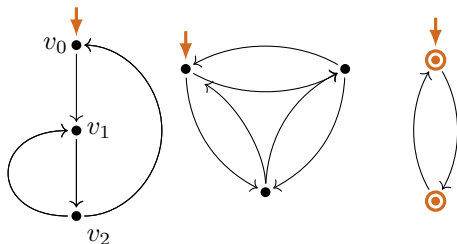
(L1), ~~(L2)~~

# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~

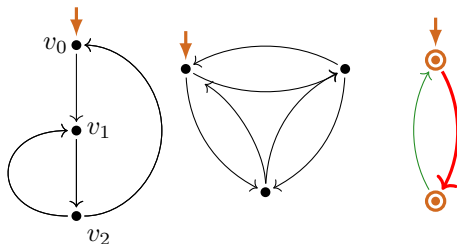


# Loop charts (interpretations of innermost iterations)

## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



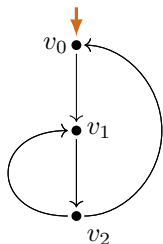
(L1), ~~(L2)~~

# Loop charts (interpretations of innermost iterations)

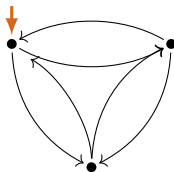
## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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

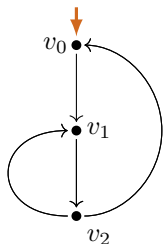


# Loop charts (interpretations of innermost iterations)

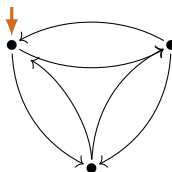
## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



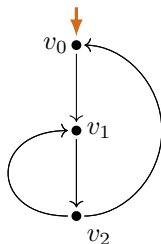
(L1), (L2), ~~(L3)~~

# Loop charts (interpretations of innermost iterations)

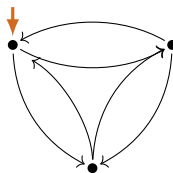
## Definition

A chart is a **loop chart** if:

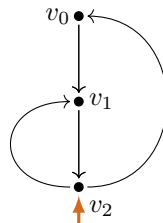
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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

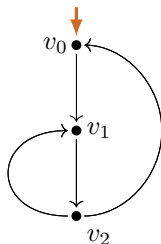


# Loop charts (interpretations of innermost iterations)

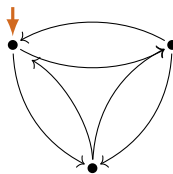
## Definition

A chart is a **loop chart** if:

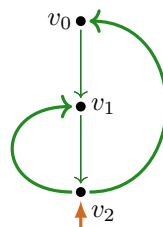
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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

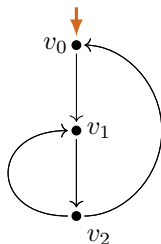


# Loop charts (interpretations of innermost iterations)

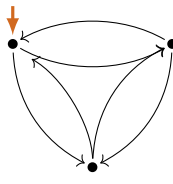
## Definition

A chart is a **loop chart** if:

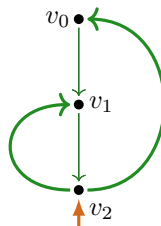
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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



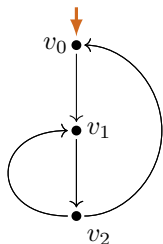
**loop chart**

# Loop charts (interpretations of innermost iterations)

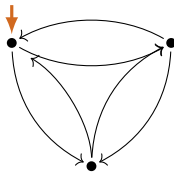
## Definition

A chart is a **loop chart** if:

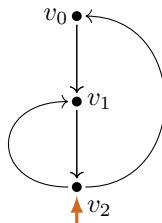
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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



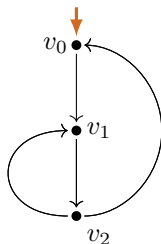
**loop chart**

# Loop charts (interpretations of innermost iterations)

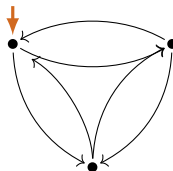
## Definition

A chart is a **loop chart** if:

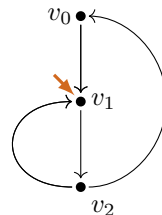
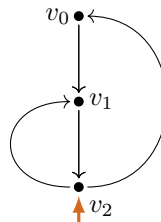
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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



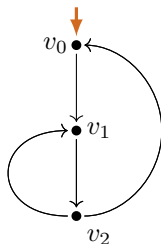


# Loop charts (interpretations of innermost iterations)

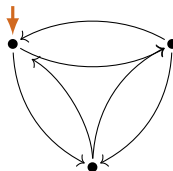
## Definition

A chart is a **loop chart** if:

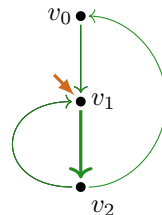
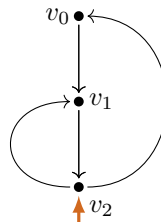
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



(L1), ~~(L2)~~



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

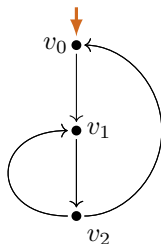


# Loop charts (interpretations of innermost iterations)

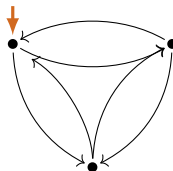
## Definition

A chart is a **loop chart** if:

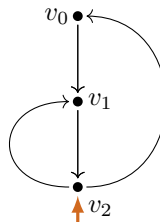
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



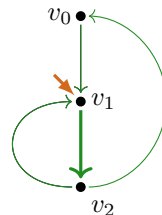
(L1), ~~(L2)~~



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



loop chart



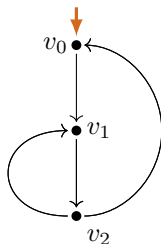
loop chart

# Loop charts (interpretations of innermost iterations)

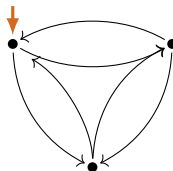
## Definition

A chart is a **loop chart** if:

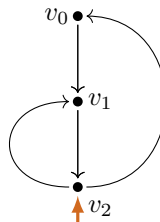
- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



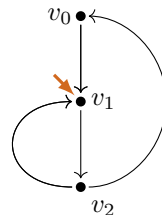
(L1), ~~(L2)~~



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



loop chart



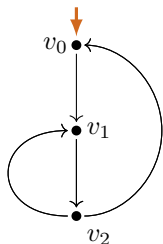
loop chart

# Loop charts (interpretations of innermost iterations)

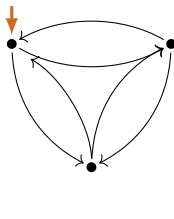
## Definition

A chart is a **loop chart** if:

- (L1) There is an infinite path from the **start vertex**.
- (L2) Every infinite path from the **start vertex** returns to it.
- (L3) Termination is **only** possible at the **start vertex**.



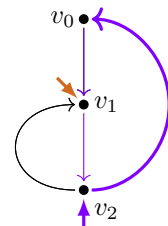
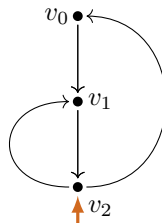
(L1), ~~(L2)~~



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

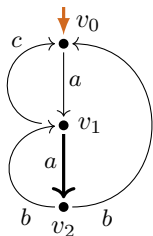


loop chart

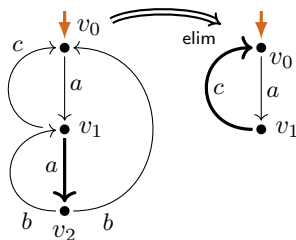


loop subchart

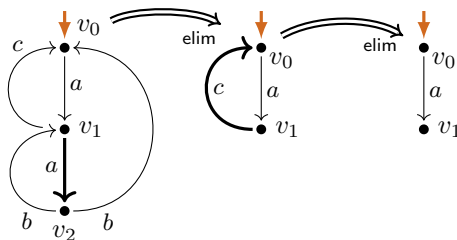
# Loop existence and elimination



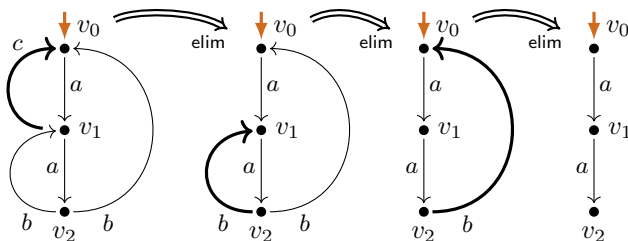
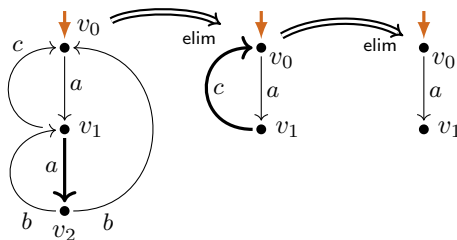
# Loop existence and elimination



# Loop existence and elimination



# Loop existence and elimination





# LEE

## Definition

A chart  $\mathcal{C}$  satisfies **LEE** (*loop existence and elimination*) if:

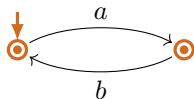
$$\exists \mathcal{C}_0 \left( \mathcal{C} \xrightarrow[\text{elim}]{*} \mathcal{C}_0 \not\rightarrow_{\text{elim}} \right. \\ \left. \wedge \mathcal{C}_0 \text{ permits no infinite path} \right).$$

# LEE

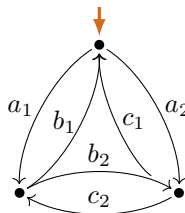
## Definition

A chart  $\mathcal{C}$  satisfies **LEE** (*loop existence and elimination*) if:

$$\exists \mathcal{C}_0 \left( \mathcal{C} \xrightarrow{\text{elim}^*} \mathcal{C}_0 \not\xrightarrow{\text{elim}} \right. \\ \left. \wedge \mathcal{C}_0 \text{ permits no infinite path} \right).$$

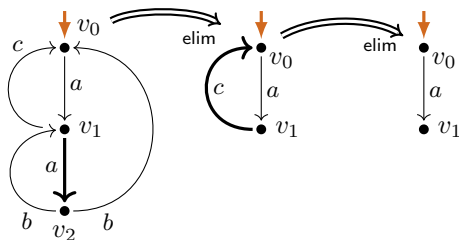


**LEE**

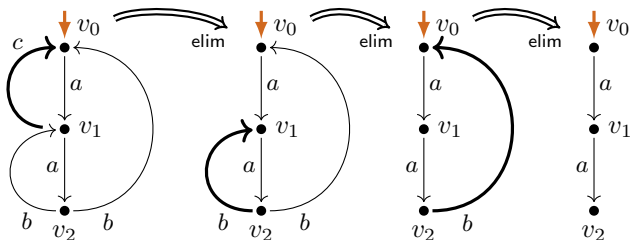
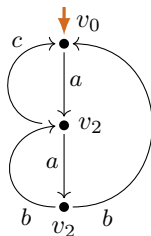


**LEE**

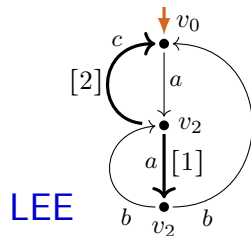
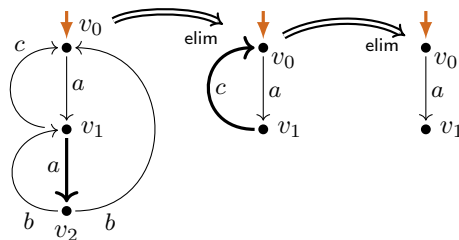
## LEE



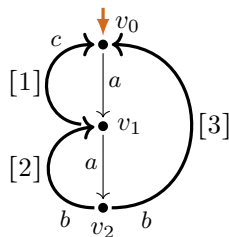
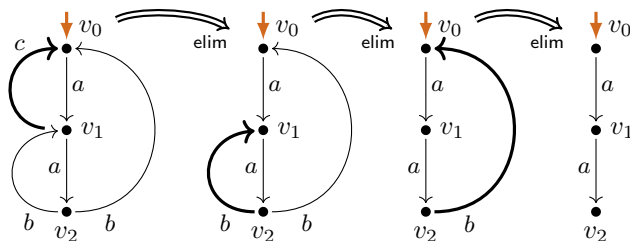
LEE



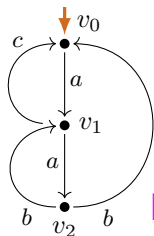
## LEE



LEE

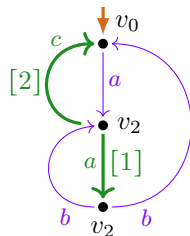


# LEE witness and LEE-charts

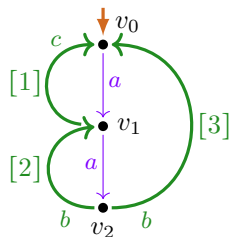


LEE-chart

LEE-witness



LEE-witness



# Properties of LEE-charts

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

A process graph  $G$

is  $\llbracket \cdot \rrbracket_P$ -expressible by a 1-free regular expression

(i.e.  $P$ -expressible modulo bisimilarity by a 1-free reg. expr.)

if and only if

the bisimulation collapse of  $G$  satisfies LEE.

# Properties of LEE-charts

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

A process graph  $G$

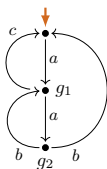
is  $\llbracket \cdot \rrbracket_P$ -expressible by a 1-free regular expression

(i.e.  $P$ -expressible modulo bisimilarity by a 1-free reg. expr.)

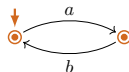
if and only if

the bisimulation collapse of  $G$  satisfies LEE.

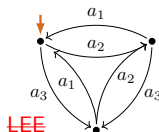
Hence  $\llbracket \cdot \rrbracket_P$ -expressible | **not**  $\llbracket \cdot \rrbracket_P$ -expressible by 1-free regular expressions:



LEE

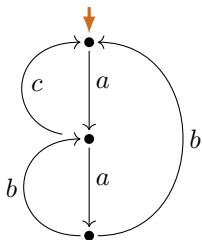


LEE



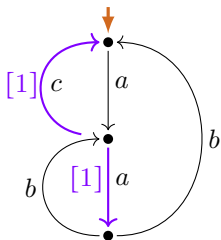
LEE

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

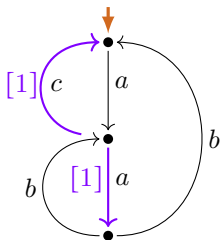
 $G_4$ 



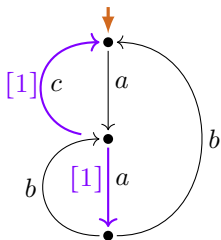
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)

 $\widehat{G}_4$ 

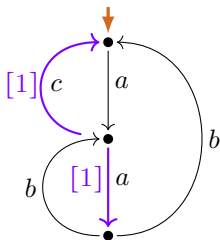
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)

 $\widehat{G}_4$ 

 $($ 
 $)^* \cdot 0$

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

 $\widehat{G}_4$ 

 $($ 
 $\downarrow a$ 
 $)^* \cdot 0$

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

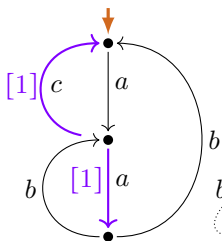
 $\widehat{G}_4$ 


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

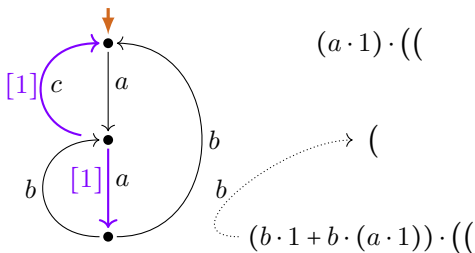
$$( \quad )^* \cdot 0$$

 $\downarrow a$

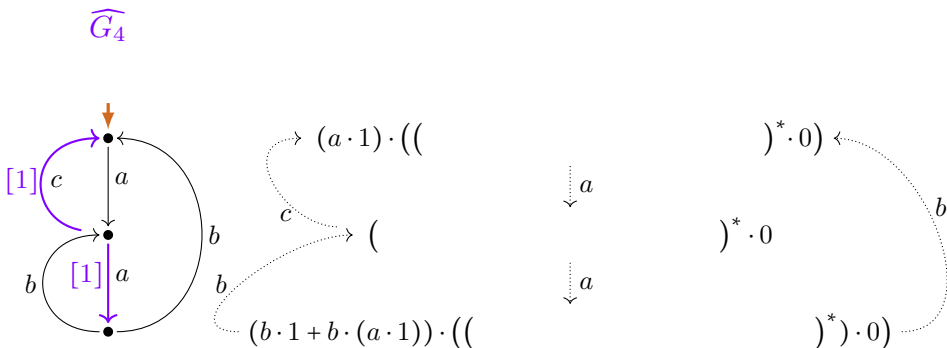
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)

 $\widehat{G}_4$ 

 $(a \cdot 1) \cdot (($ 
 $)$ 
 $\downarrow a$ 
 $)^* \cdot 0) \leftarrow$   
 $)^* \cdot 0$

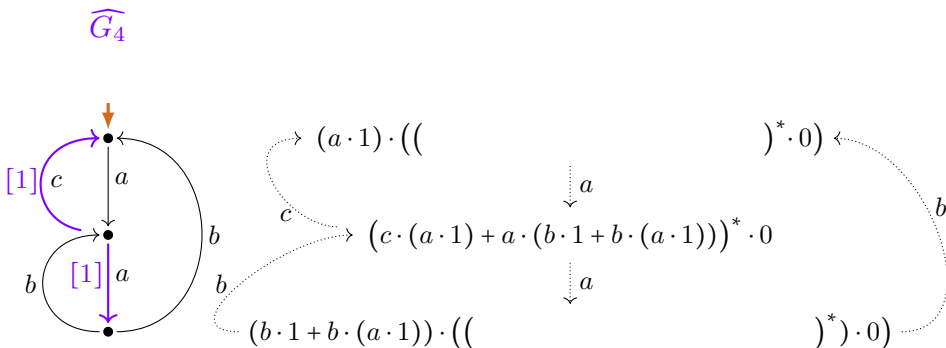
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)

 $\widehat{G}_4$ 

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

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

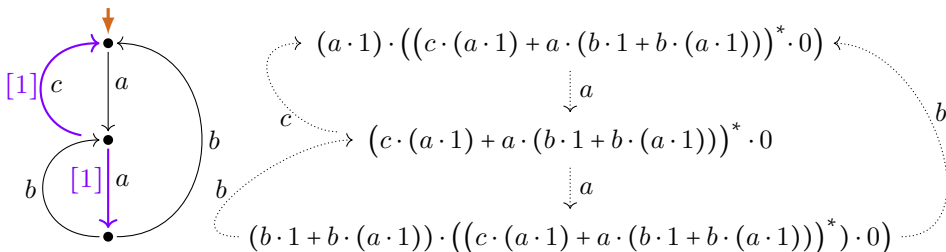


# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)

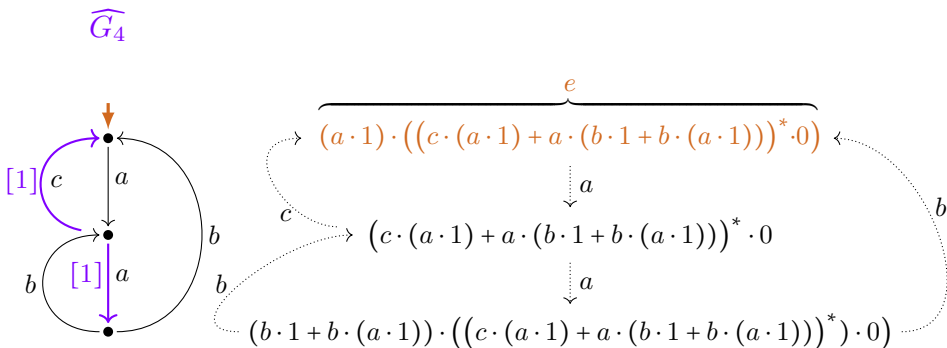




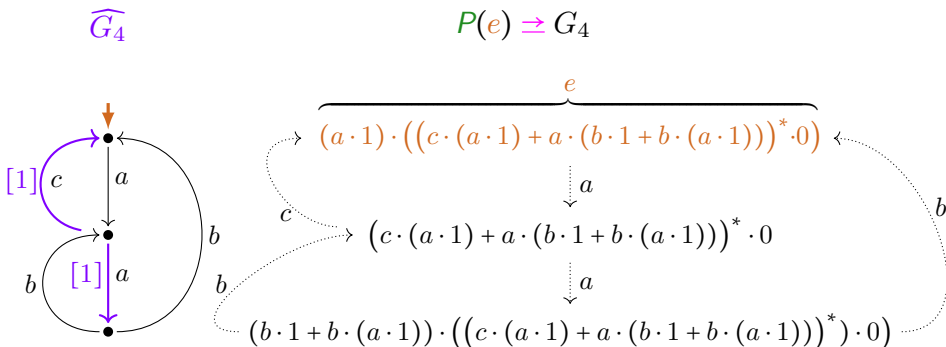
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)

 $\widehat{G}_4$ 


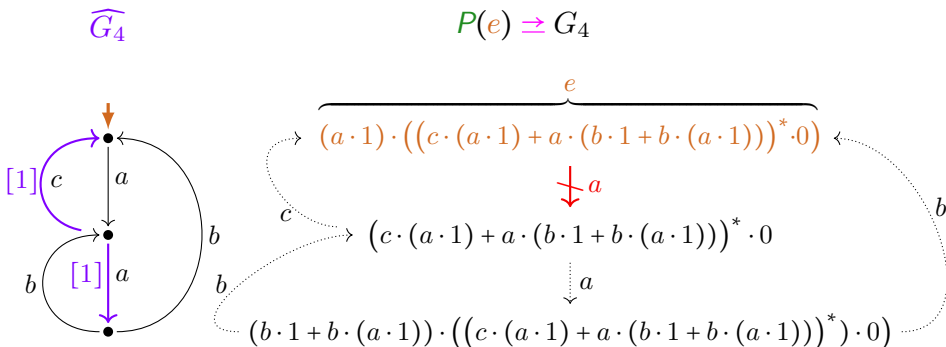
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)



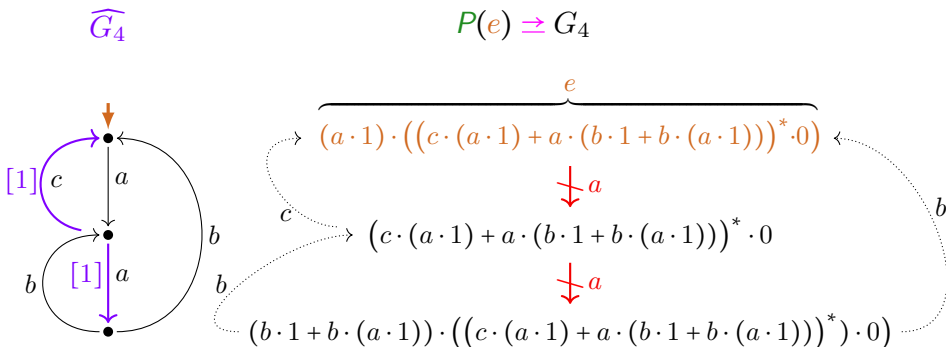
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)



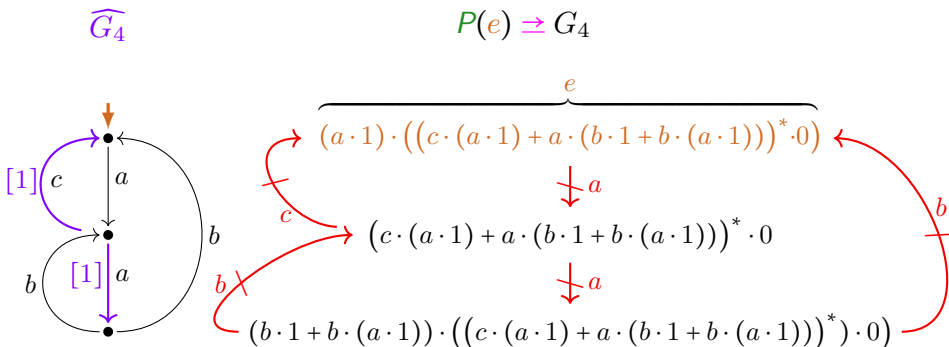
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)



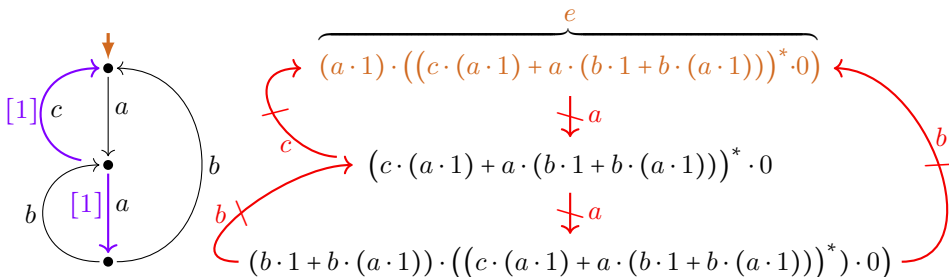
# Expression extraction using LEE (G/Fokkink 2020, G 2021/22)



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



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

 $\widehat{G}_4$ 
 $P(e) \Rightarrow G_4 \not\Rightarrow P(e)$ 


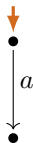
# Interpretation of extracted expression

 $G_5$  $P(e) = G_5$ 

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



# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5$ 


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

# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5$ 


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

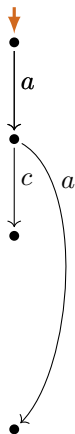
 $\downarrow a$ 

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

 $\downarrow c$ 

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

# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5$ 


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

 $\downarrow a$ 

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

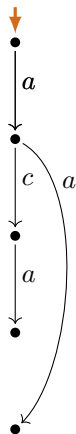
 $\downarrow c$ 

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

 $a$ 

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

# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5$ 


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

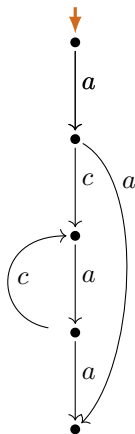
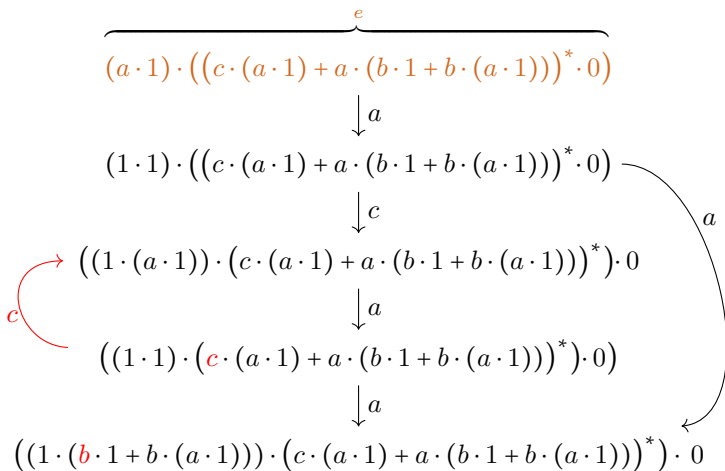
# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5$ 

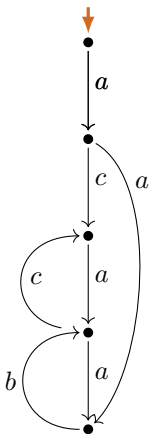
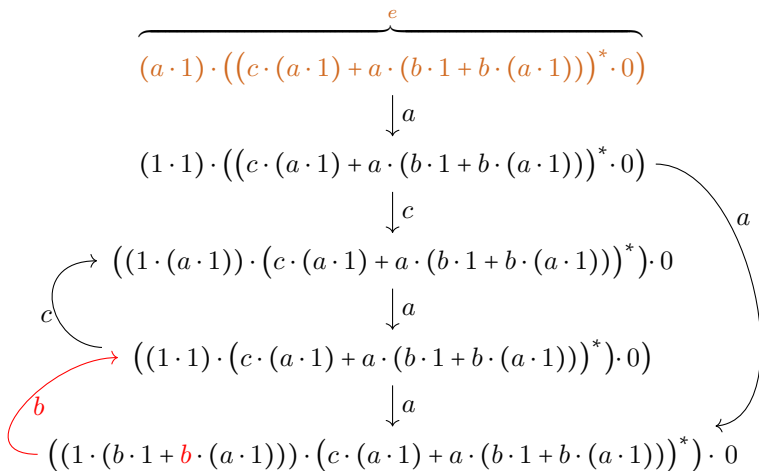

$$\begin{array}{c}
 \overbrace{(a \cdot 1) \cdot ((c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)))^* \cdot 0)}^e \\
 \downarrow a \\
 (1 \cdot 1) \cdot ((c \cdot (a \cdot 1) + a \cdot (b \cdot 1 + b \cdot (a \cdot 1)))^* \cdot 0) \\
 \downarrow c \\
 ((1 \cdot (a \cdot 1)) \cdot (c \cdot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

A curved arrow labeled  $a$  points from the second expression to the final expression.

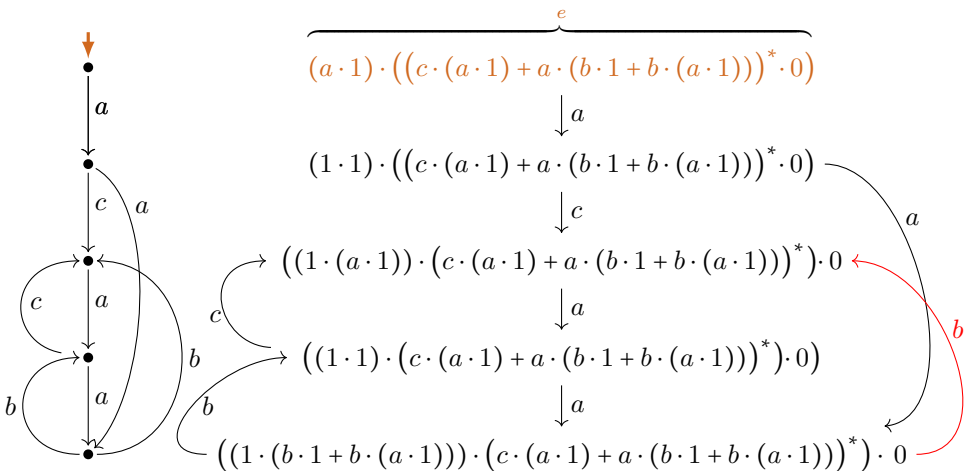
# Interpretation of extracted expression

 $G_5$ 

 $P(e) = G_5$ 


# Interpretation of extracted expression

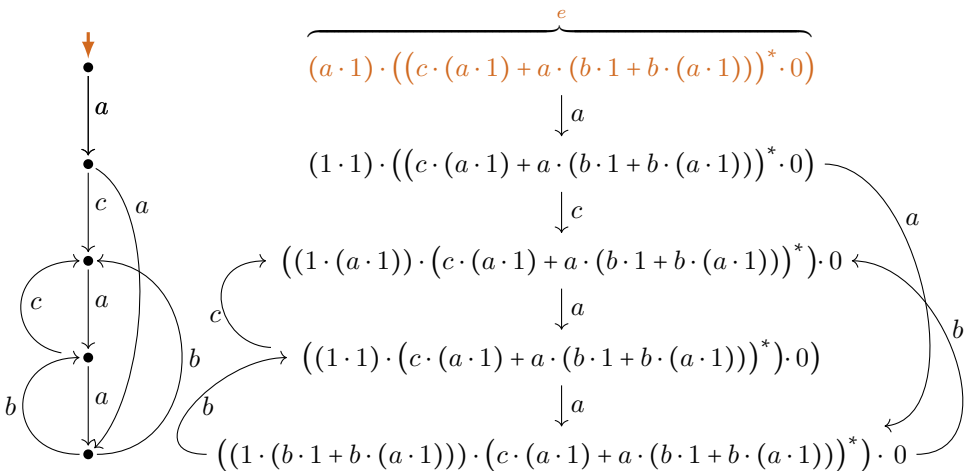
 $G_5$ 

 $P(e) = G_5$ 


# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5$ 




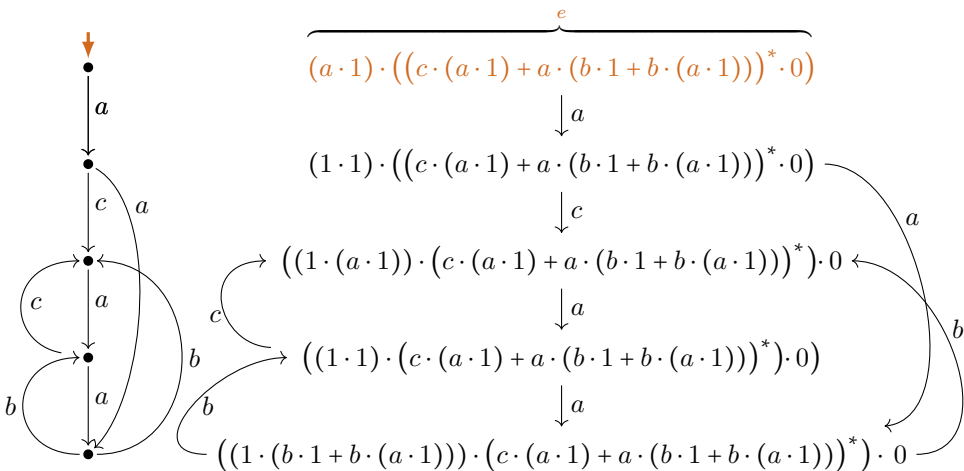
# Interpretation of extracted expression

 $G_5$ 
 $P(e) = G_5 \xrightarrow{\text{pink}} G_4$ 


# Interpretation of extracted expression

 $G_5$ 

$$P(e) = G_5 \xrightarrow{\text{pink}} G_4 \not\equiv G_5$$



# Under-Star-/1-Free regular expressions

## Definition

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

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

# Under-Star-/1-Free regular expressions

## Definition

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

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

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

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

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

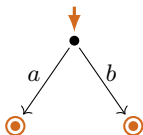
 $P(uf)$ 

 $P(uf)$ 

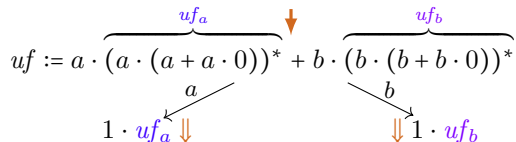
$$uf := a \cdot \overbrace{(a \cdot (a + a \cdot 0))^*}^{uf_a} + b \cdot \overbrace{(b \cdot (b + b \cdot 0))^*}^{uf_b}$$

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

$P(uf)$

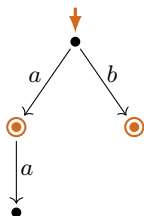


$P(uf)$

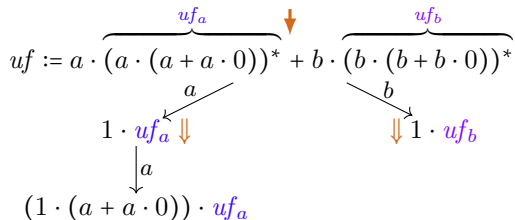


# Image of $P$ is **not** closed under bisimulation collapse

$P(uf)$

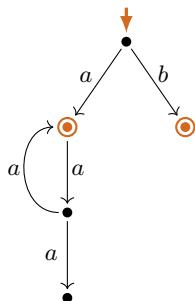


$P(uf)$

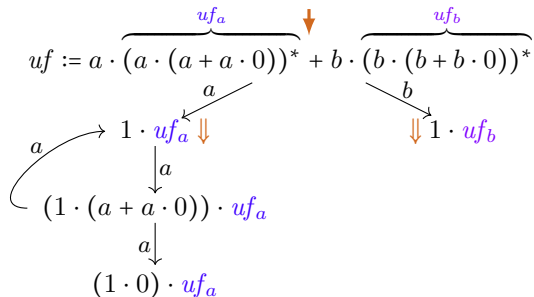


# Image of $P$ is **not** closed under bisimulation collapse

$P(uf)$



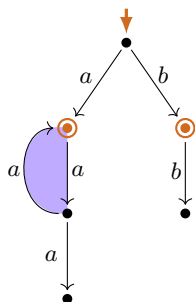
$P(uf)$



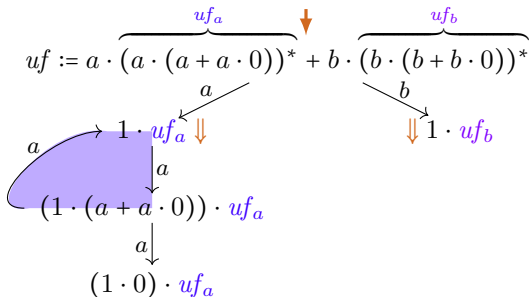


# Image of $P$ is **not** closed under bisimulation collapse

$P(uf)$

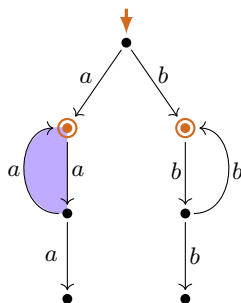


$P(uf)$

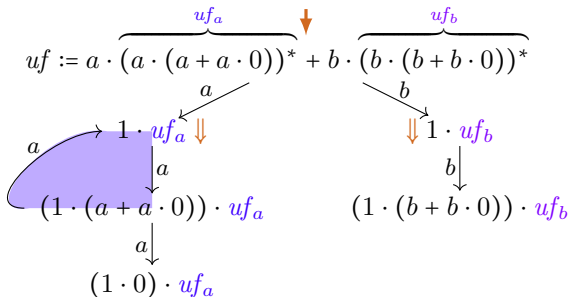


# Image of $P$ is **not** closed under bisimulation collapse

$P(uf)$

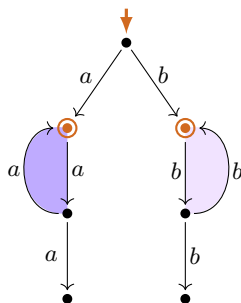


$P(uf)$

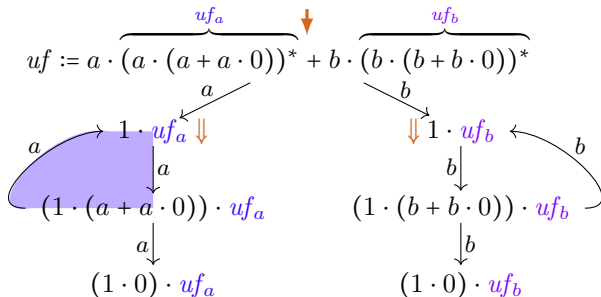


# Image of $P$ is **not** closed under bisimulation collapse

$P(uf)$

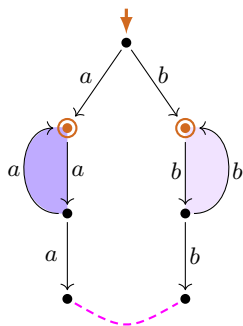


$P(uf)$

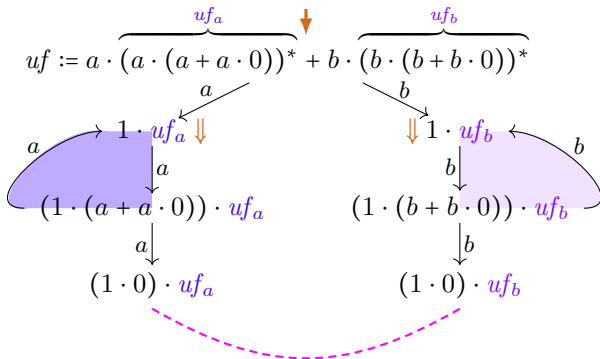


# Image of $P$ is **not** closed under bisimulation collapse

$P(uf)$



$P(uf)$



# Compact process interpretation $P^\bullet$

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

$$\begin{array}{c}
 \frac{}{1 \Downarrow} \qquad \frac{e_i \Downarrow}{(e_1 + e_2) \Downarrow} \ (i \in \{1, 2\}) \qquad \frac{e_1 \Downarrow \quad e_2 \Downarrow}{(e_1 \cdot e_2) \Downarrow} \qquad \frac{}{(e^*) \Downarrow} \\
 \\
 \frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} e'_i}{e_1 + e_2 \xrightarrow{a} e'_i} \ (i \in \{1, 2\}) \\
 \\
 \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \qquad \frac{e_1 \Downarrow \quad e_2 \xrightarrow{a} e'_2}{e_1 \cdot e_2 \xrightarrow{a} e'_2} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}
 \end{array}$$

# Compact process interpretation $P^\bullet$

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

$$\frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2}$$

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

# Compact process interpretation $P^\bullet$

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

$$\frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \text{ (if } e'_1 \text{ is normed)}$$

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

# Compact process interpretation $P^\bullet$

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

$$\begin{array}{c}
 \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \text{ (if } e'_1 \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1} \text{ (if } e'_1 \text{ is not normed)} \\
 \\
 \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*} \text{ (if } e' \text{ is normed)} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e'} \text{ (if } e' \text{ is not normed)}
 \end{array}$$



# Compact process interpretation $P^\bullet$

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

$$\begin{array}{c}
 \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \text{ (if } e'_1 \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1} \text{ (if } e'_1 \text{ is not normed)} \\
 \\
 \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*} \text{ (if } e' \text{ is normed)} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e'} \text{ (if } e' \text{ is not normed)}
 \end{array}$$

## Definition

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

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

# Compact process interpretation $P^\bullet$

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

$$\begin{array}{c}
 \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \text{ (if } e'_1 \text{ is normed)} \qquad \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1} \text{ (if } e'_1 \text{ is not normed)} \\
 \\
 \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*} \text{ (if } e' \text{ is normed)} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e'} \text{ (if } e' \text{ is not normed)}
 \end{array}$$

## Definition

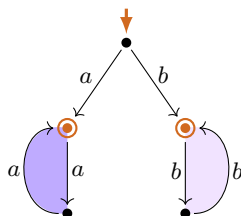
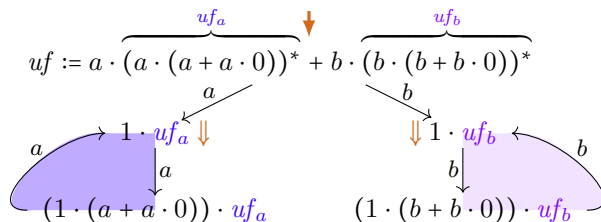
The compact process (graph) interpretation  $P^\bullet(e)$  of a reg. expr's  $e$ :

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

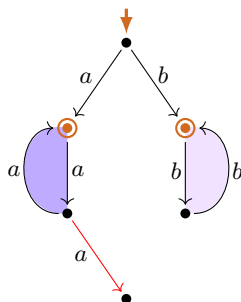
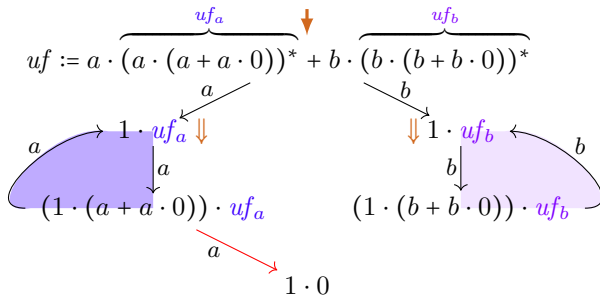
Lemma ( $P^\bullet$  increases sharing;  $P^\bullet, P$  have same bisimulation semantics)

- (i)  $P(e) \Rightarrow P^\bullet(e)$  for all regular expressions  $e$ .
- (ii) ( $G$  is  $\llbracket \cdot \rrbracket_{P^\bullet}$ -expressible  $\iff G$  is  $\llbracket \cdot \rrbracket_P$ -expressible) for all graphs  $G$ .

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

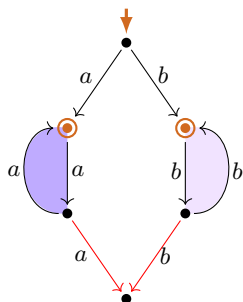
 $P^\bullet(uf)$ 

 $P^\bullet(uf)$ 


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

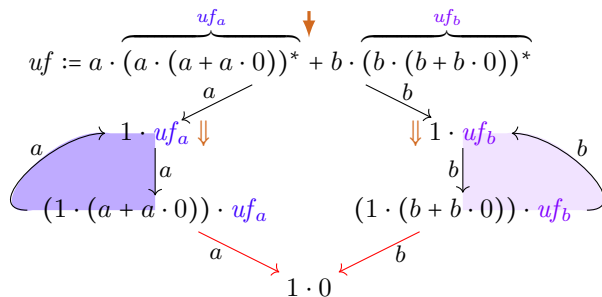
 $P^\bullet(uf)$ 

 $P^\bullet(uf)$ 


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

$P^\bullet(uf)$



$P^\bullet(uf)$



# Interpretation correspondence of $P^\bullet$ with LEE

$(\text{Int})_{P^\bullet}^{(\perp \setminus *)}$ : By *under-star-1-free* expressions  $P^\bullet$ -expressible graphs satisfy LEE:  
Compact process interpretations  $P^\bullet(uf)$   
of *under-star-1-free* regular expressions  $uf$   
are finite process graphs that satisfy LEE.

# Interpretation correspondence of $P^\bullet$ with LEE

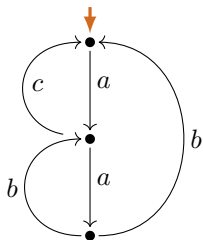
**(Int)** $_{P^\bullet}^{(\perp \setminus *)}$ : By *under-star-1-free* expressions  $P^\bullet$ -expressible graphs satisfy LEE:

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

**(Extr)** $_{P^\bullet}^{(\perp \setminus *)}$ : LEE implies  $\llbracket \cdot \rrbracket_{P^\bullet}$ -expressibility by *under-star-1-free* reg. expr's:

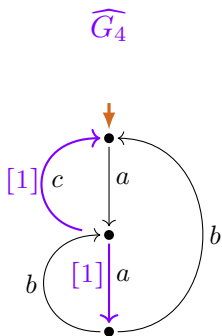
From every finite process graph  $G$  with LEE  
 an *under-star-1-free* regular expression  $uf$  can be extracted  
 such that  $G \Rightarrow P^\bullet(uf)$ .

# Refined extraction expression (example)

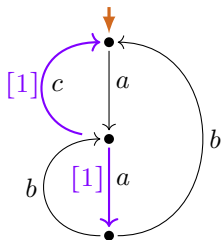
 $G_4$ 



# Refined extraction expression (example)

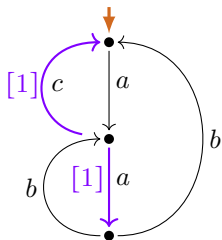


# Refined extraction expression (example)

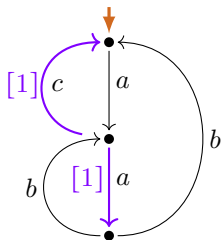
 $\widehat{G}_4$ 


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

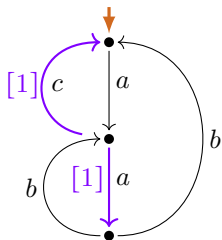
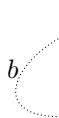
# Refined extraction expression (example)

 $\widehat{G}_4$ 

 $(1 \cdot ($ 
 $\begin{array}{c} \vdots \\ a \\ \vee \end{array}$ 
 $)^*) \cdot 0$

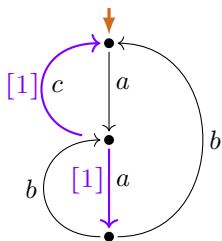
# Refined extraction expression (example)

 $\widehat{G}_4$ 

 $((1 \cdot a) \cdot ( \quad )^*) \cdot 0$ 
 $\begin{array}{c} \vdots \\ a \\ \vee \end{array}$ 
 $(1 \cdot ( \quad )^*) \cdot 0$

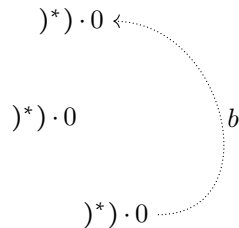
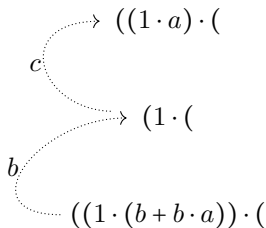
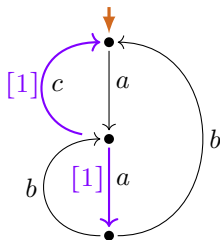
# Refined extraction expression (example)

 $\widehat{G}_4$ 

 $((1 \cdot a) \cdot ($ 
 $(1 \cdot ($ 

 $\downarrow a$ 
 $)^*) \cdot 0 \leftarrow$ 
 $)^*) \cdot 0$ 
 $b$ 

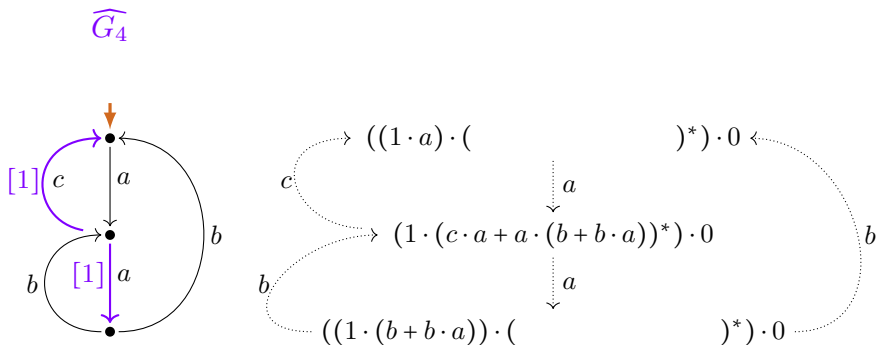

# Refined extraction expression (example)

 $\widehat{G}_4$ 

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

# Refined extraction expression (example)

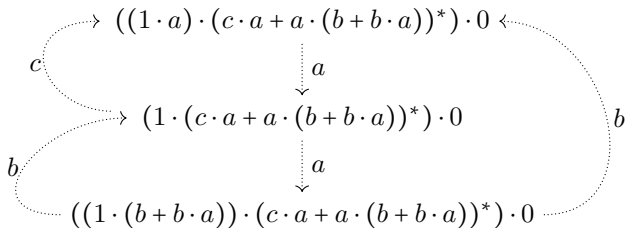
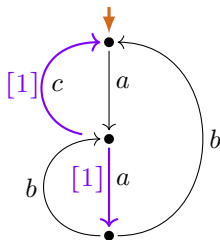
 $\widehat{G}_4$ 


### Refined extraction expression (example)

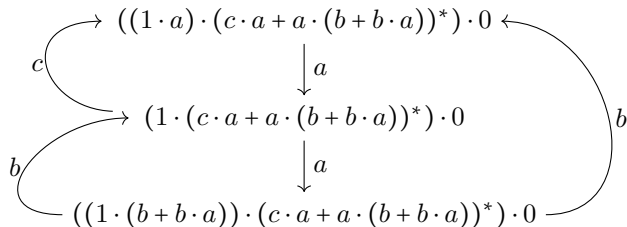
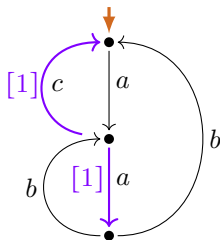




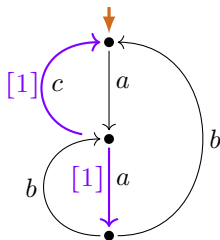
# Refined extraction expression (example)

 $\widehat{G}_4$ 


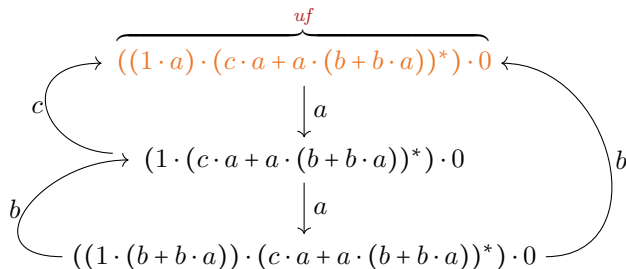
# Refined extraction expression (example)

 $\widehat{G}_4$ 


# Refined extraction expression (example)

 $\widehat{G}_4$ 


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



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

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

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

**(Extr)** $_{P^\bullet}^{(\pm \setminus *)}$ : LEE implies  $\llbracket \cdot \rrbracket_{P^\bullet}$ -expressibility by *under-star-1-free* reg. expr's:

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

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

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

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

$(\text{Extr})_{P^\bullet}^{(\pm \setminus *)}$ : LEE implies  $\llbracket \cdot \rrbracket_{P^\bullet}$ -expressibility by *under-star-1-free* reg. expr's:

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

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

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

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

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

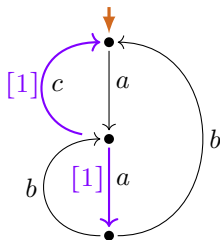
**(Extr)** $_{P^\bullet}^{(\pm \setminus *)}$ : LEE implies  $\llbracket \cdot \rrbracket_{P^\bullet}$ -expressibility by *under-star-1-free* reg. expr's:

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

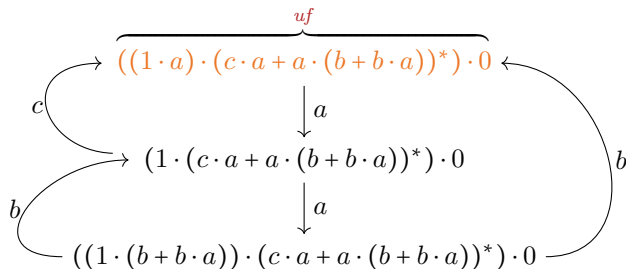
From every finite collapsed process graph  $G$  with LEE  
 an *under-star-1-free* regular expression  $uf$  can be extracted  
 such that  $G \simeq P^\bullet(uf)$ .

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

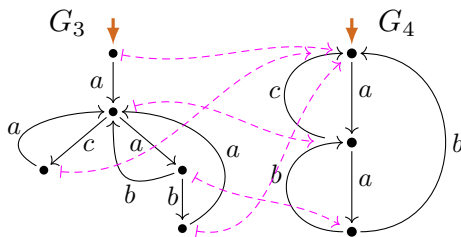
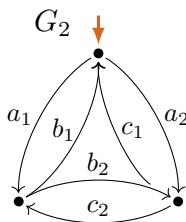
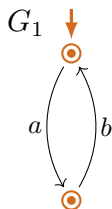
# Refined expression extraction (example)

 $\widehat{G}_4$ 


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



# $P_-/P^\bullet$ -expressibility and $\llbracket \cdot \rrbracket_P$ -expressibility (examples)



not  $P$ -expressible

not  $\llbracket \cdot \rrbracket_P$ -expressible

$P_-/P^\bullet$ -expressible

$\llbracket \cdot \rrbracket_P$ -expressible

$P^\bullet$ -expressible

$\llbracket \cdot \rrbracket_P$ -expressible



# Summary and outlook

- ▶ 1-free regular expressions defined (also) with unary star
- ▶ image of 1-free regular expressions under the process interpretation  $P$  is **not** closed under bisimulation collapse

# Summary and outlook

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

# Summary and outlook

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

# Summary and outlook

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

# Summary and outlook

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

Outlook on an extension:

- ▶ image of 1-free reg. expr's under  $P^\bullet =$  finite process graphs with LEE.

# Summary and outlook

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

Outlook on an extension:

- ▶ image of 1-free reg. expr's under  $P^\bullet =$  finite process graphs with LEE.

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

# Summary and outlook

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

Outlook on an extension:

- ▶ image of 1-free reg. expr's under  $P^\bullet =$  finite process graphs with LEE.

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