Expressing safe communication protocols

Luc Edixhoven

26 January 2021

Who am I?

Luc Edixhoven, internal PhD student,
under supervision of Sung-Shik Jongmans

- France (Rennes), 1995 – 2002
- Netherlands (Leiden), 2002 – now
- Computer Science (Leiden), 2013 – 2019
- Open University (CWI), 2019 – 2023?

What are my research interests?

- In a word: puzzles
- Formal methods (formal languages, automata theory, formal logic)
- Combinatorial game theory

- Bachelor thesis: SAT solvers and sliding puzzles
- Research project: CGT and 3-player Clobber
- Master thesis: BDDs, theoretical analysis and 2048

The remainder of this presentation

1. What are we studying?
2. What are we hoping to achieve?
3. How are we getting along?

Parallelisation

"The age of single-threading is over.
The time for multithreading has come."

Parallelisation

"With great power comes great responsibility."

Parallelisation

"With great power comes great responsibility."

$\downarrow$

"With parallelisation comes synchronisation."

Parallelisation

"With great power comes great responsibility."

↓

"With parallelisation comes synchronisation."

The classical way

(Image from Wikipedia)

Parallelisation

"With great power comes great responsibility."

↓

"With parallelisation comes synchronisation."

The classical way

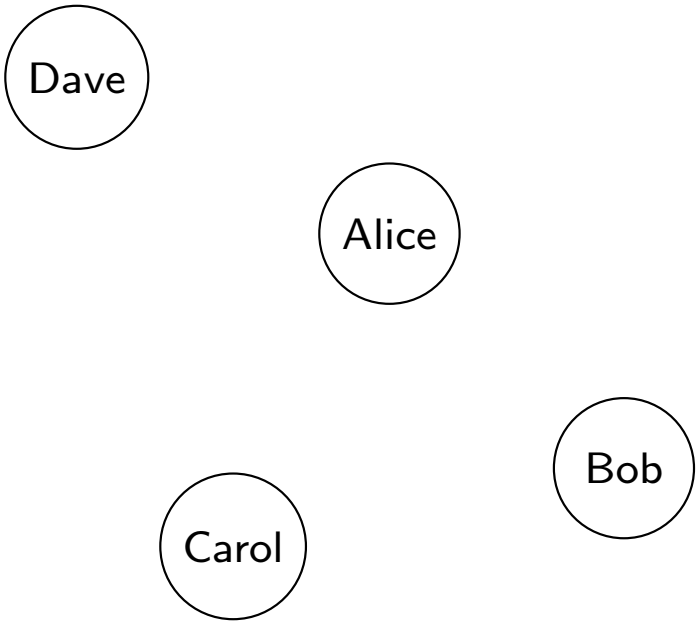
(Image from Wikipedia)

The modern way

(Image from PostNL)

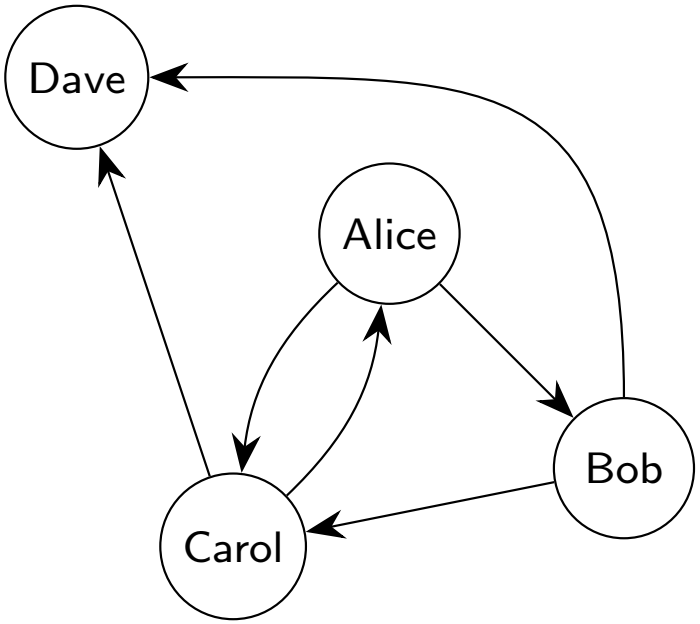
A message passing model

A message passing model



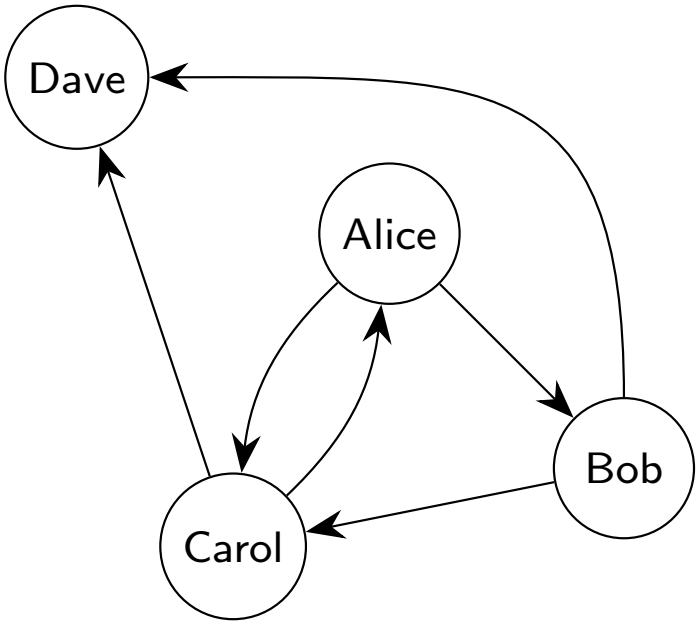
1. A set of participants (Alice, Bob, Carol, Dave, . . .)

A message passing model



1. A set of participants (Alice, Bob, Carol, Dave, . . .)
2. A set of one-directional channels between participants

A message passing model



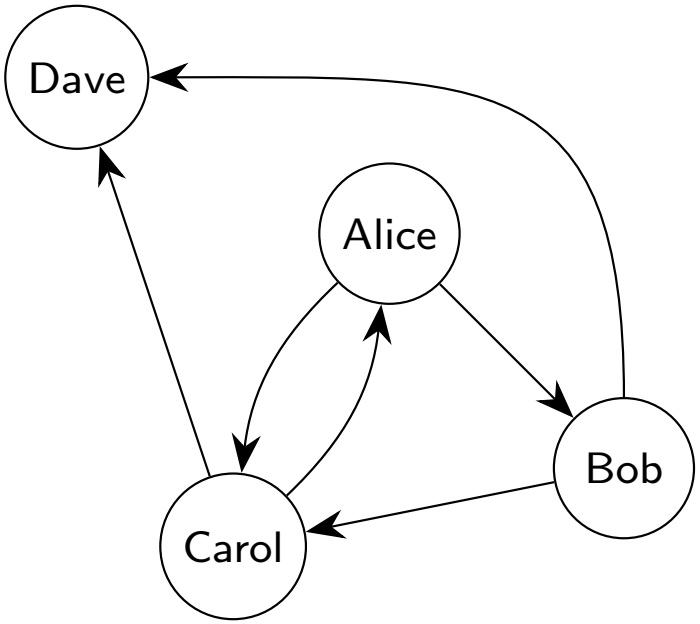
Alice sends a message to either Bob or Carol

If to Bob, then Bob sends a message to Carol and Dave

If to Carol, then Carol sends a message to Alice and Dave

1. A set of participants (Alice, Bob, Carol, Dave, . . .)
2. A set of one-directional channels between participants
3. A description of allowed/desired behaviour
   = communication protocol

A message passing model



Alice sends a message to either Bob or Carol

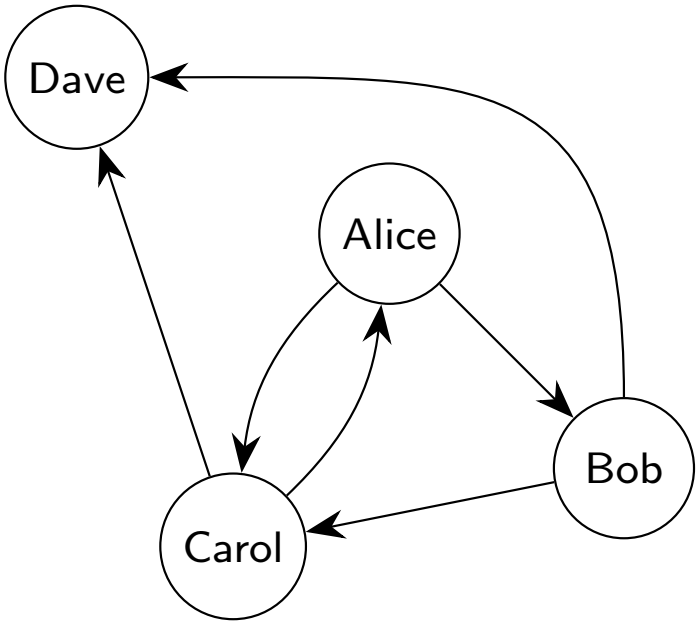
If to Bob, then Bob sends a message to Carol and Dave

If to Carol, then Carol sends a message to Alice and Dave

Does a given system implement this protocol's behaviour?

2. A set of one-directional channels between participants
3. A description of allowed/desired behaviour
   = communication protocol

A message passing model



Alice sends a message to either Bob or Carol

If to Bob, then Bob sends a message to Carol and Dave

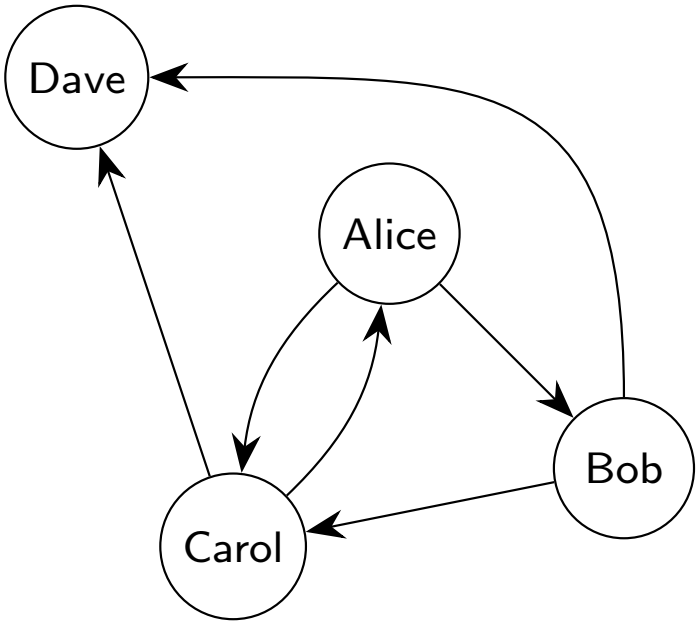
If to Carol, then Carol sends a message to Alice and Dave

Does a given system implement this protocol's behaviour?

Does *any* system implement this protocol's behaviour?

3. A description of allowed/desired behaviour
   = communication protocol

A message passing model



Alice sends a message to either Bob or Carol

If to Bob, then Bob sends a message to Carol and Dave

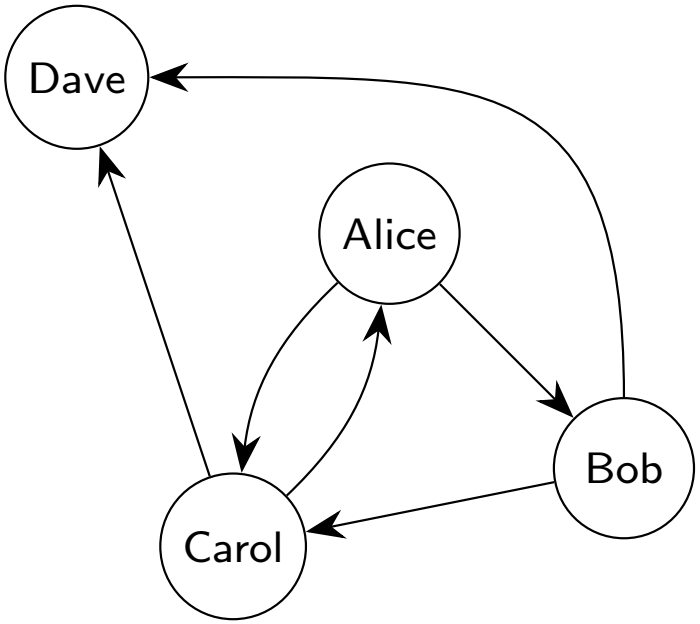
If to Carol, then Carol sends a message to Alice and Dave

Does a given system implement this protocol's behaviour?

Does *any* system implement this protocol's behaviour?

Is this protocol safe to begin with?

A message passing model



Alice sends a message to either Bob or Carol

If to Bob, then Bob sends a message to Carol and Dave

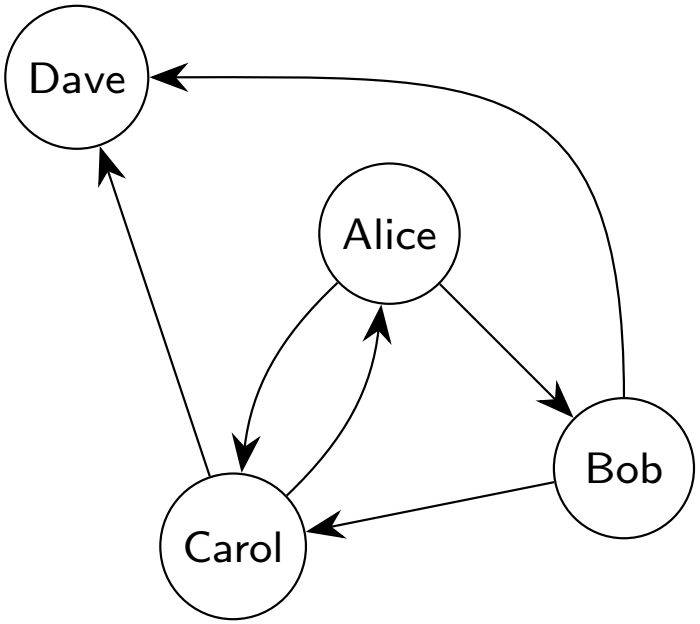
If to Carol, then Carol sends a message to Alice and Dave

Does a given system implement this protocol's behaviour?

Does *any* system implement this protocol's behaviour?

Is this protocol safe to begin with?

A message passing model



Alice sends a message to either Bob or Carol

If to Bob, then Bob sends a message to Carol and Dave

If to Carol, then Carol sends a message to Alice and Dave

Does a given system implement this protocol's behaviour?

Does *any* system implement this protocol's behaviour?

Is this protocol safe to begin with?

What is safety?

Safety

What is sent must be received

Safety

What is sent must be received
What is received must have been sent

The problem

Expressiveness

The problem

Expressiveness

Can we do better?

Safety in formal languages

Formal languages over send and receive actions: $x!, x?, y!, y?, \ldots$

Safety in formal languages

Formal languages over send and receive actions: $x!, x?, y!, y?, \ldots$

Safety is similar to the Dyck language of balanced brackets

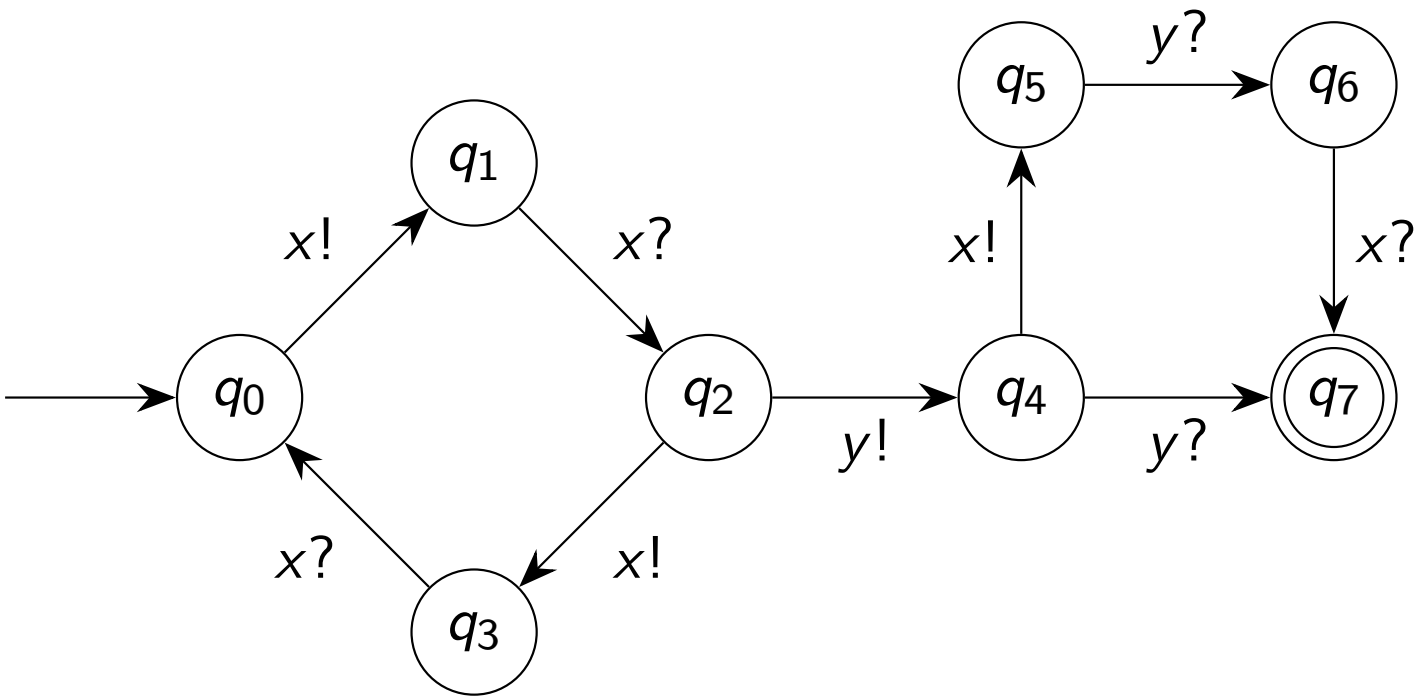
Safety in formal languages

Formal languages over send and receive actions: $x!, x?, y!, y?, \ldots$

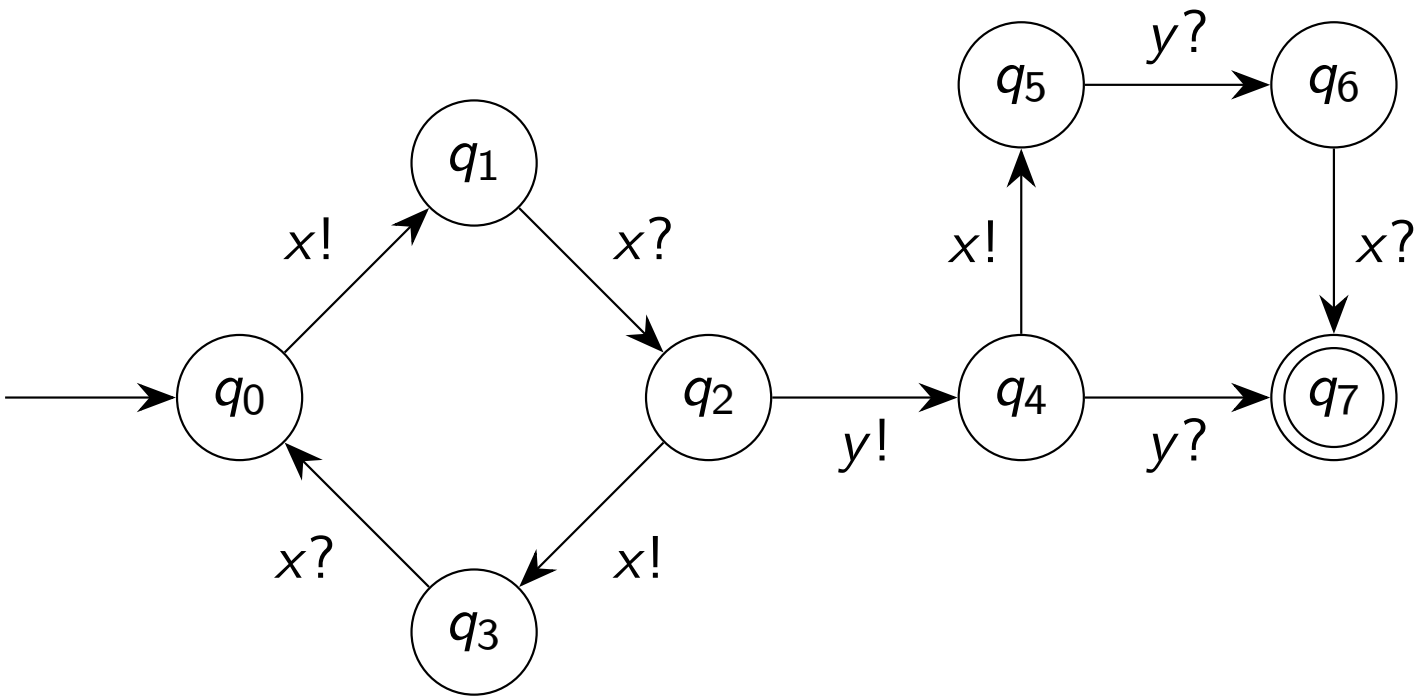
Safety is similar to the Dyck language of balanced brackets

Focus on regular languages

Safety in finite automata

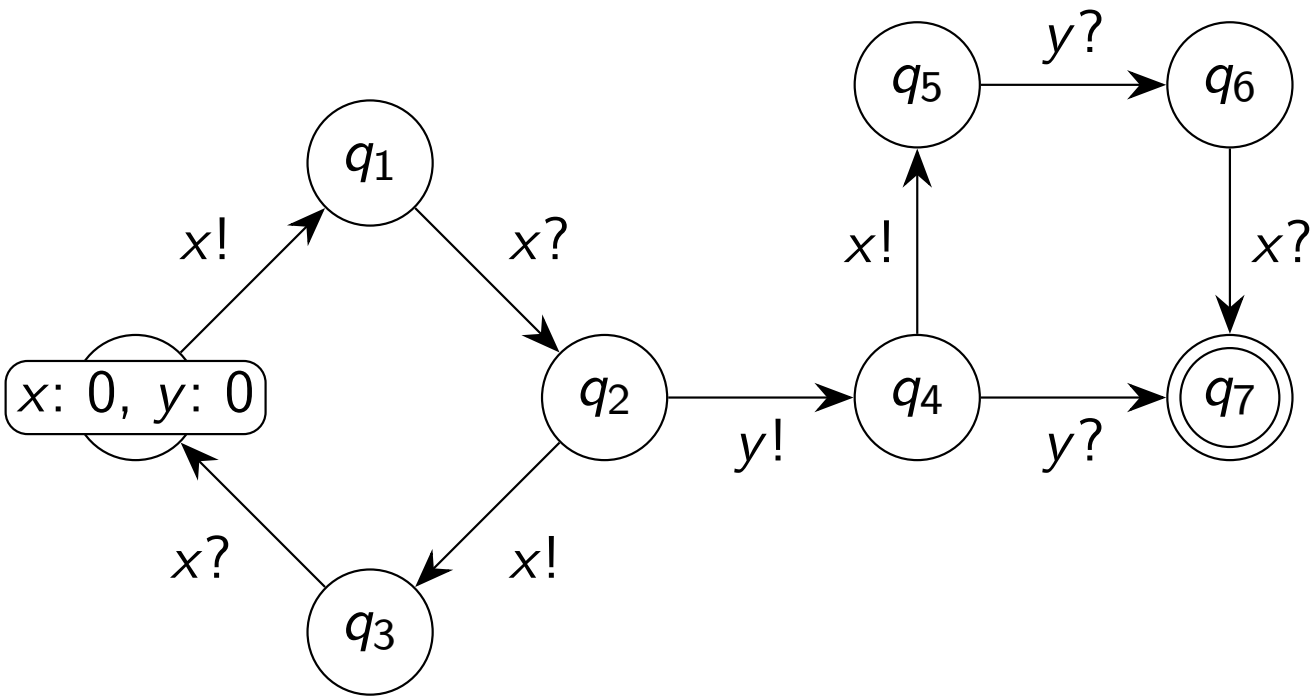


Safety in finite automata



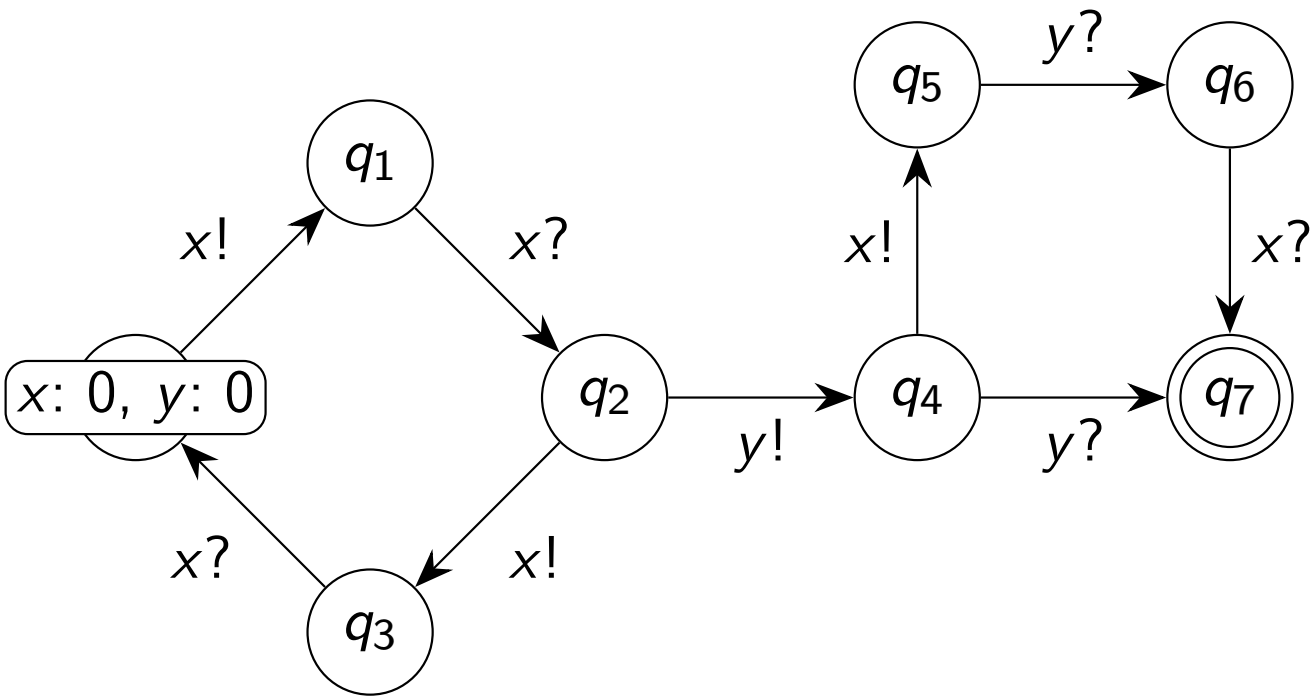
1. All channels should start out empty

Safety in finite automata



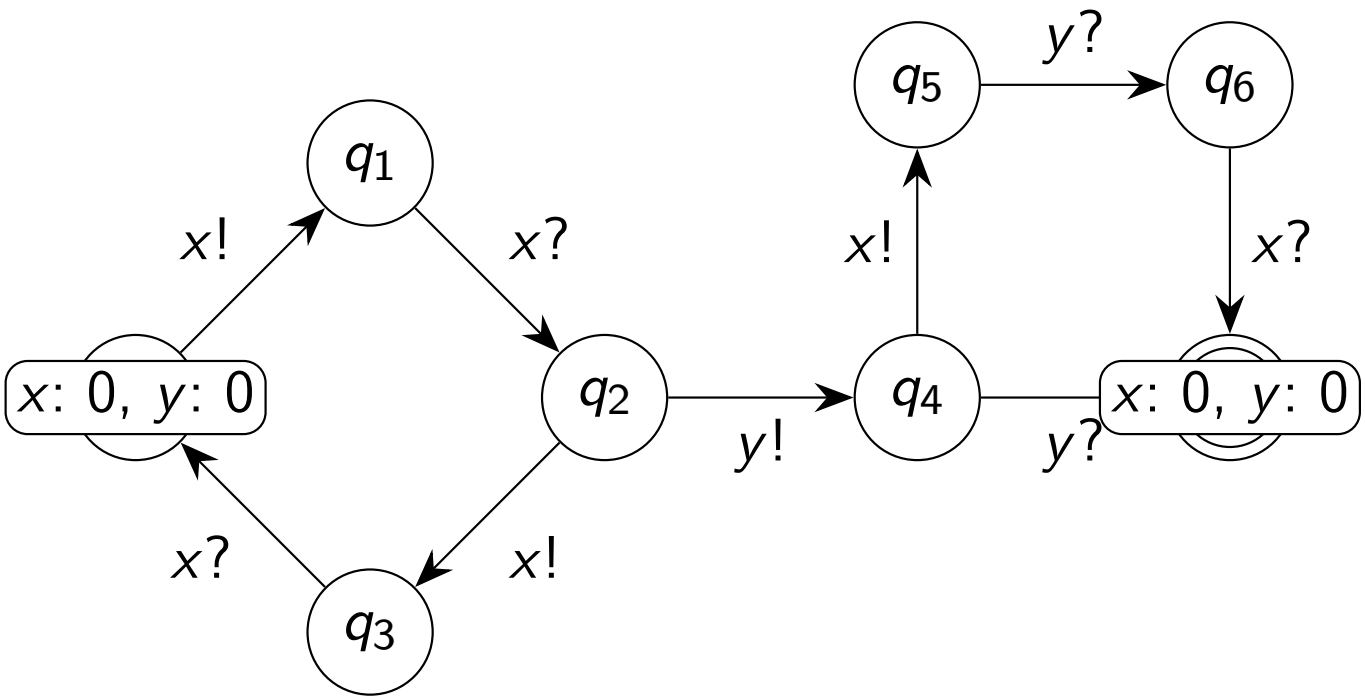
1. All channels should start out empty

Safety in finite automata



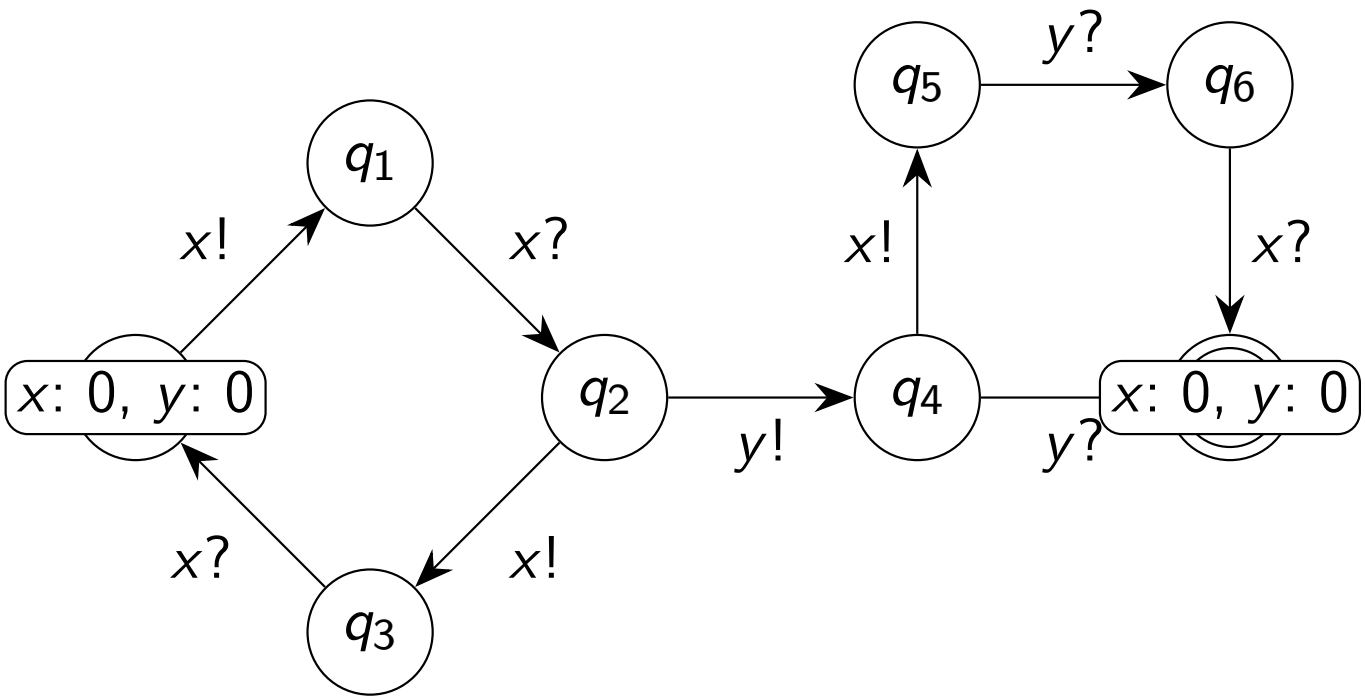
1. All channels should start out empty and end up empty

Safety in finite automata



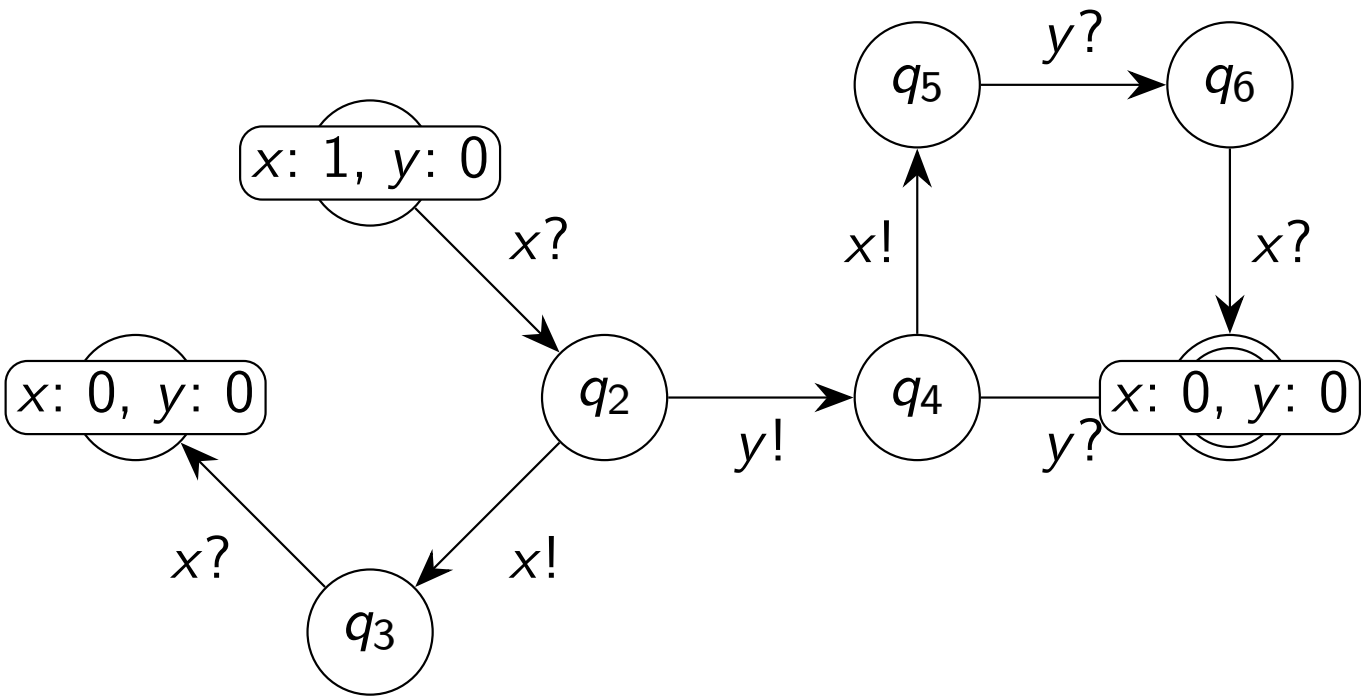
1. All channels should start out empty and end up empty

Safety in finite automata



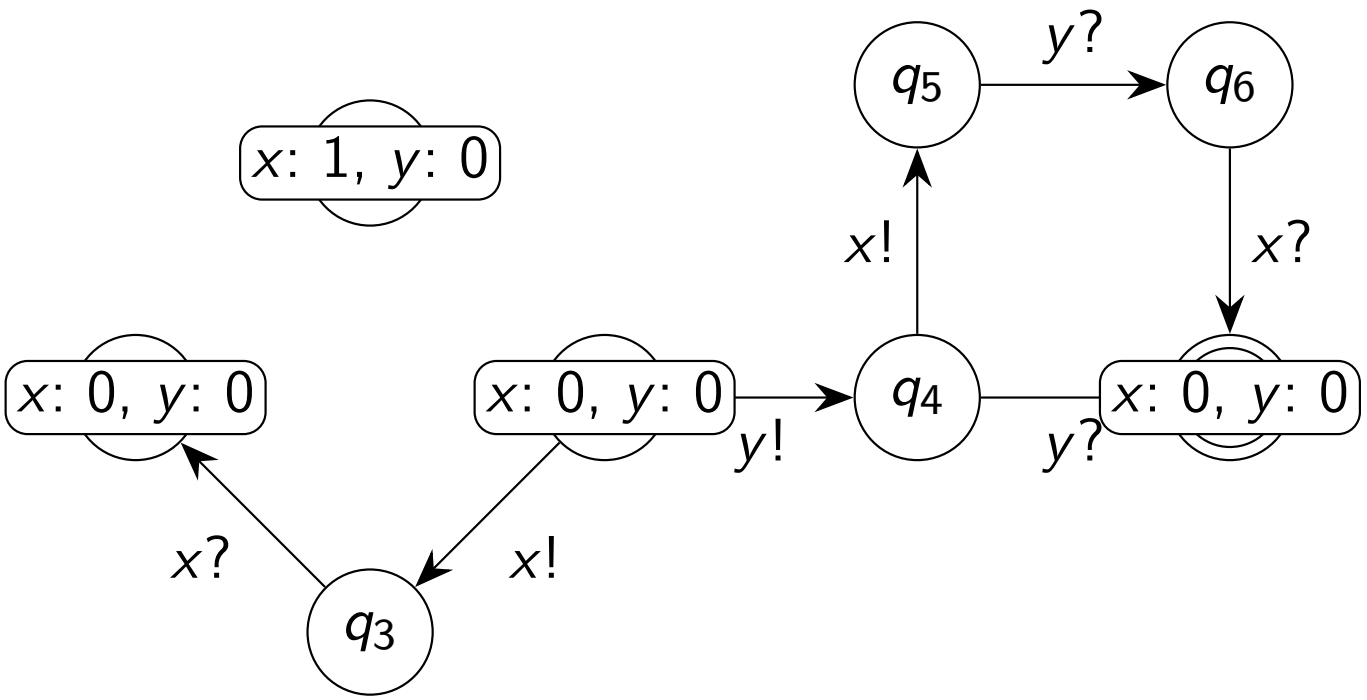
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



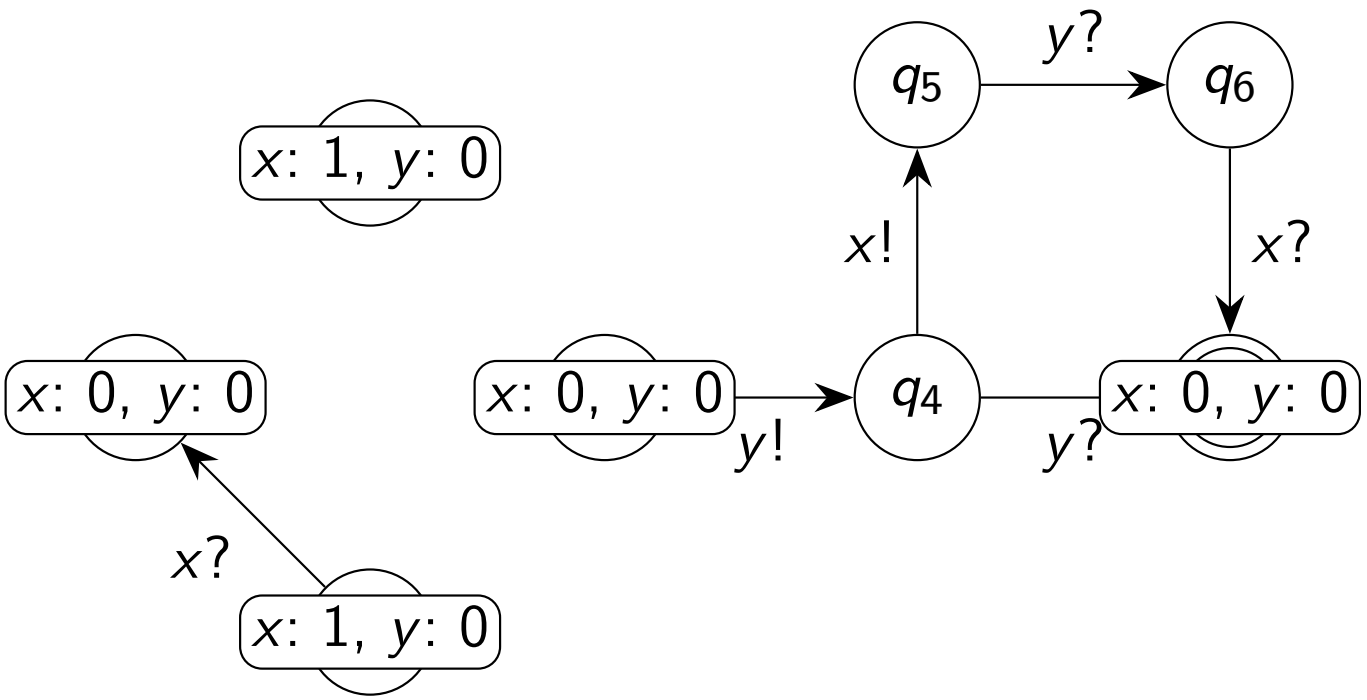
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



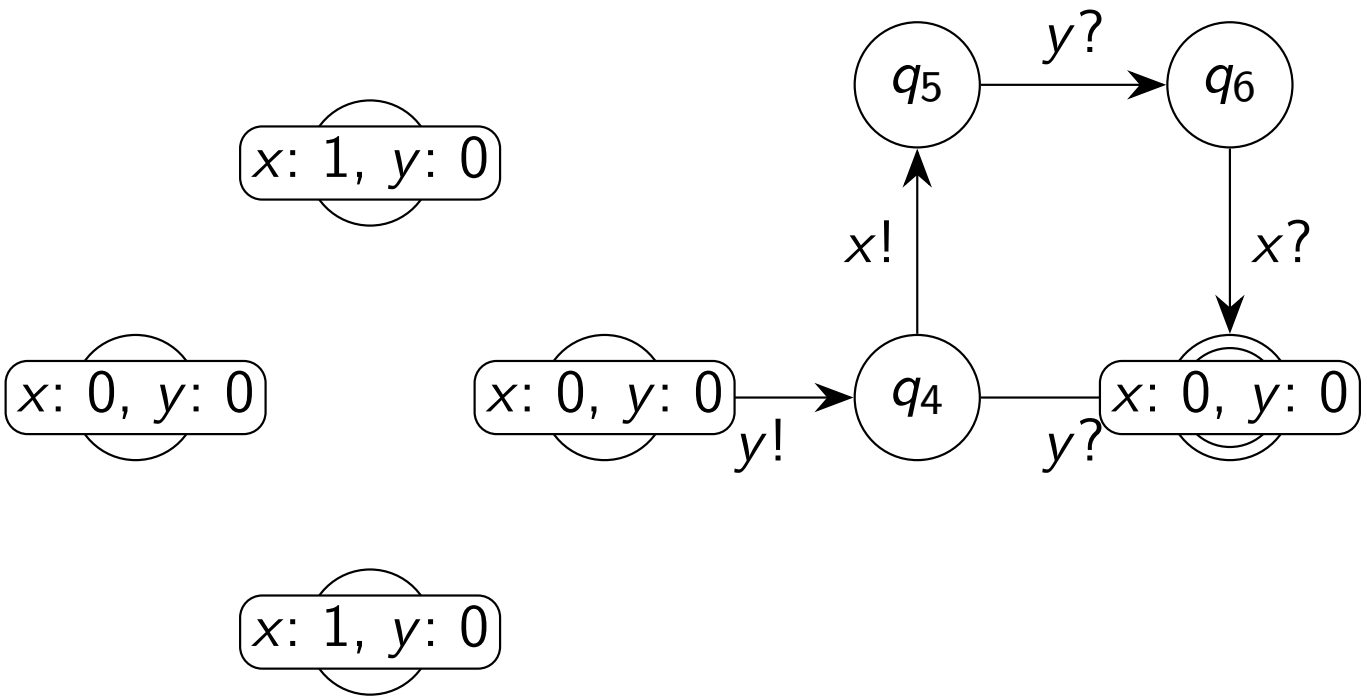
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



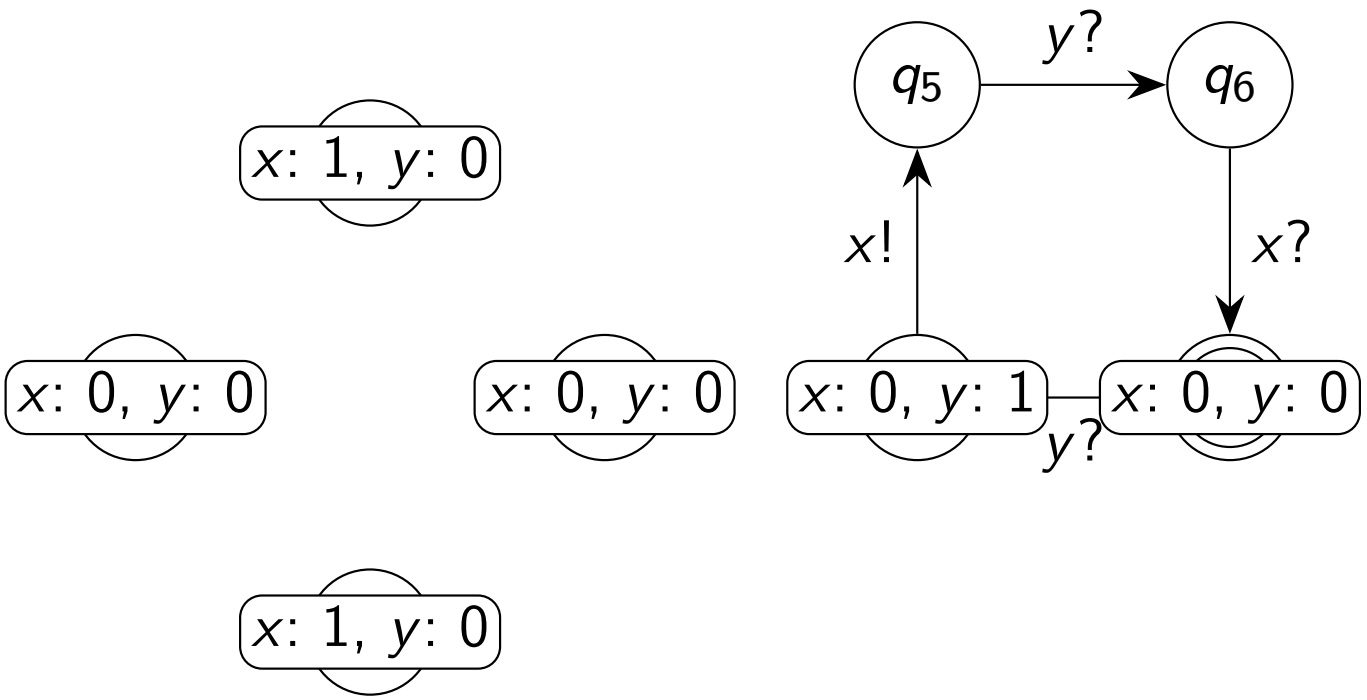
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



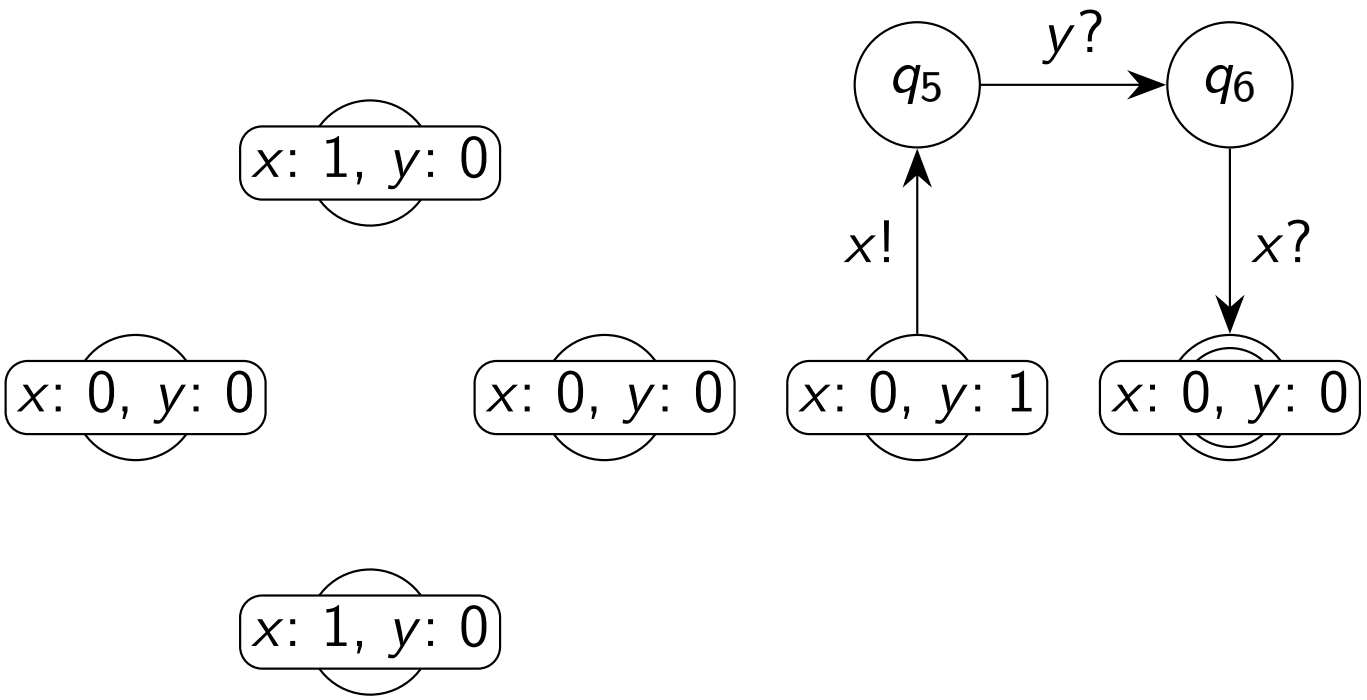
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



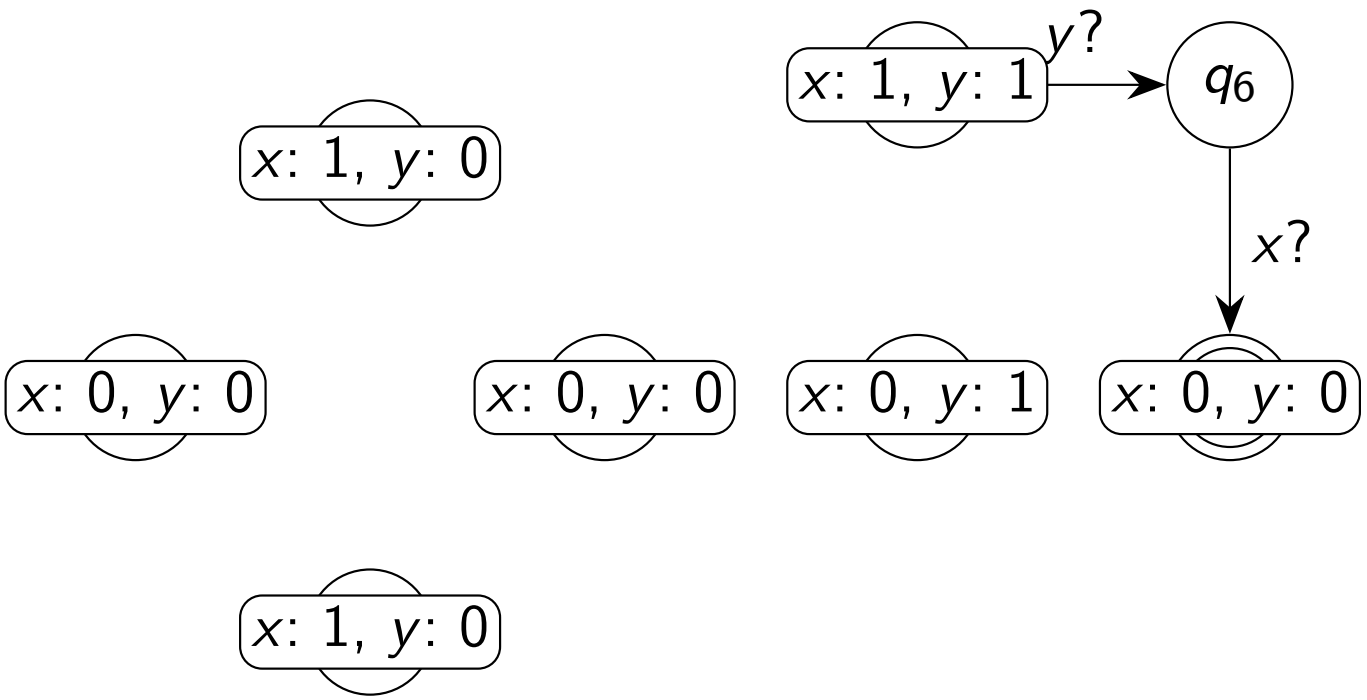
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



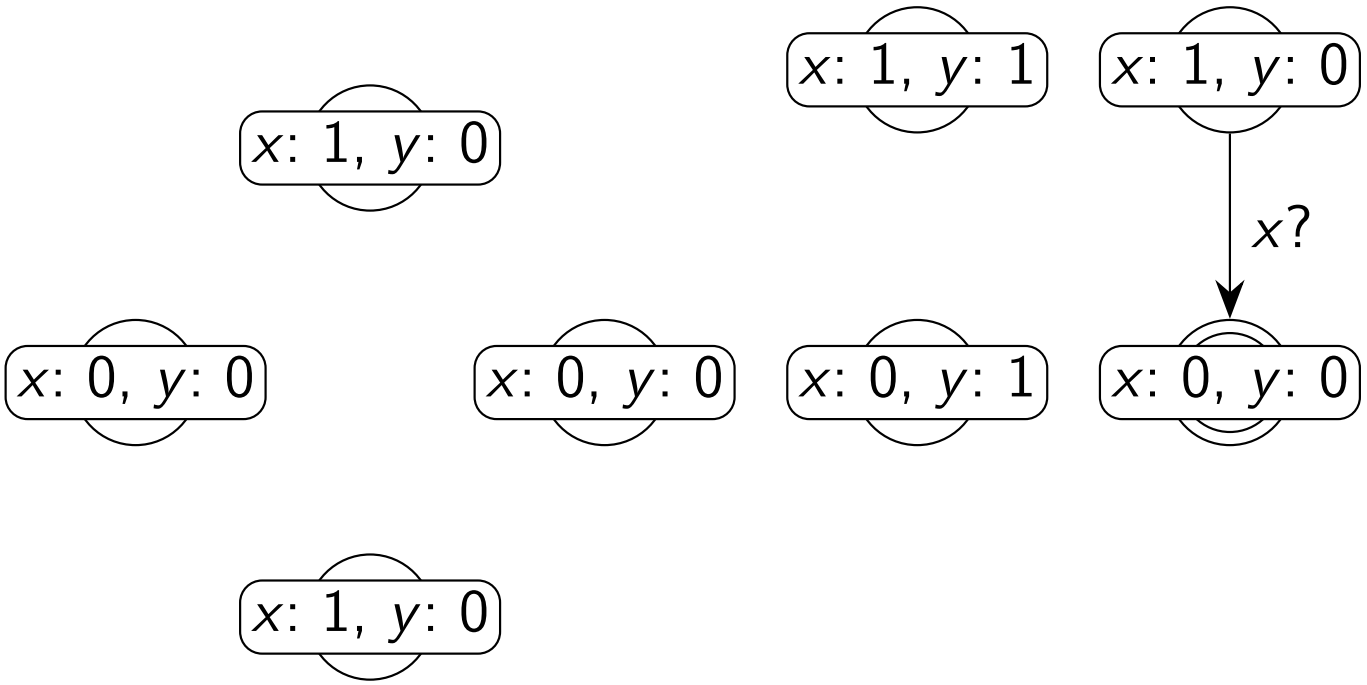
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



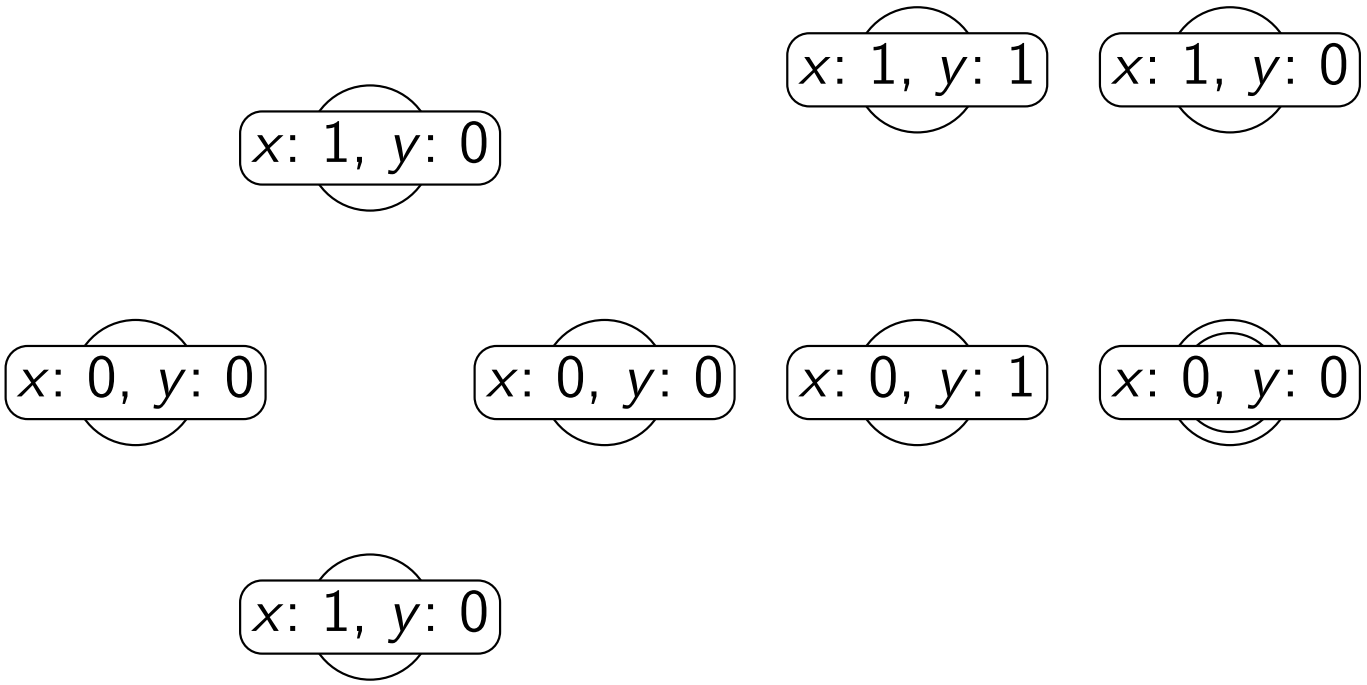
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



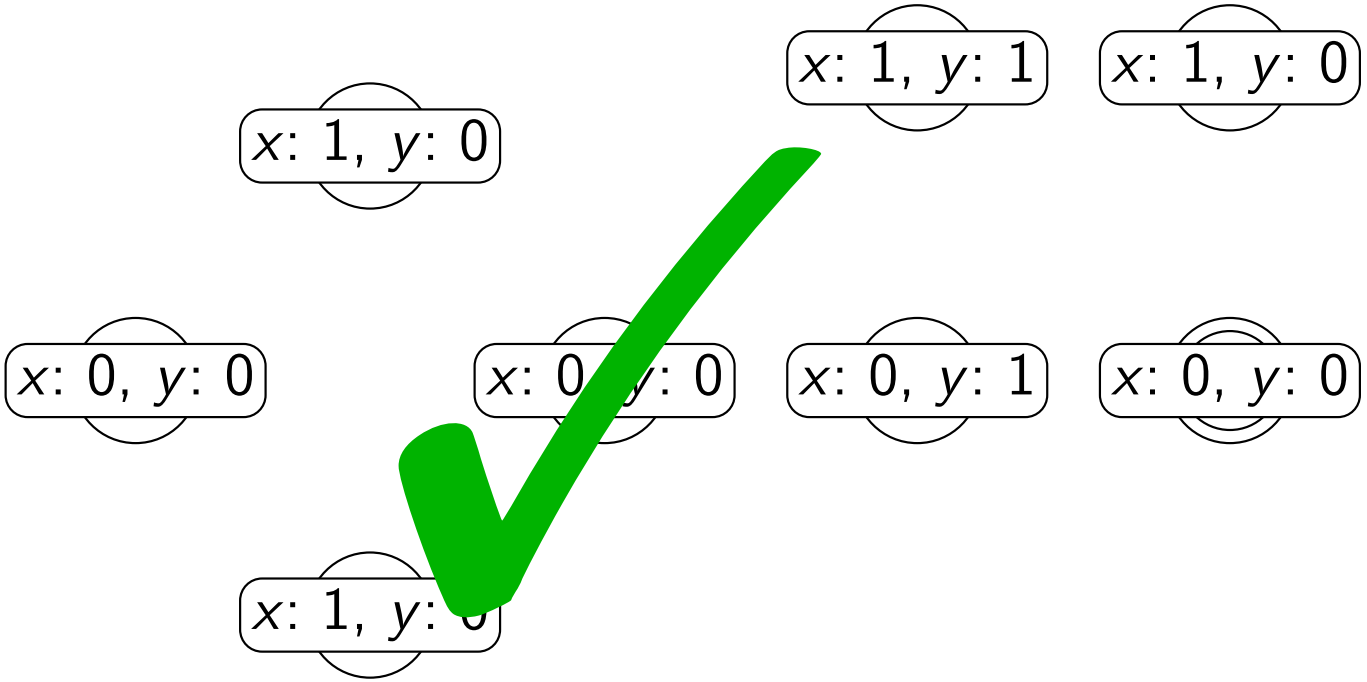
1. All channels should start out empty and end up empty
2. Everything in between should be consistent

Safety in finite automata



1. All channels should start out empty and end up empty
2. Everything in between should be consistent
3. No balance should ever be negative

Safety in finite automata



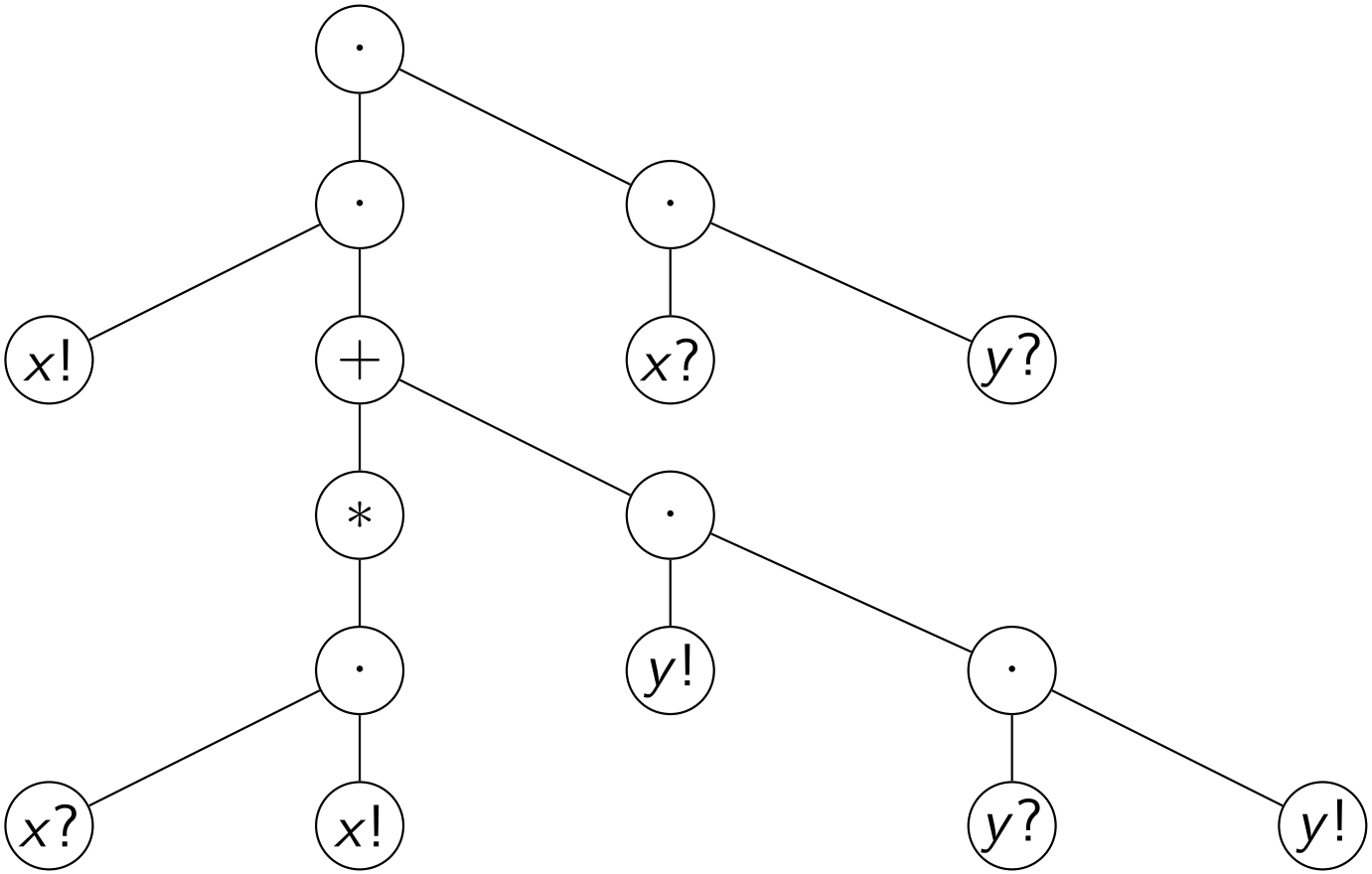
1. All channels should start out empty and end up empty
2. Everything in between should be consistent
3. No balance should ever be negative

Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$

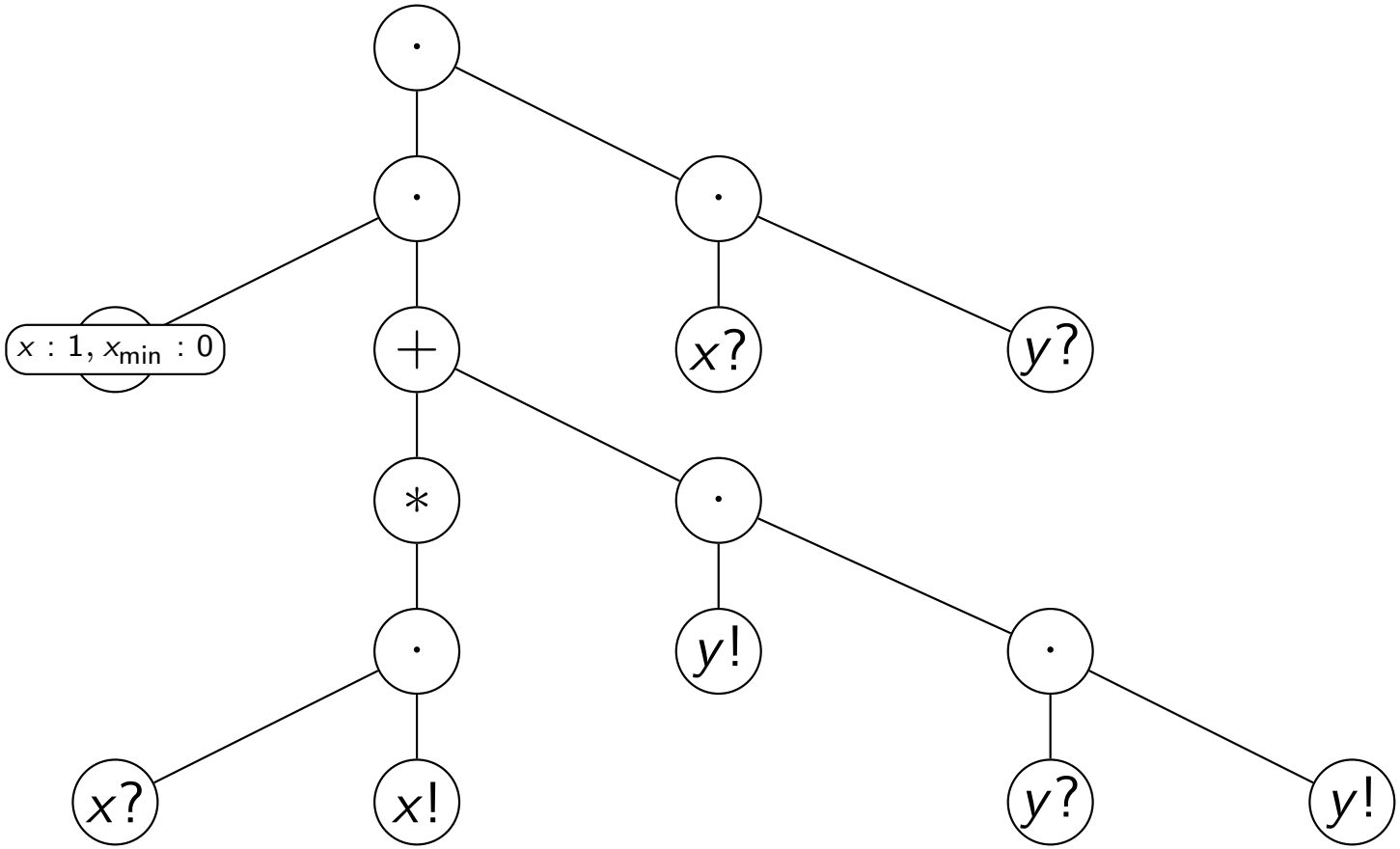
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



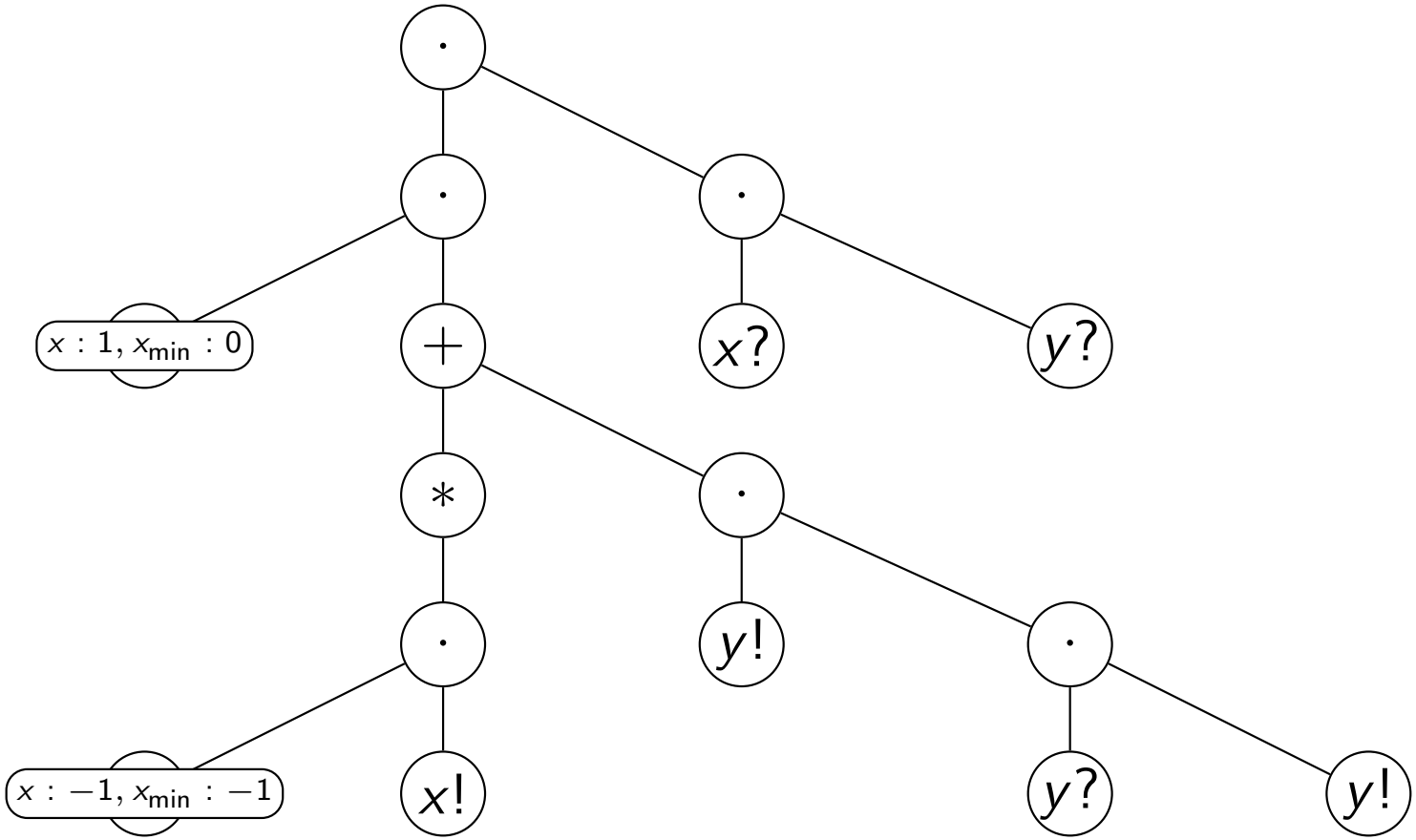
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



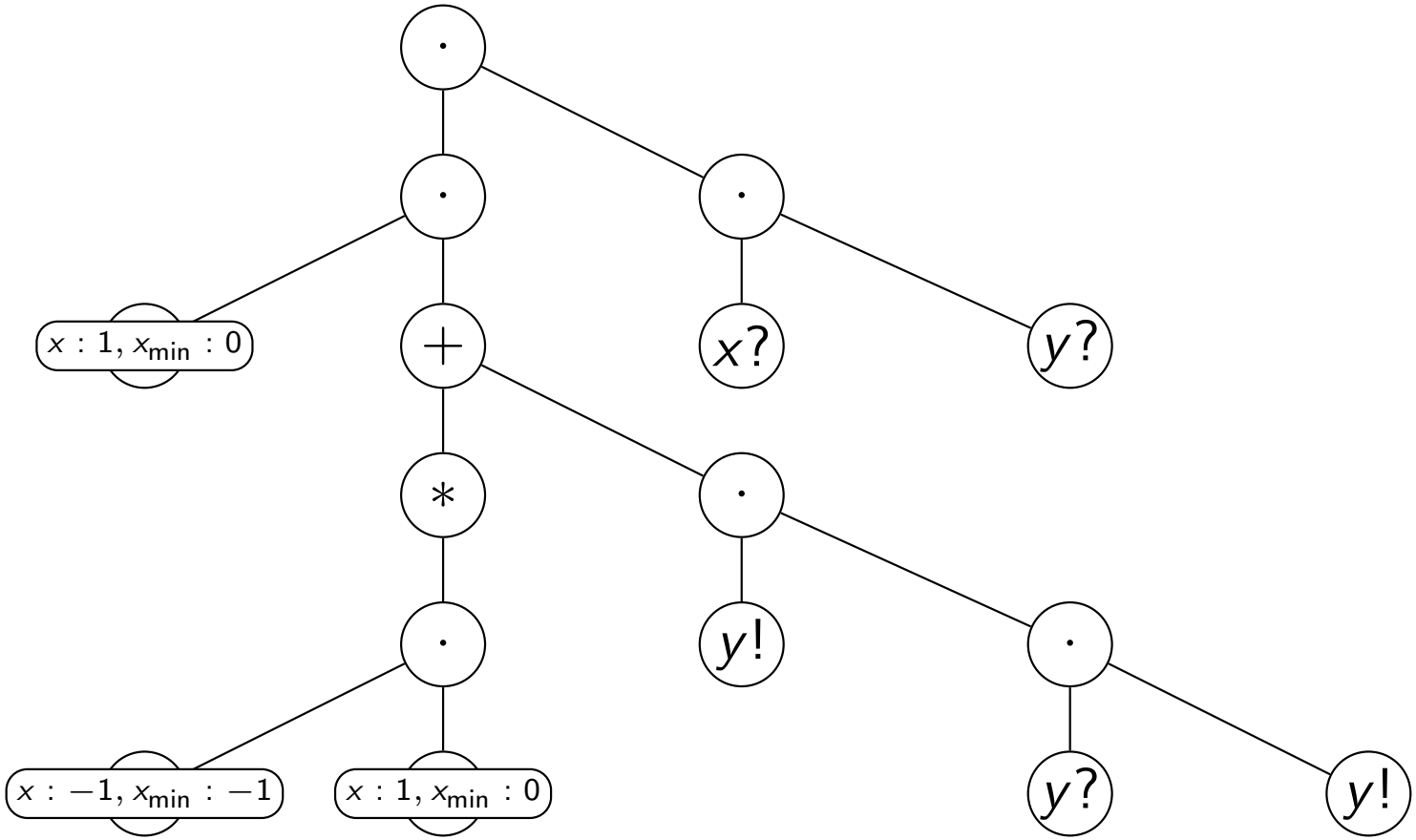
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



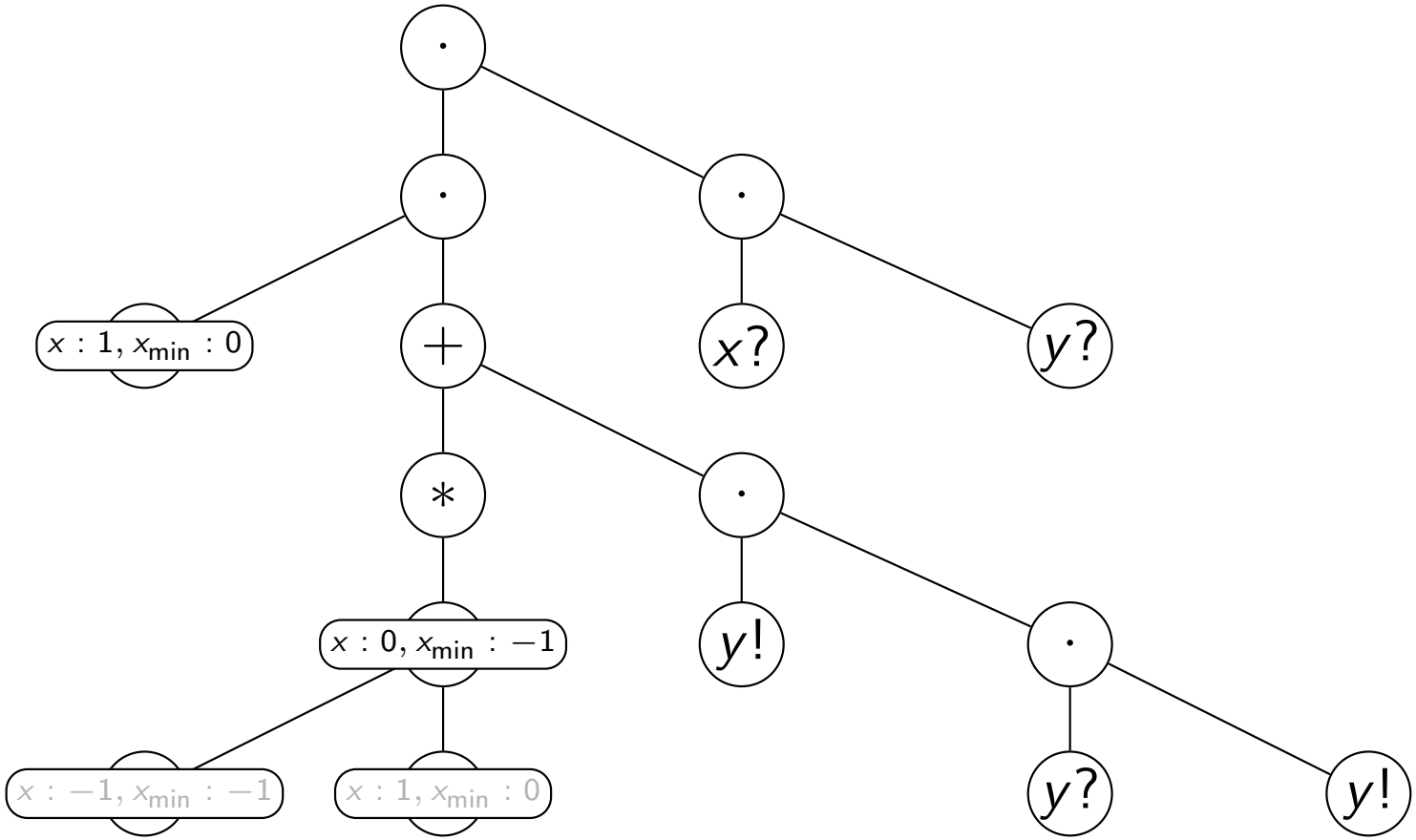
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



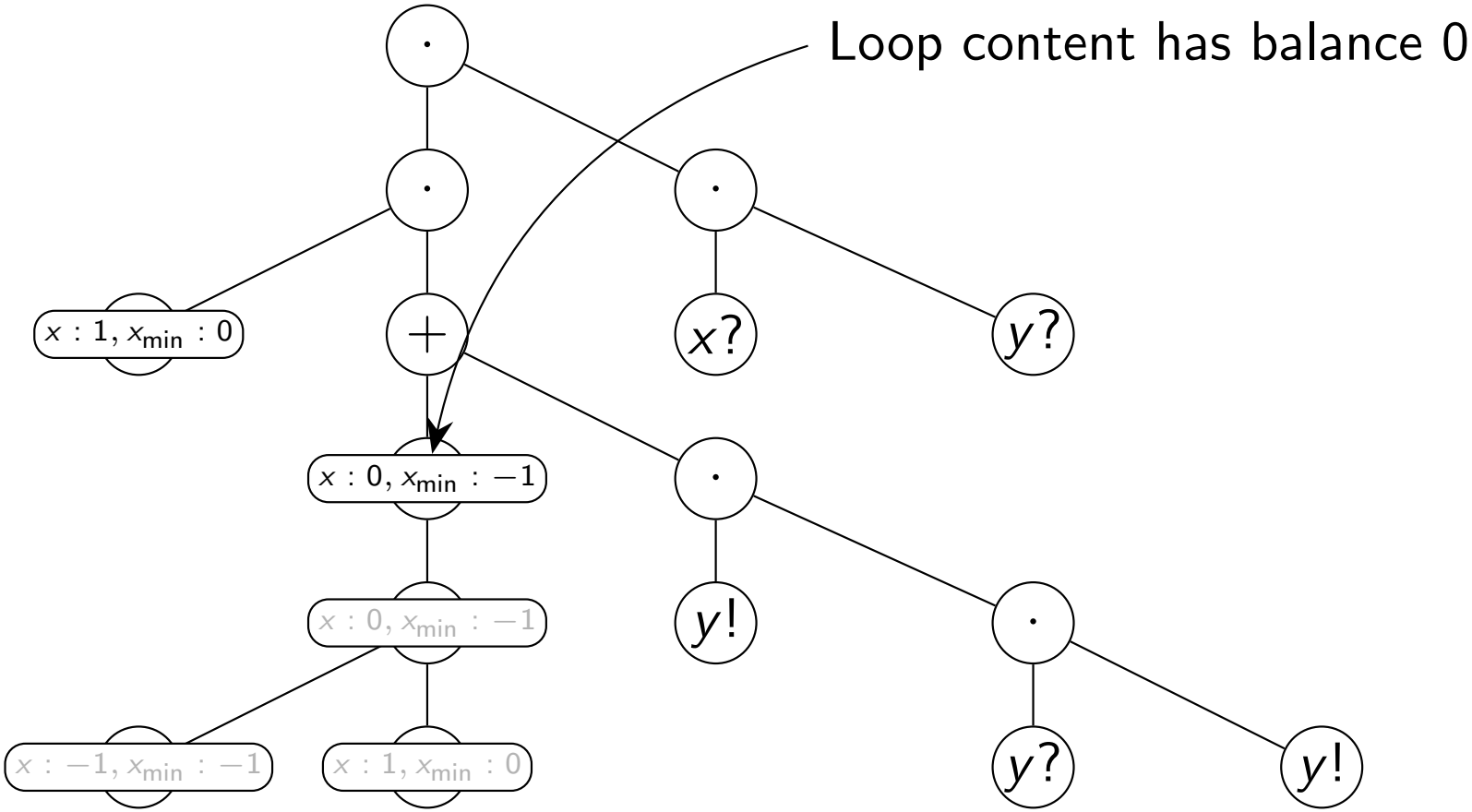
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



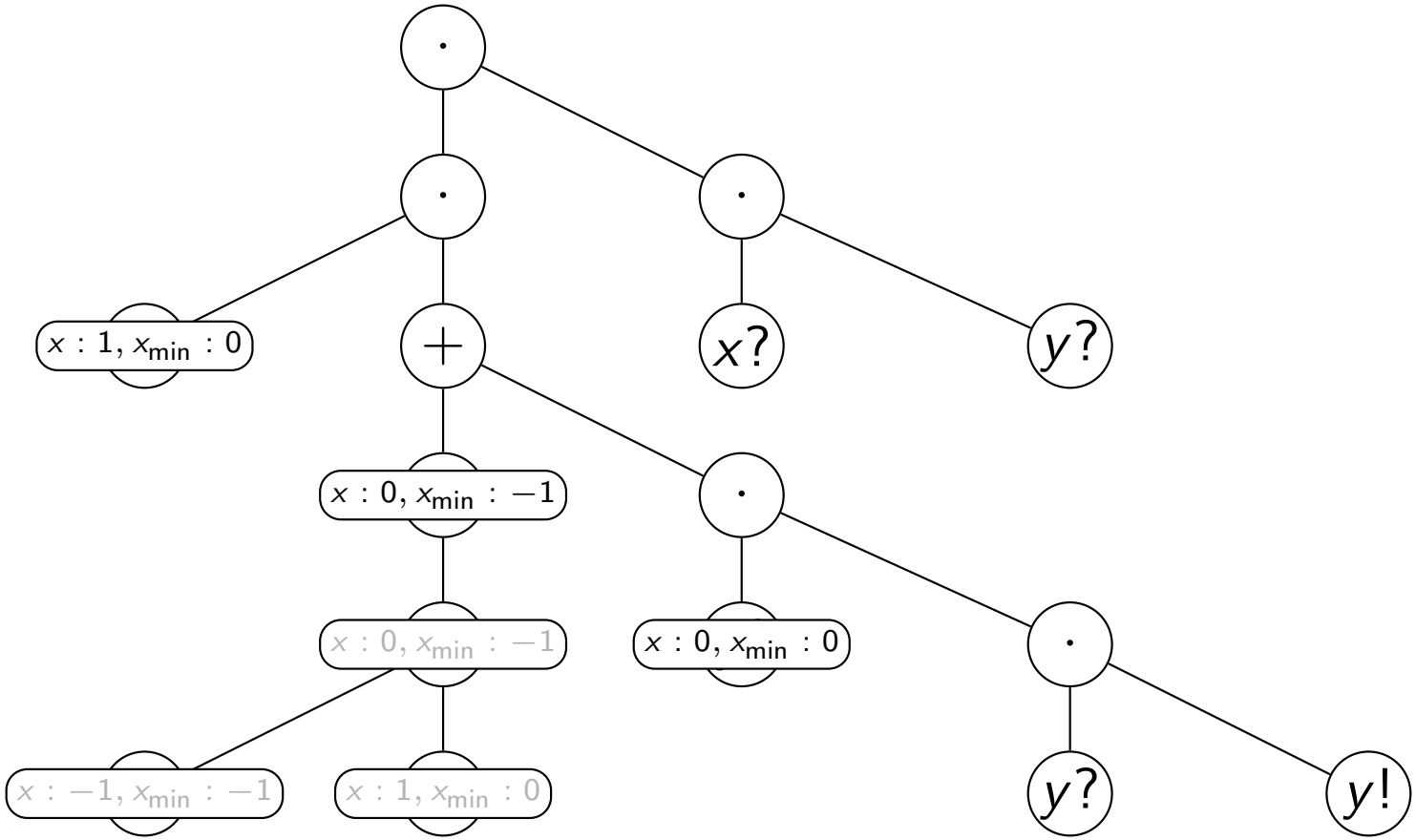
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



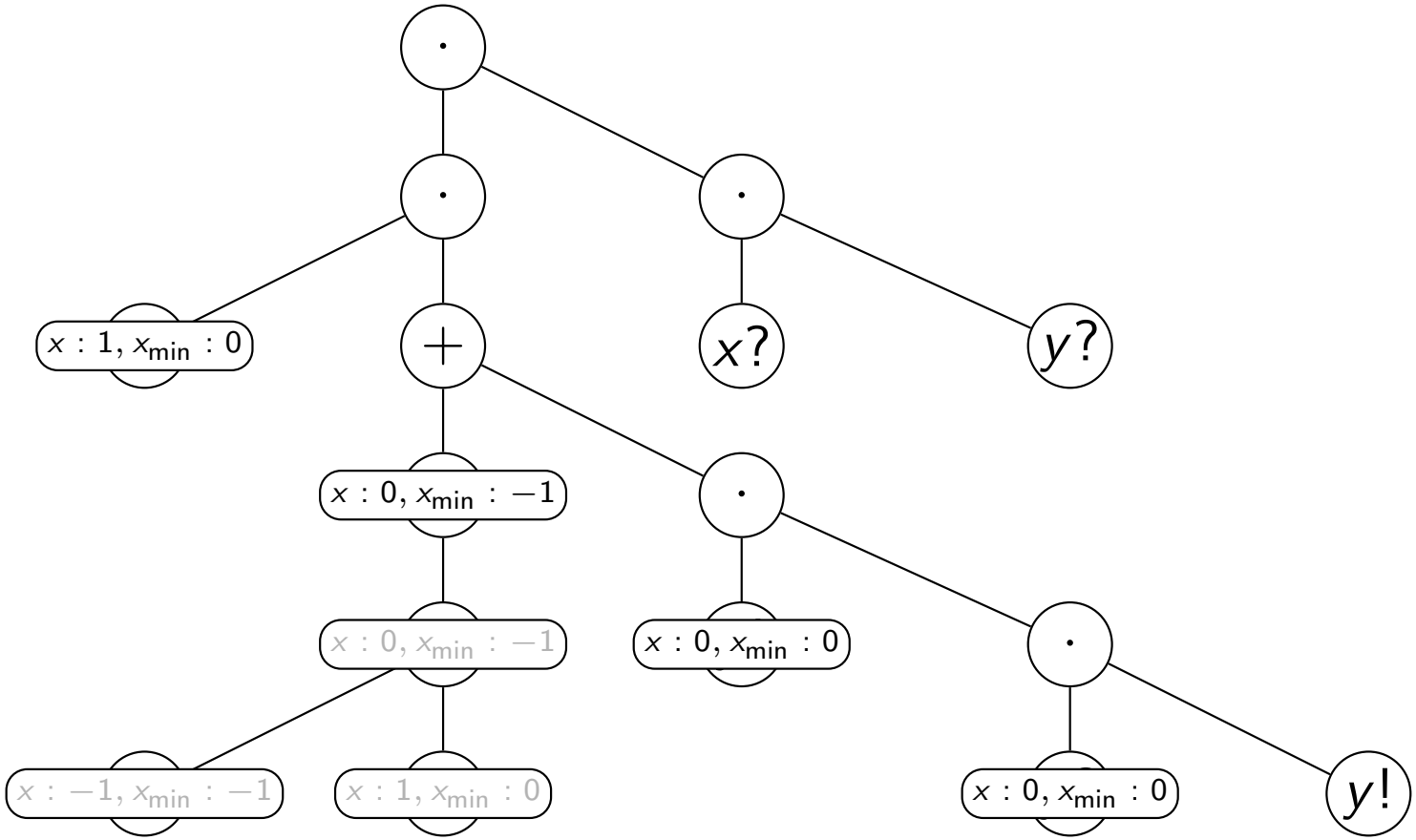
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



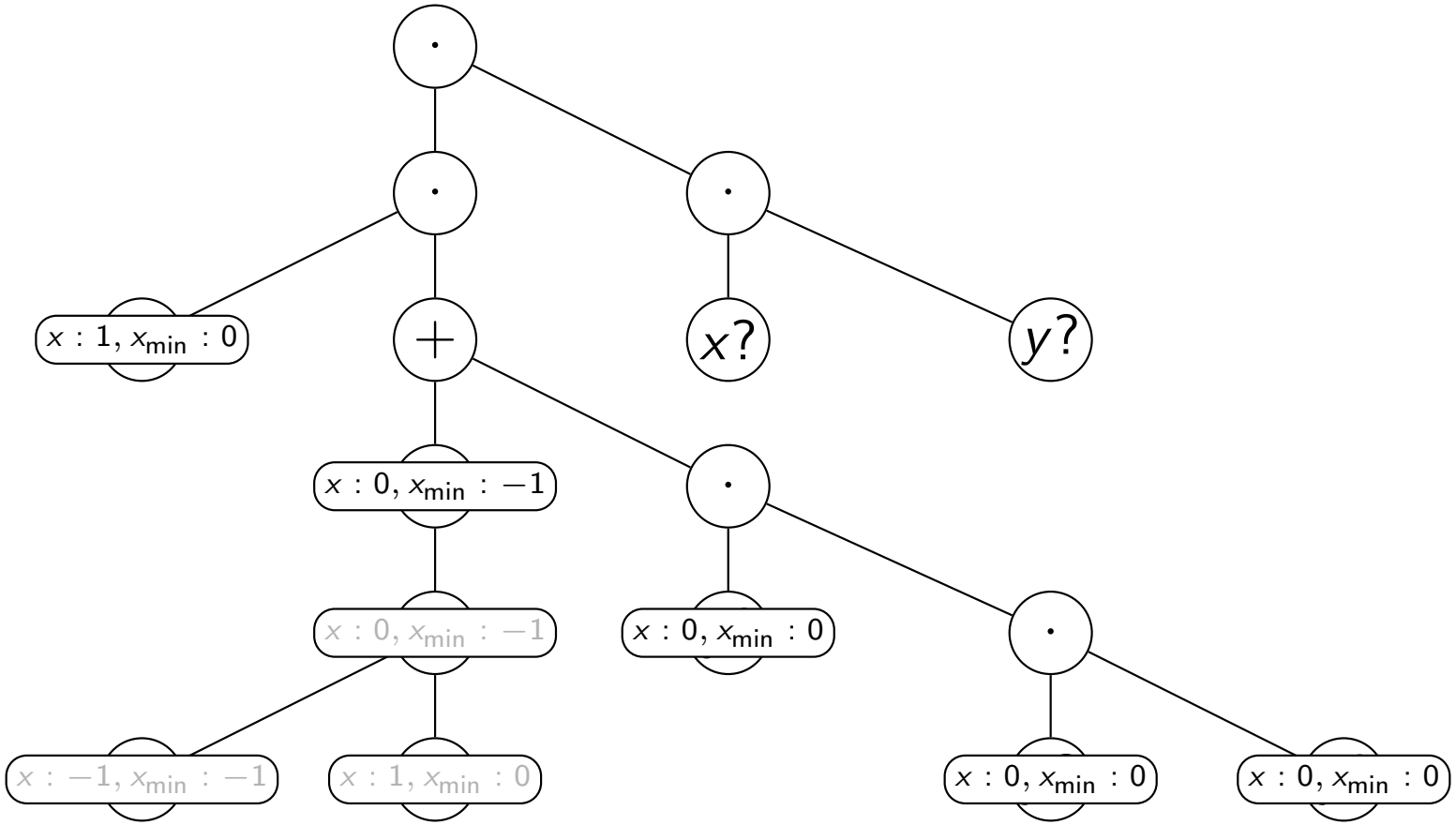
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



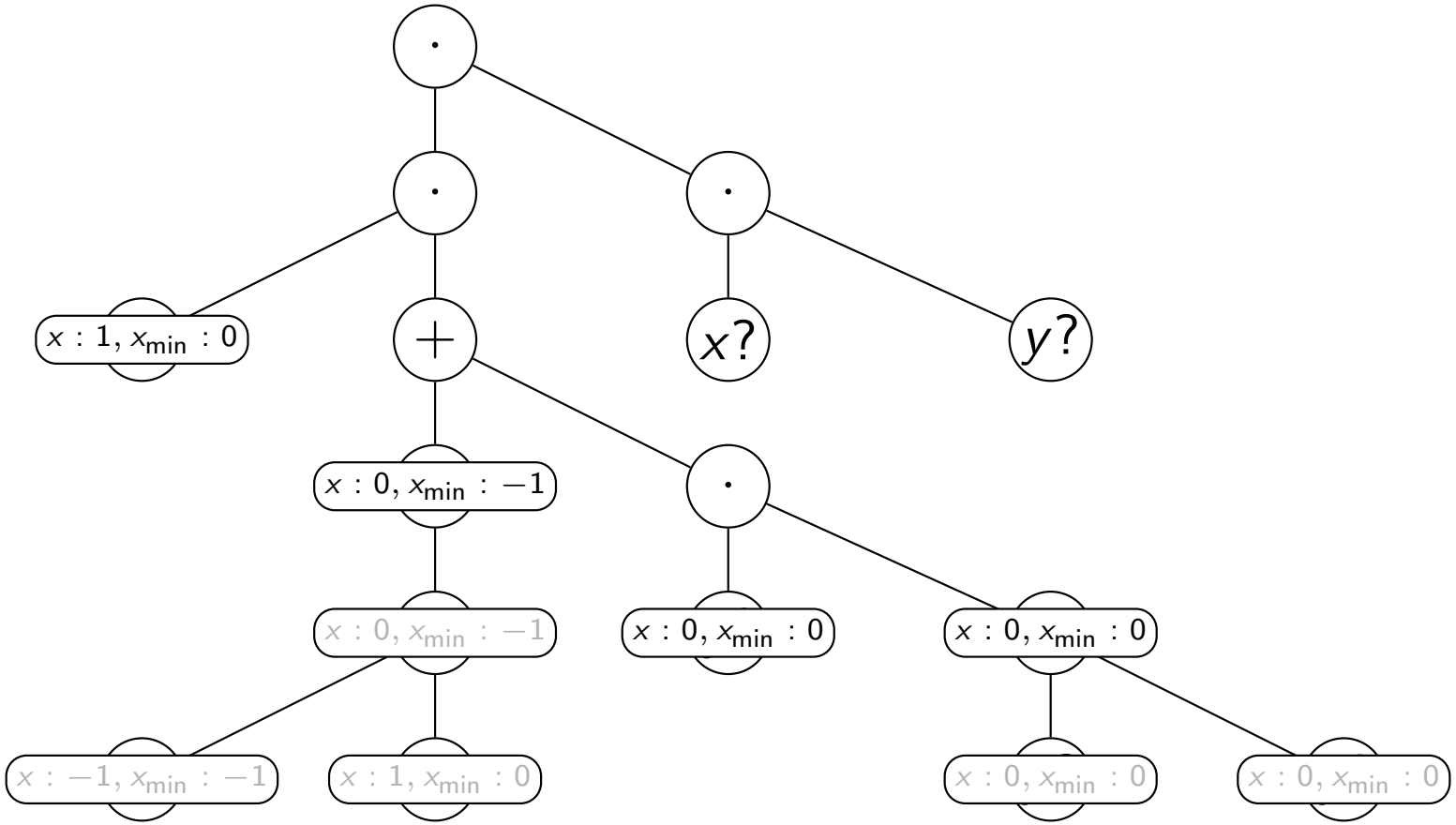
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



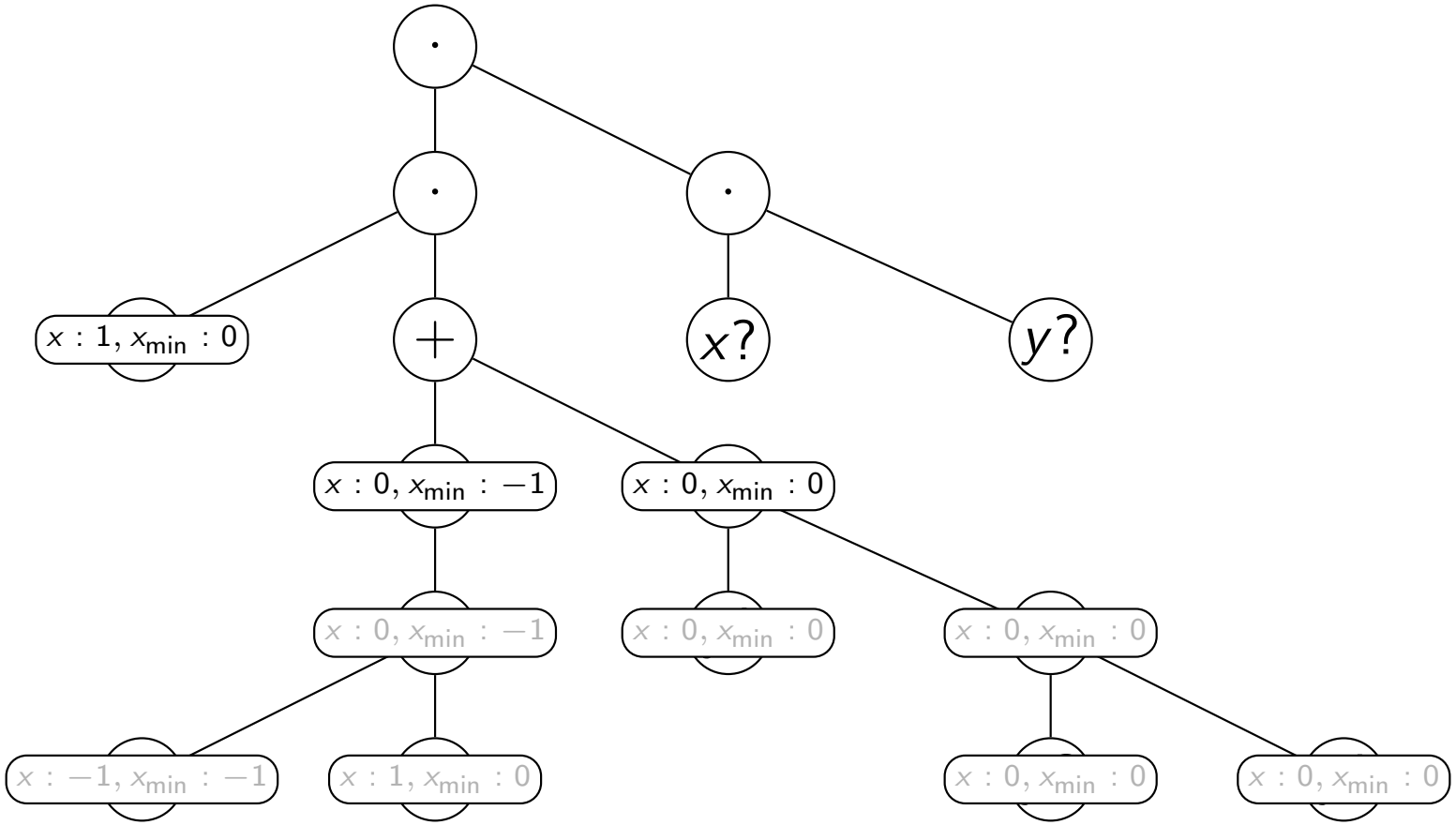
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



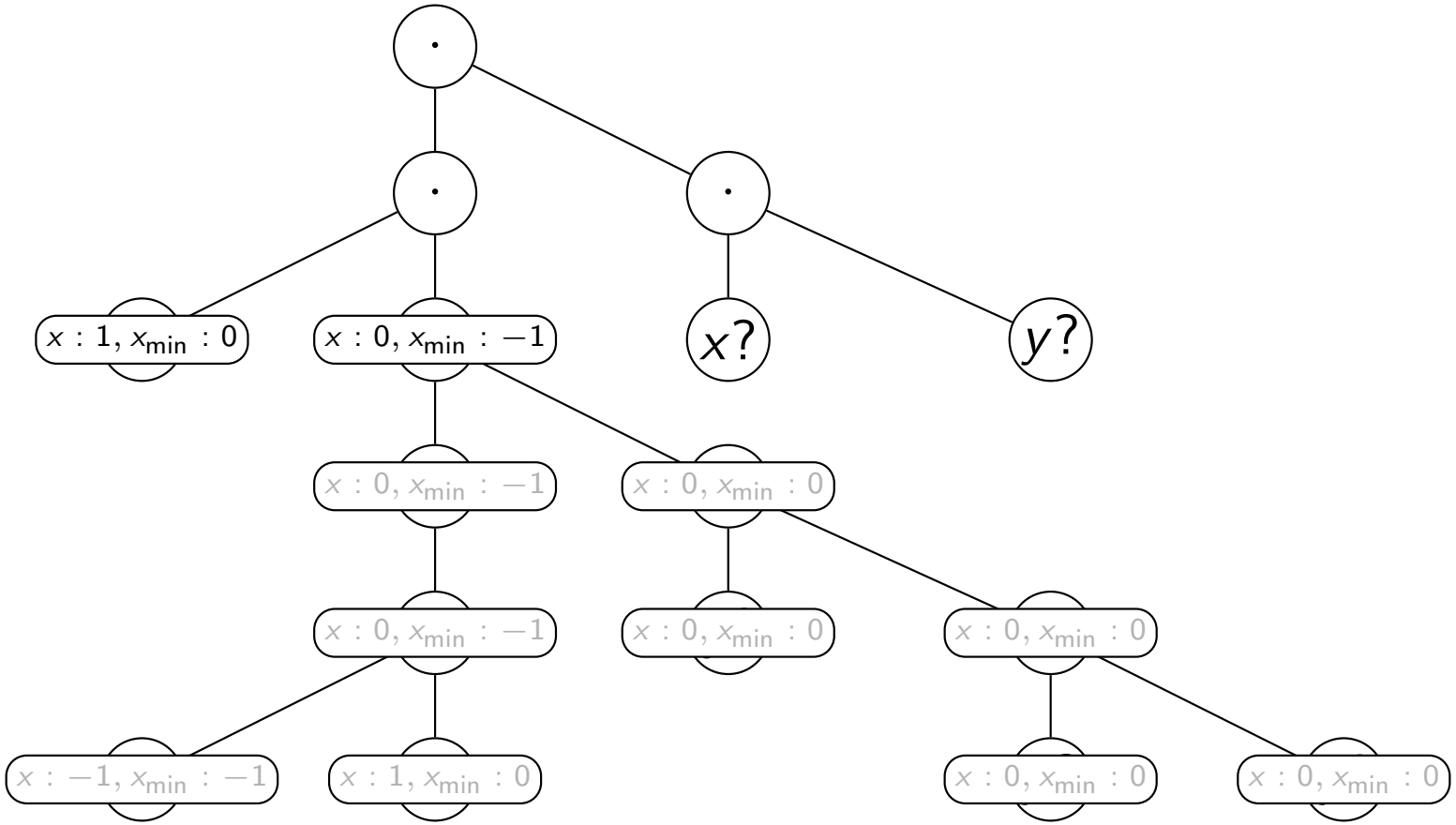
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



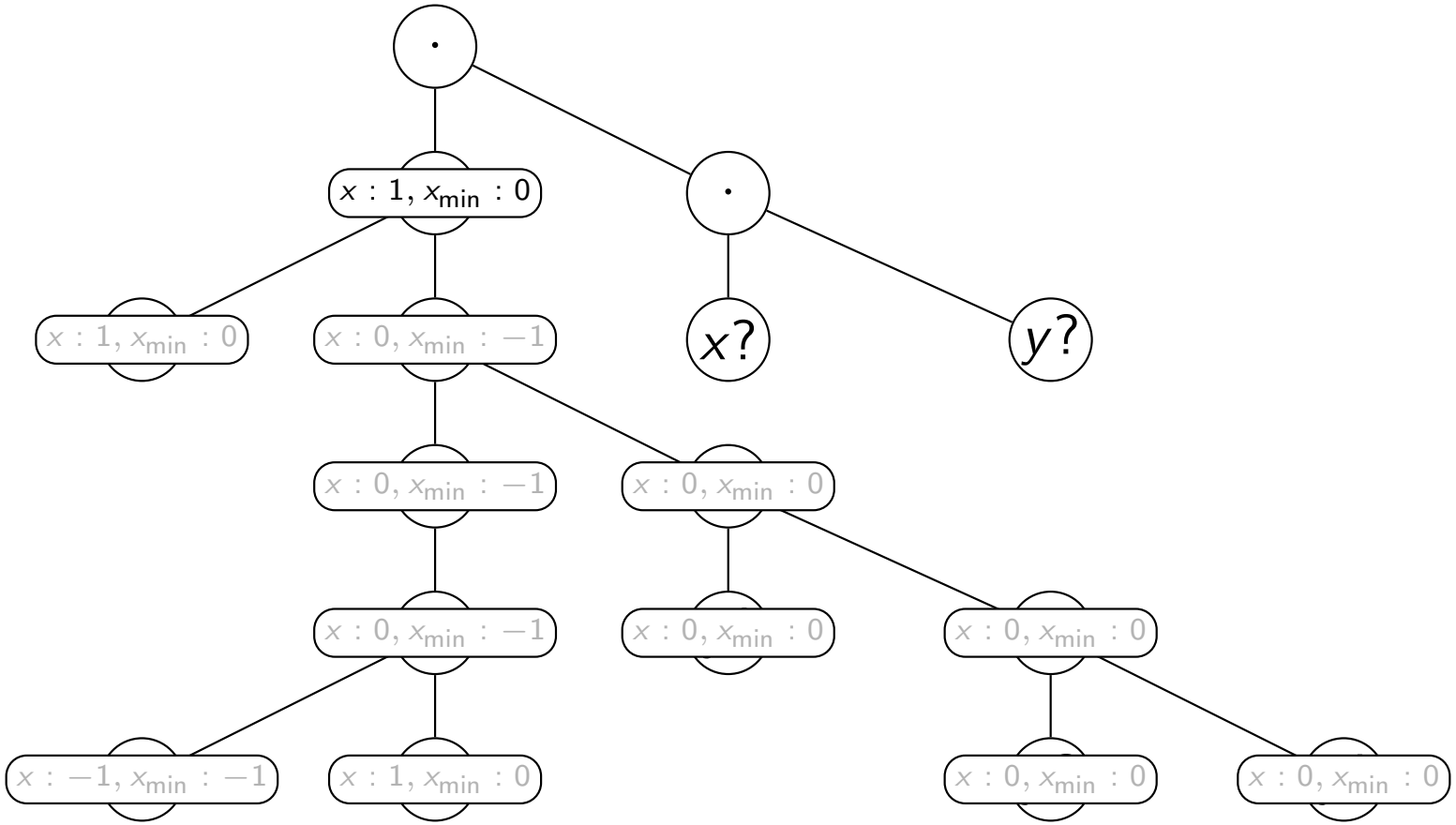
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



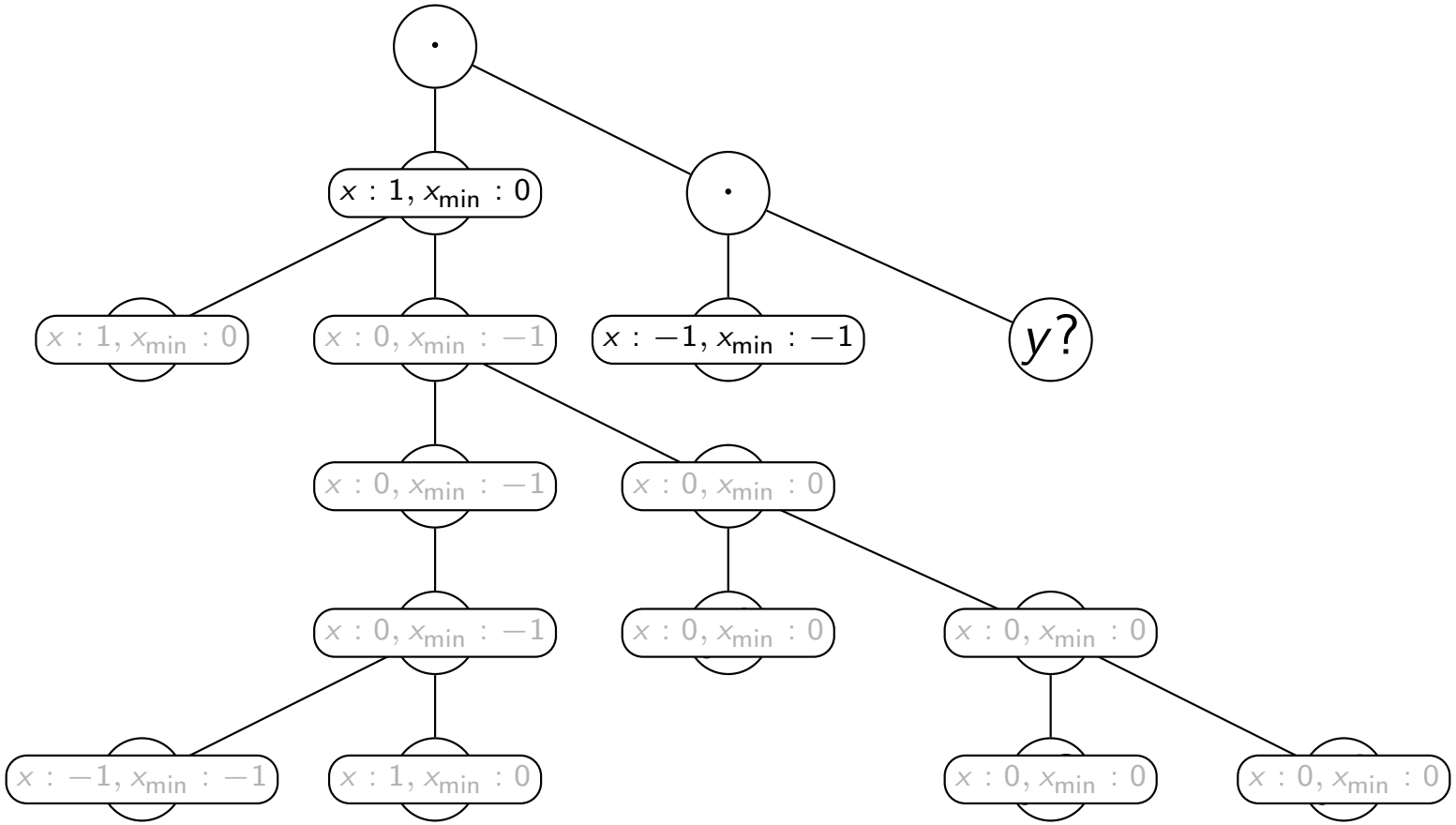
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



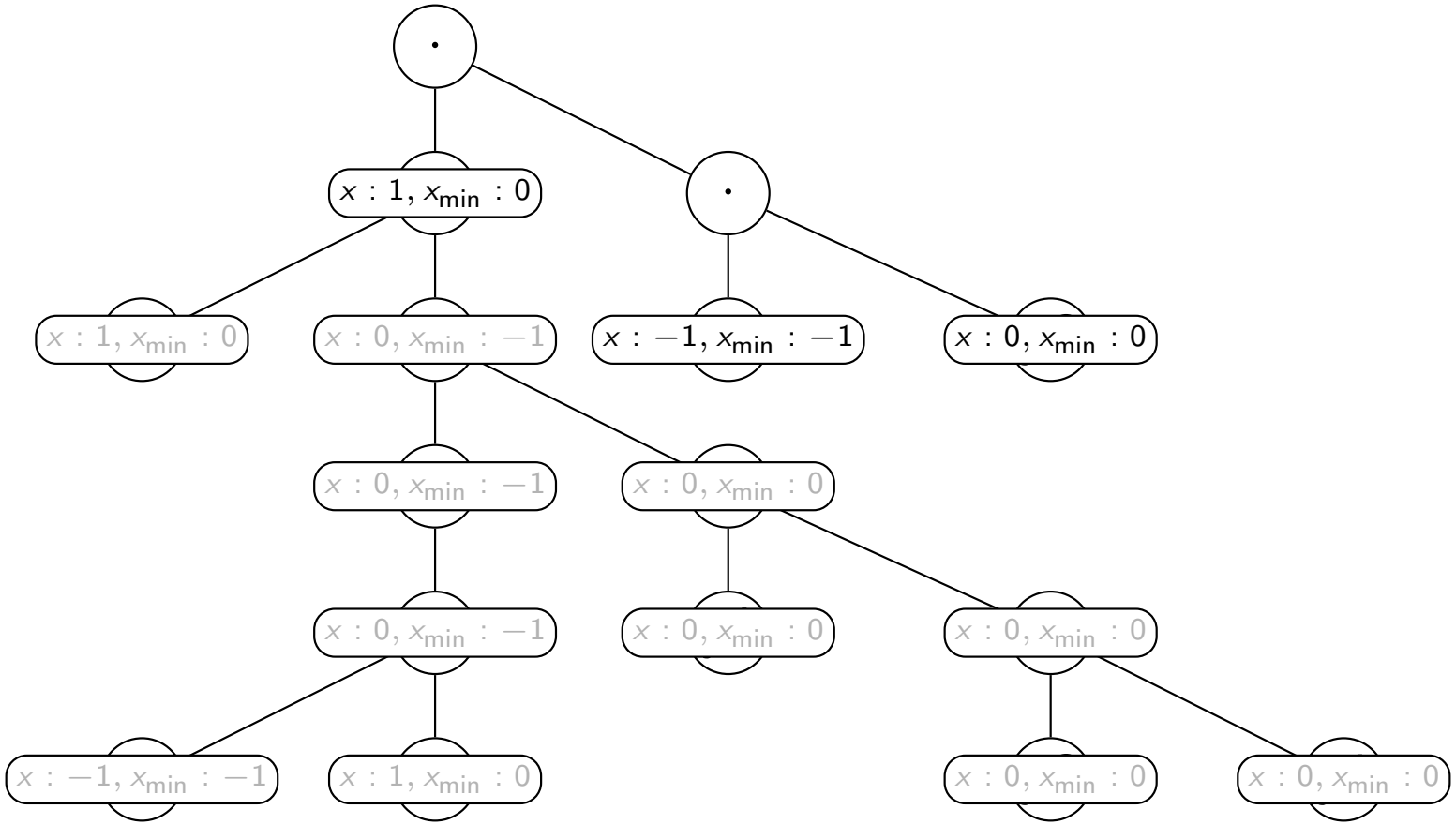
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



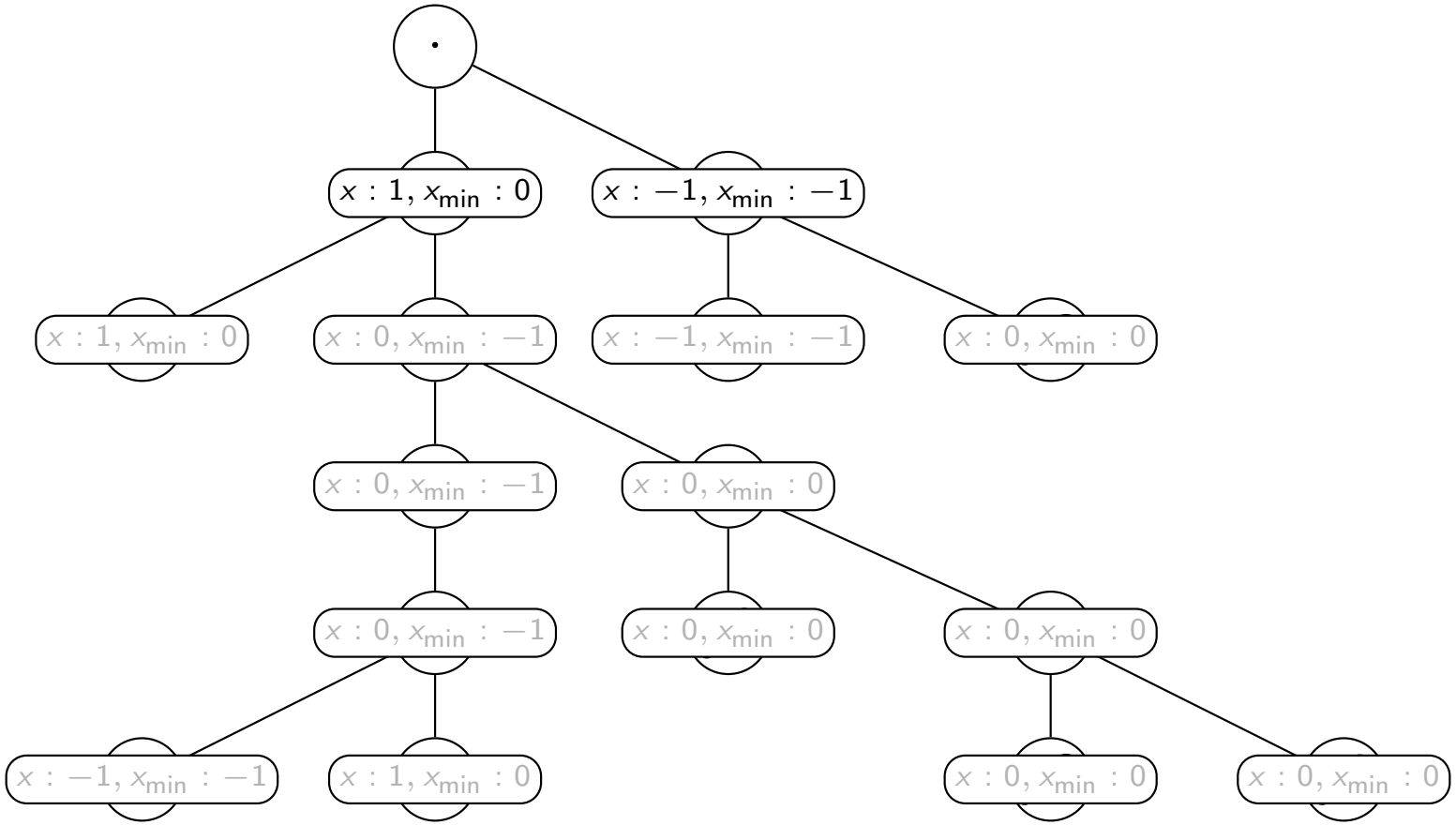
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



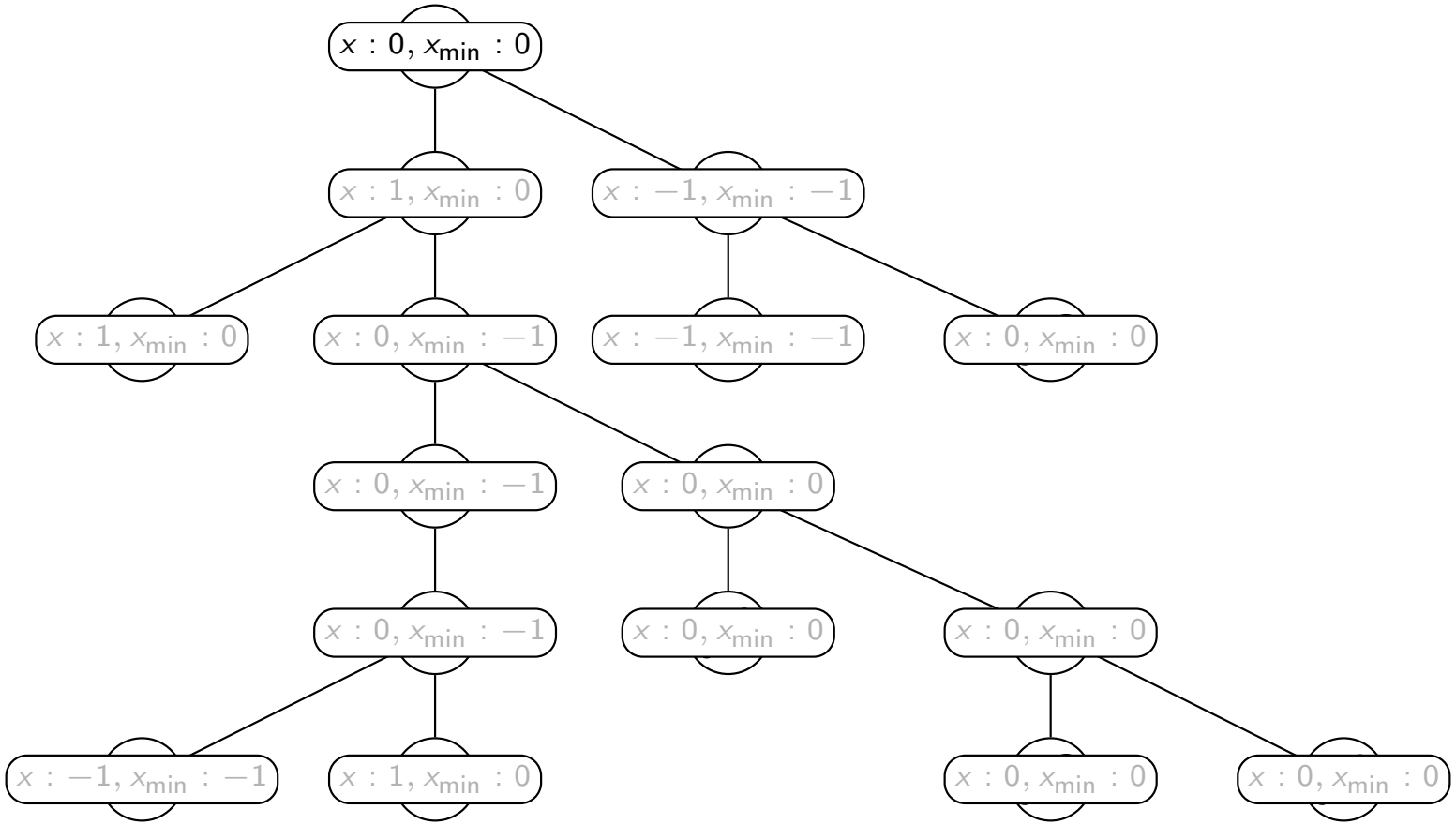
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



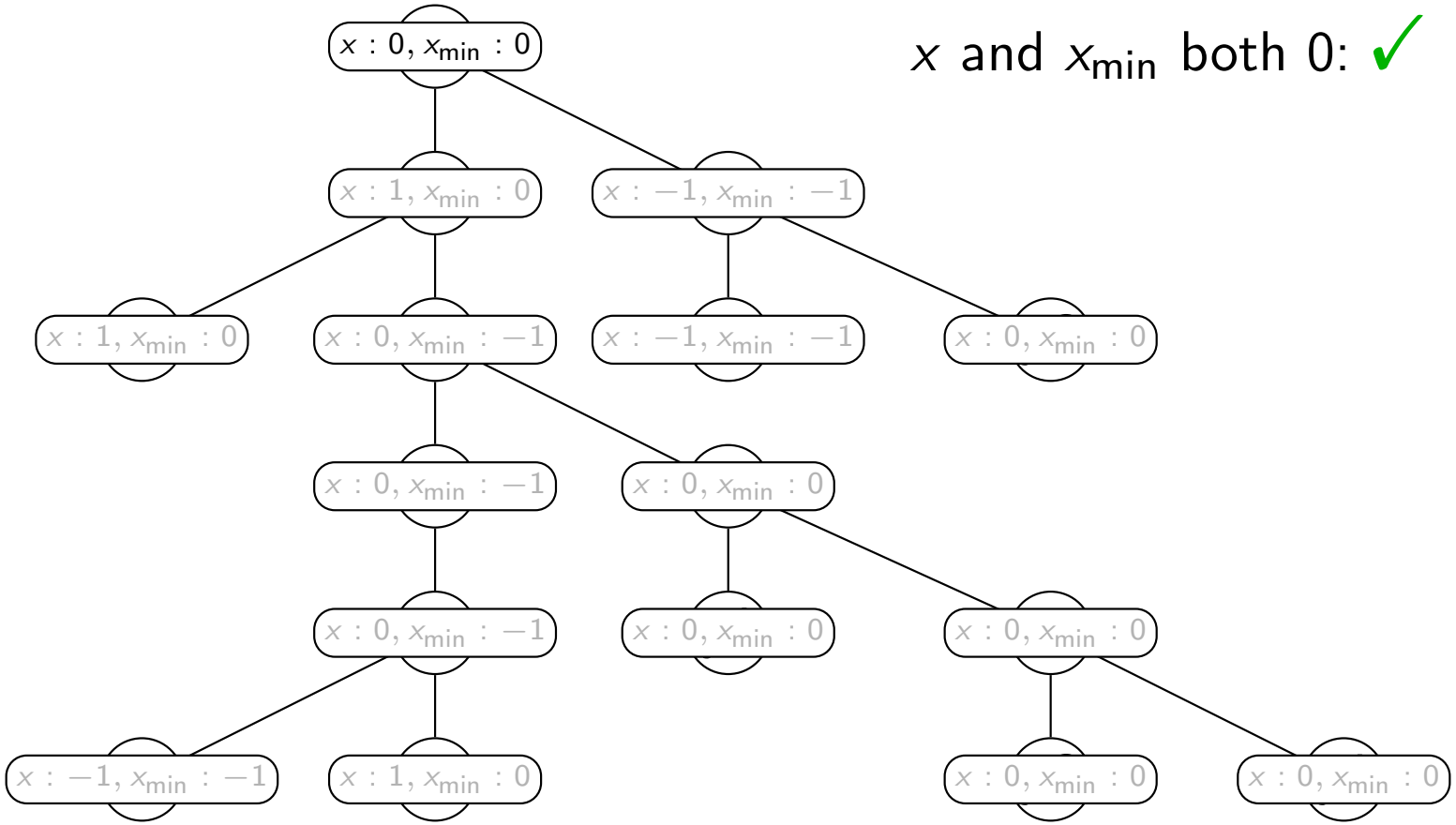
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



Safety in regular expressions

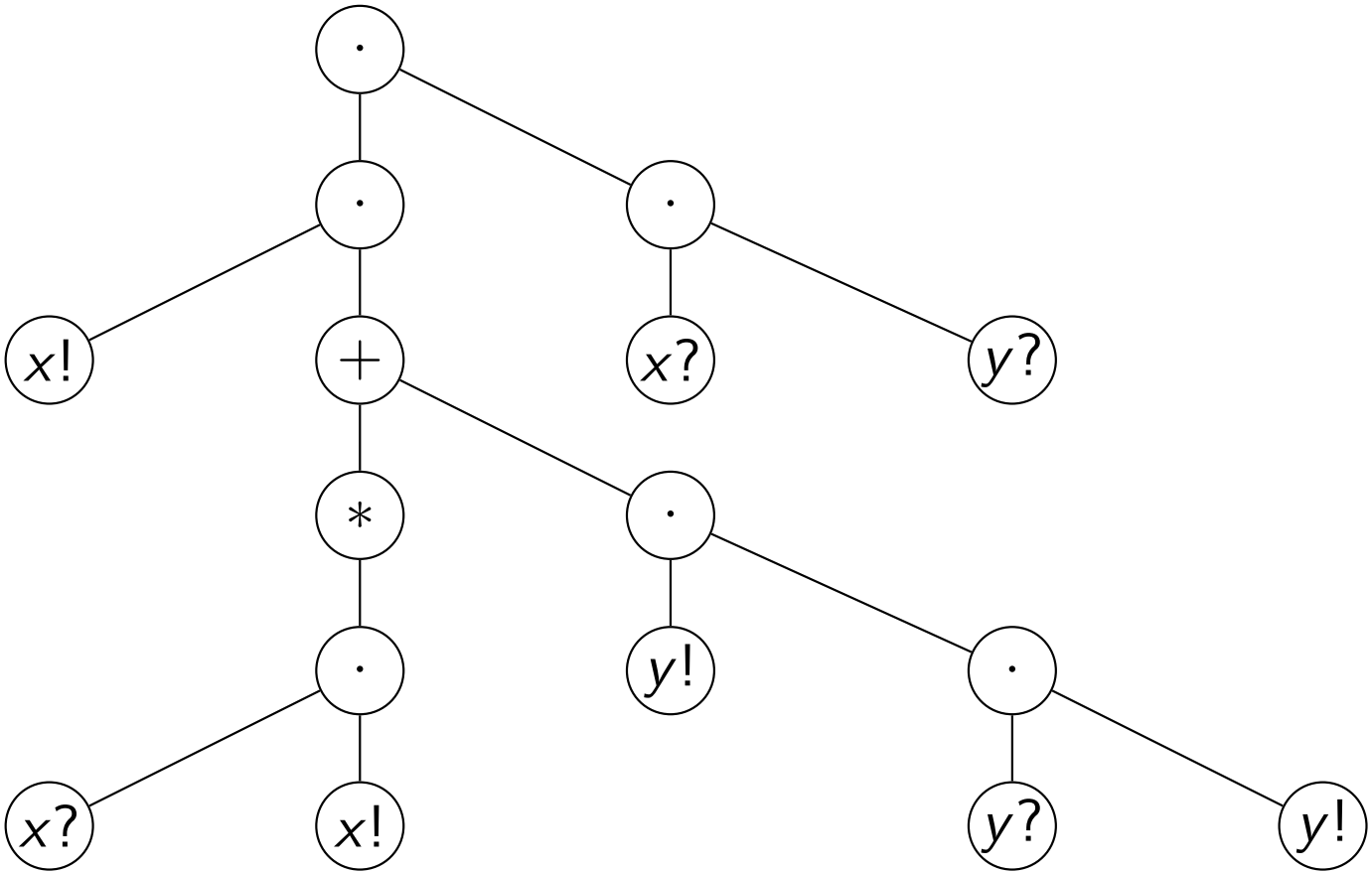
$$e = x!((x?x!)^* + y!y?y!)x?y?$$



x and $x_{\min}$ both 0: ✓

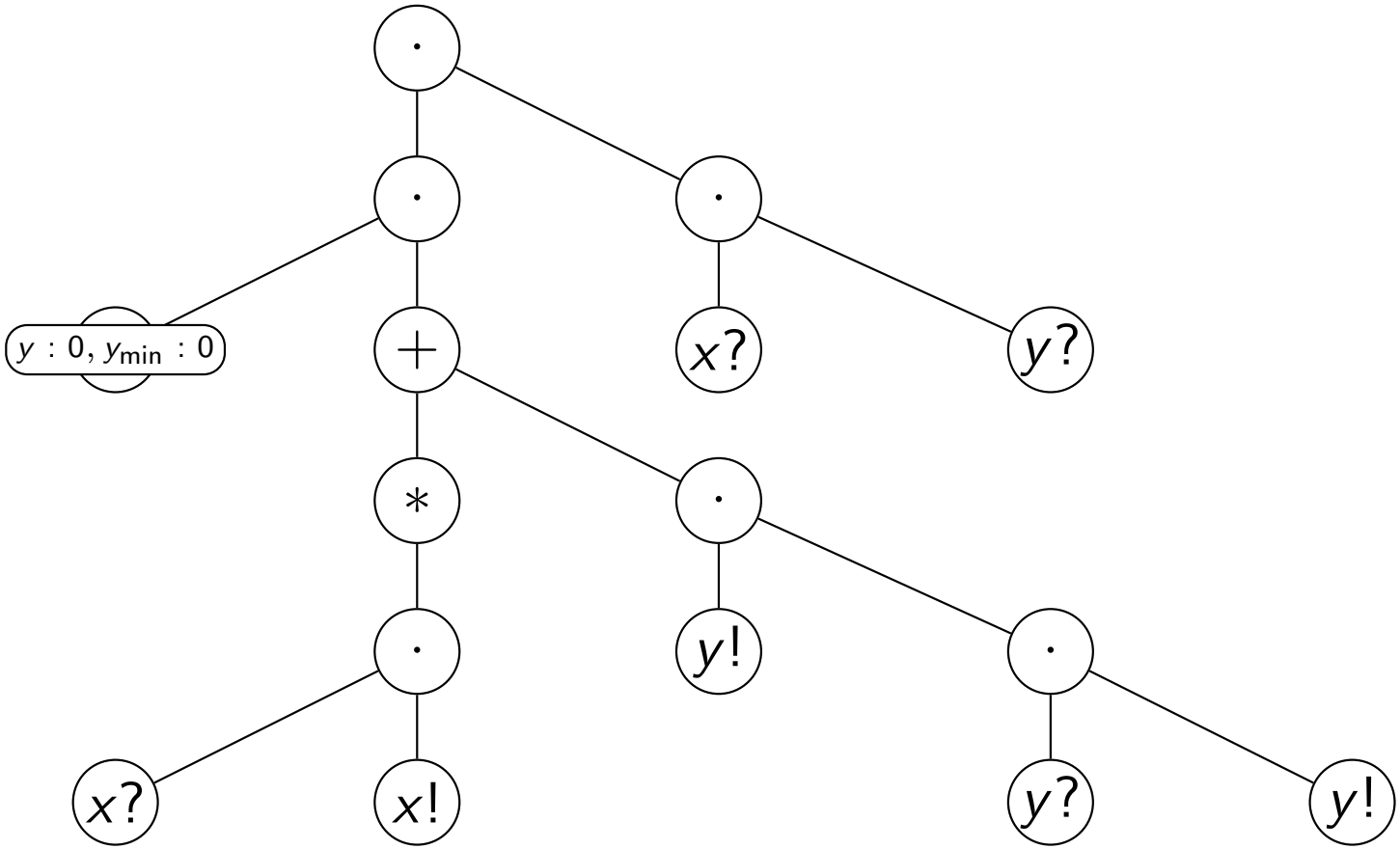
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



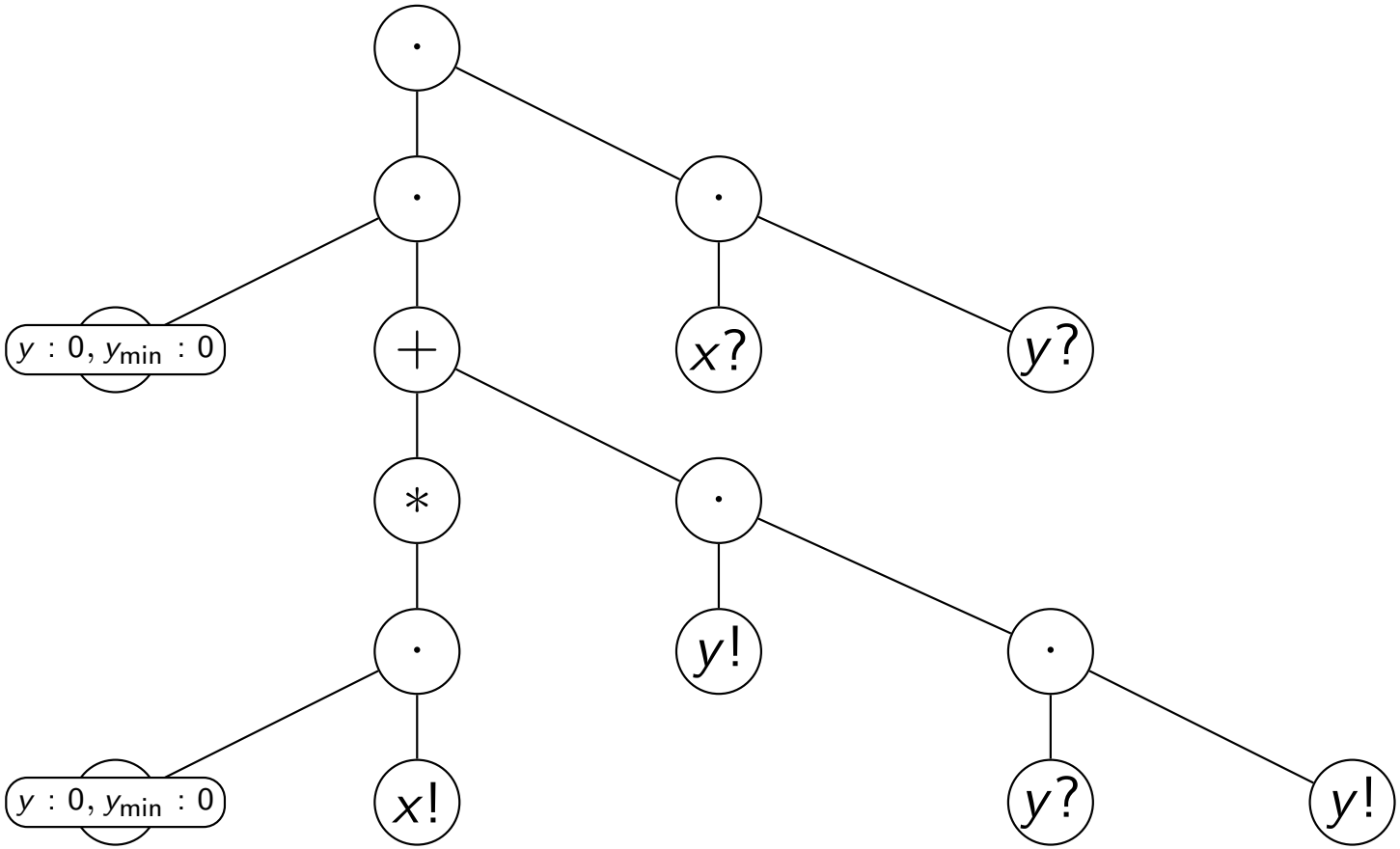
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



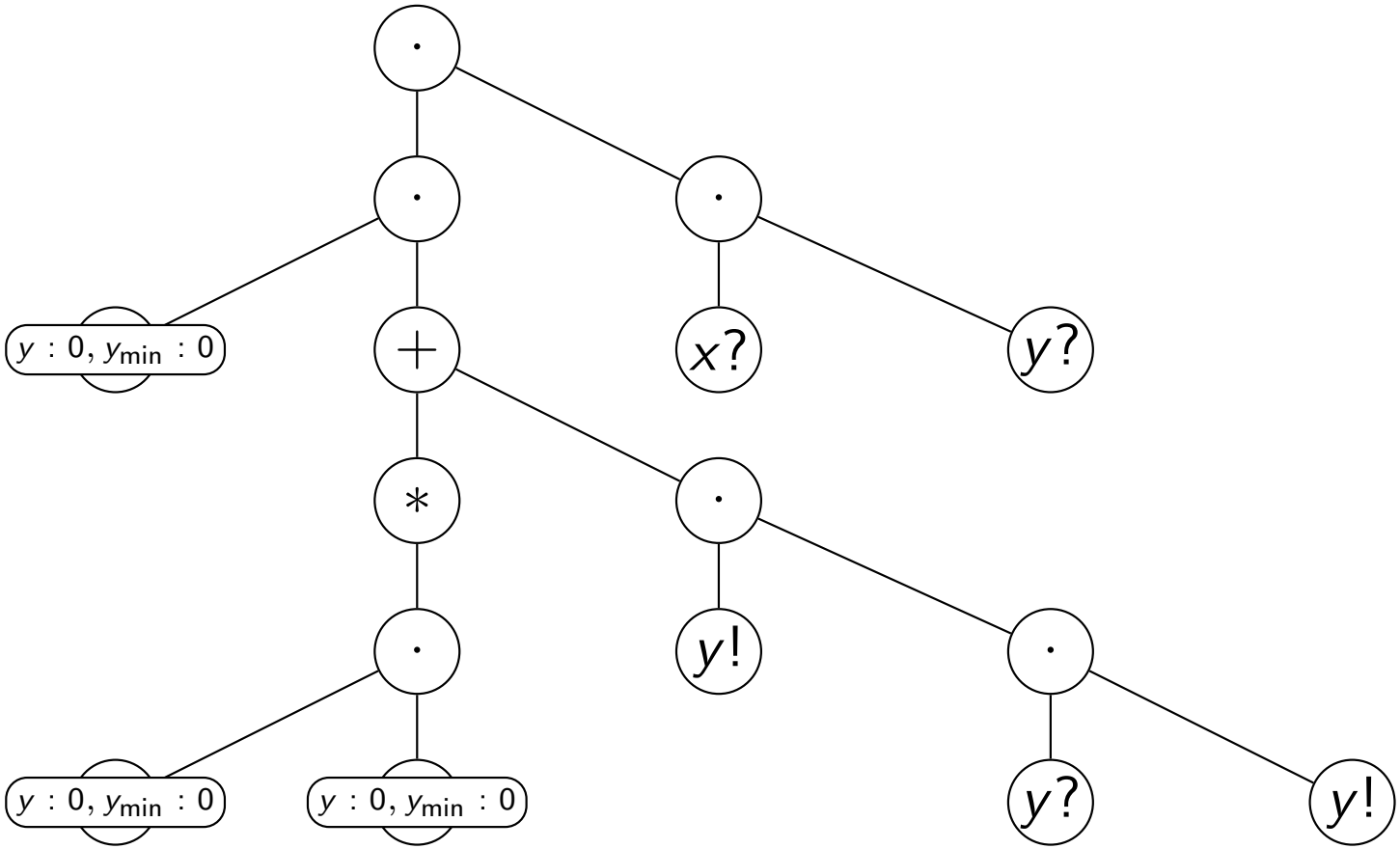
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



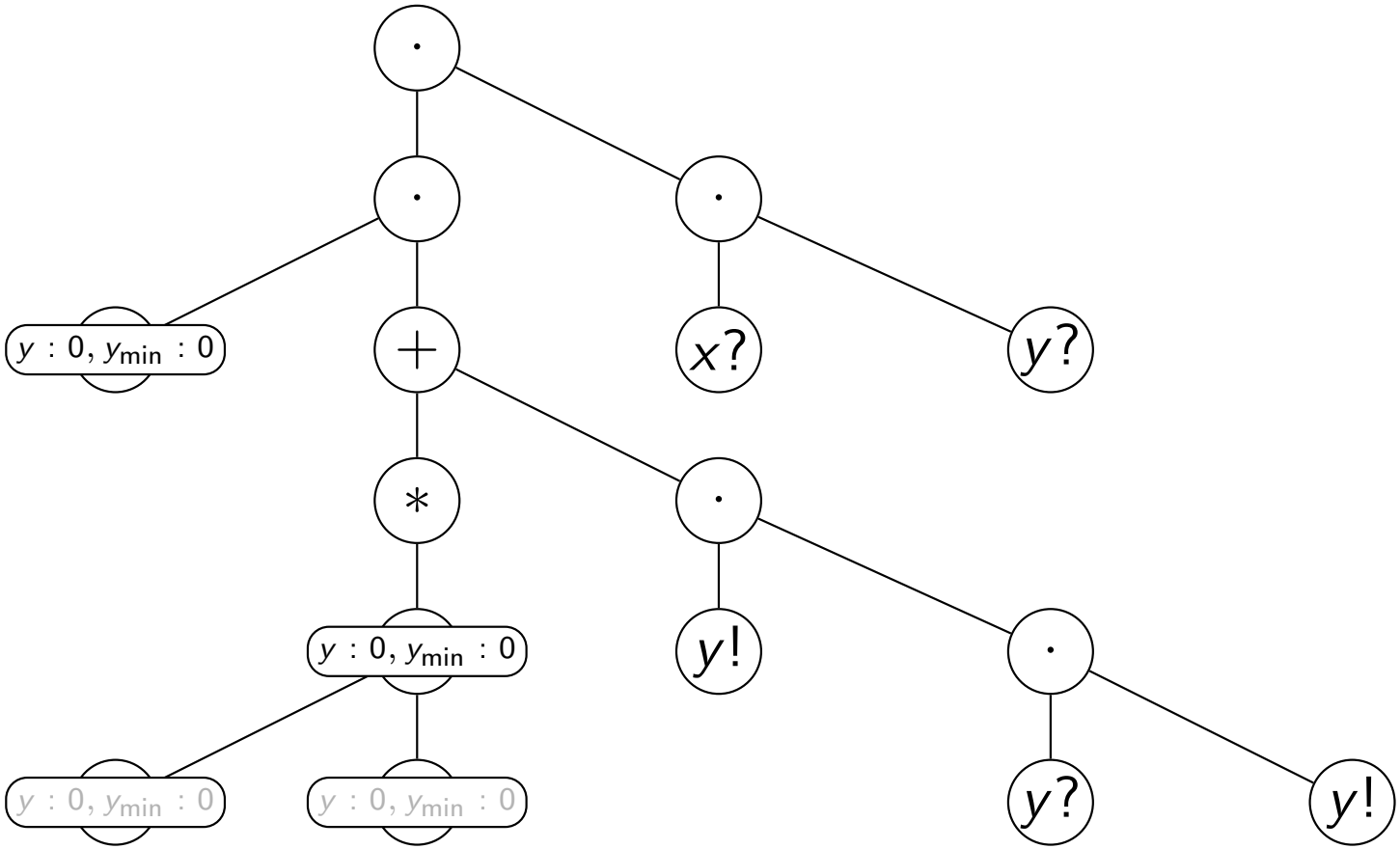
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



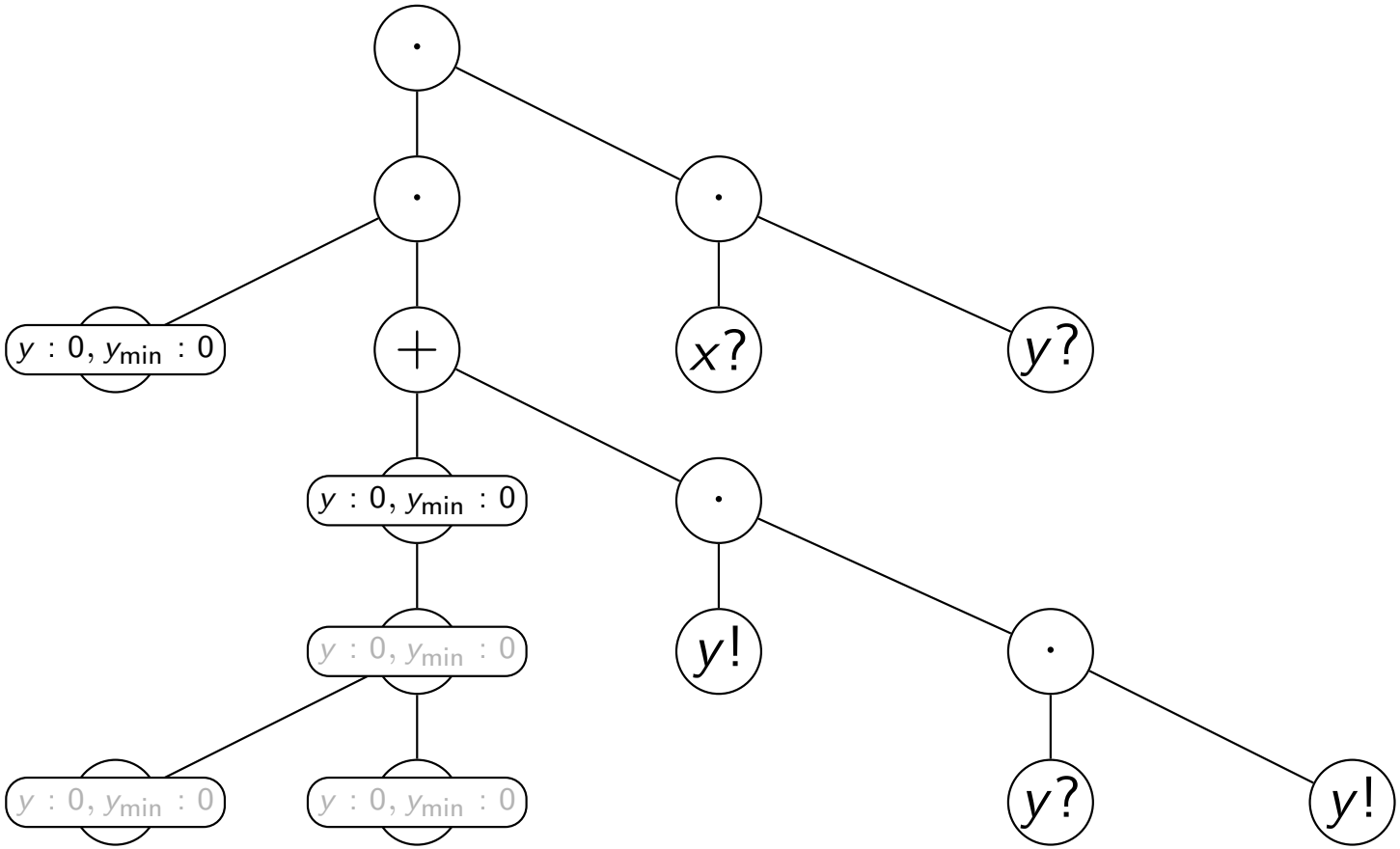
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



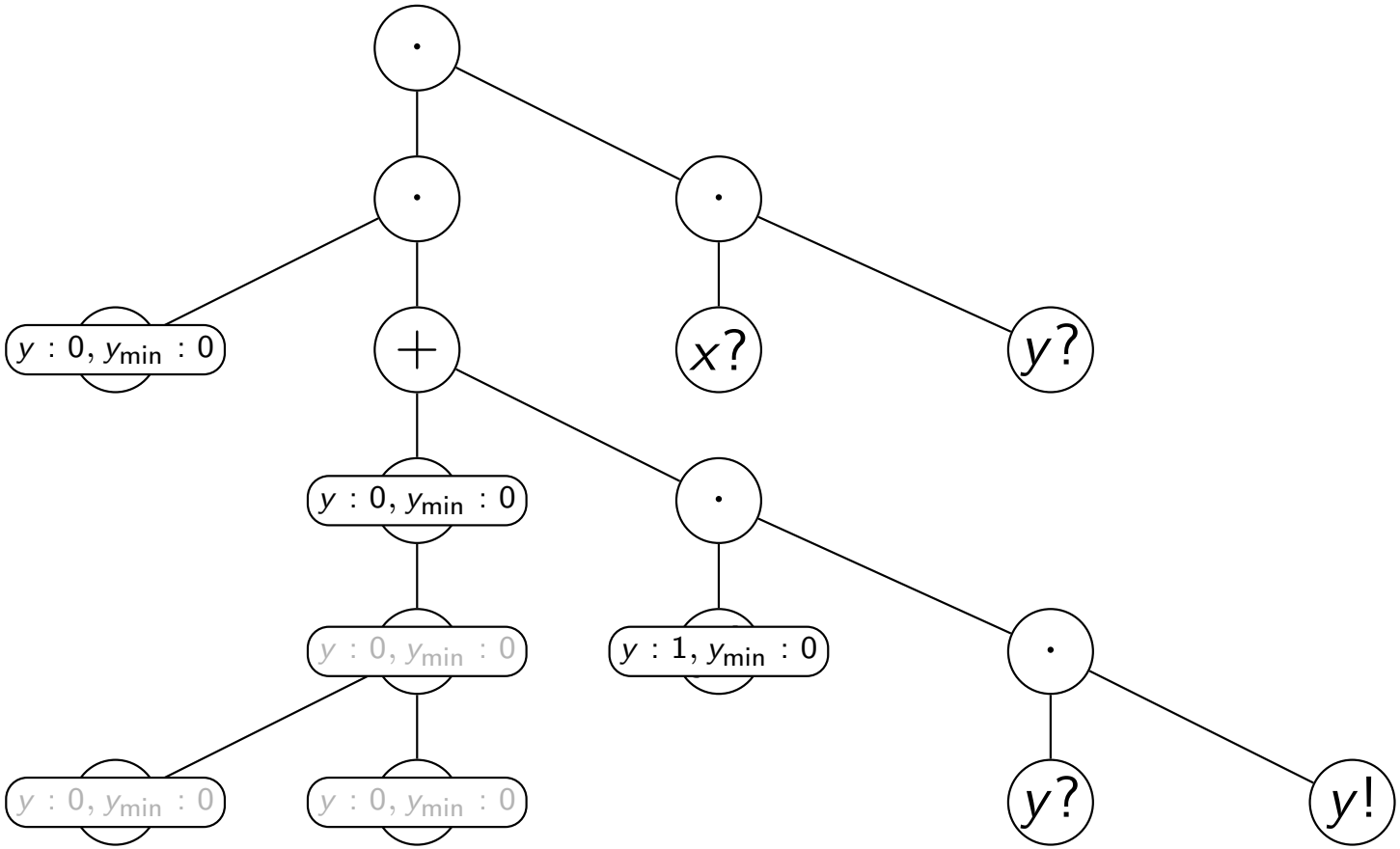
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



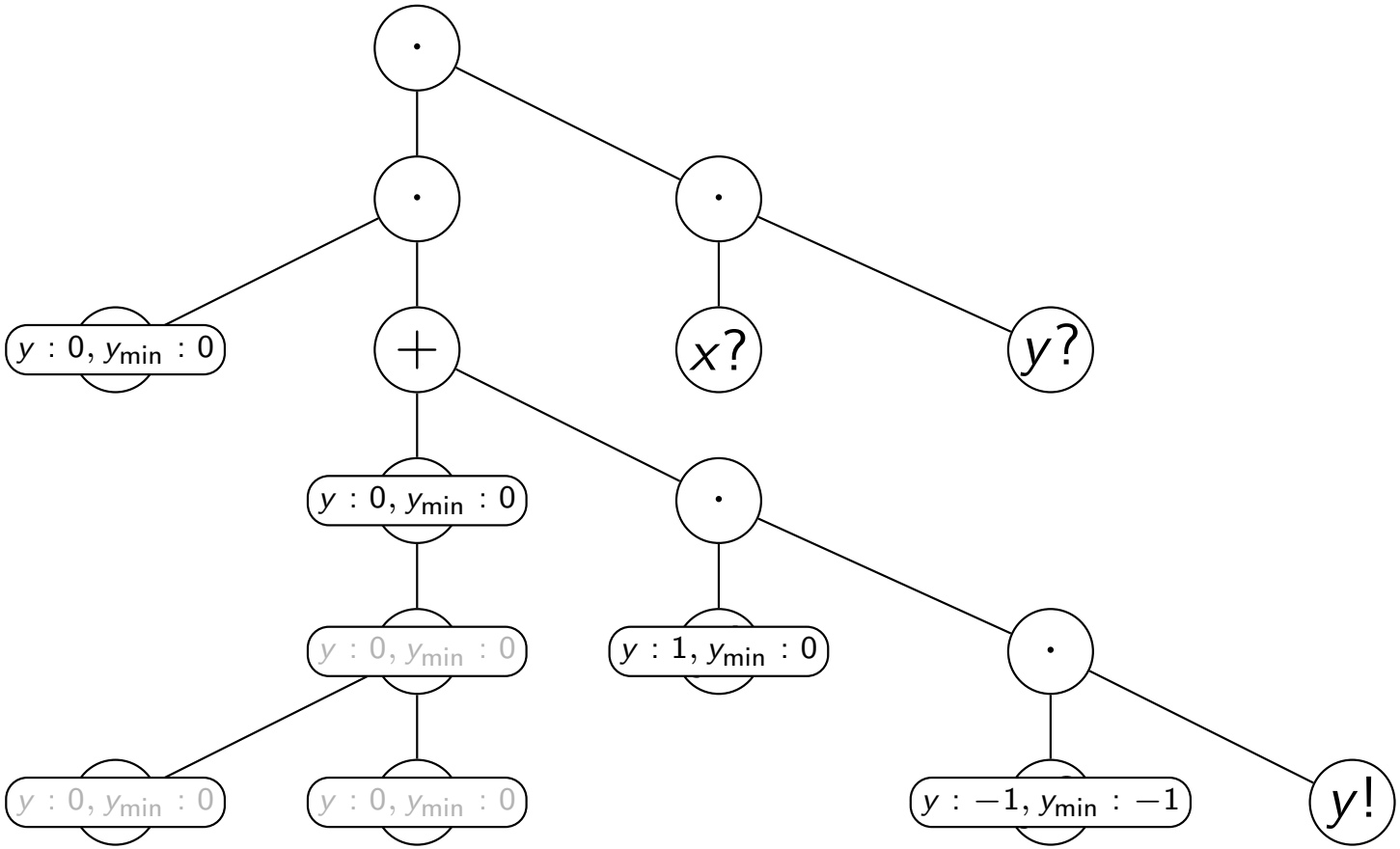
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



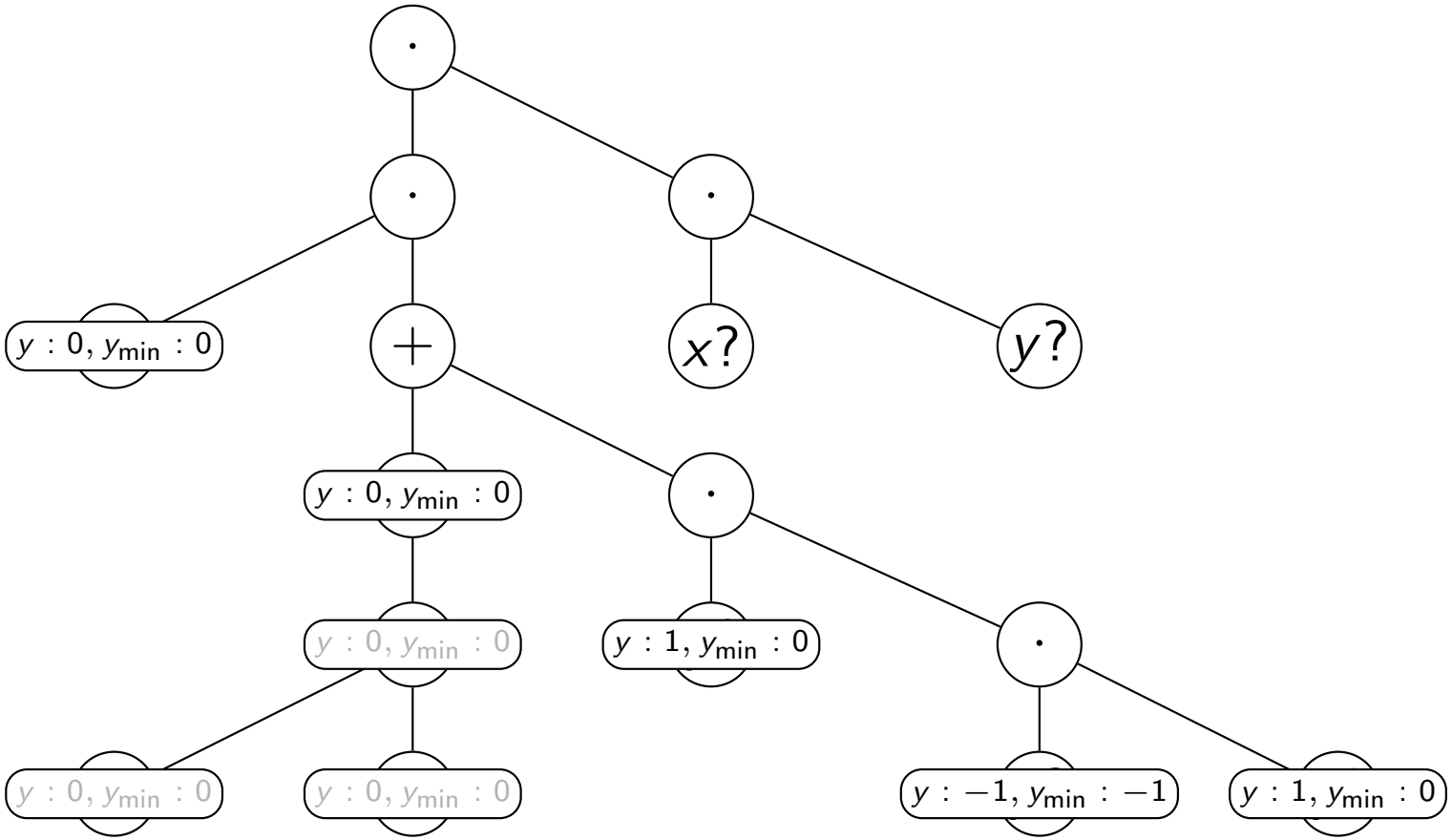
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



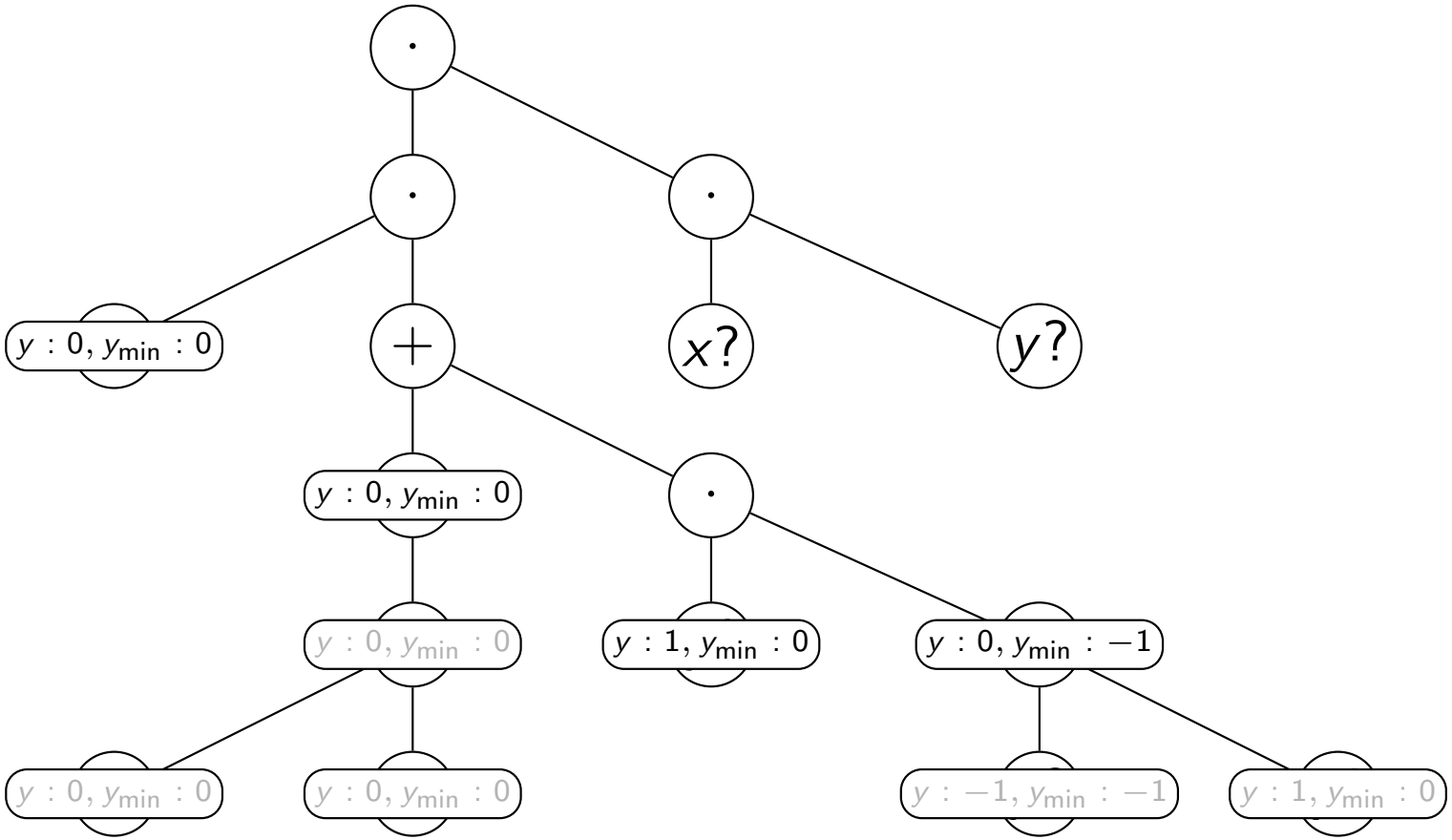
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



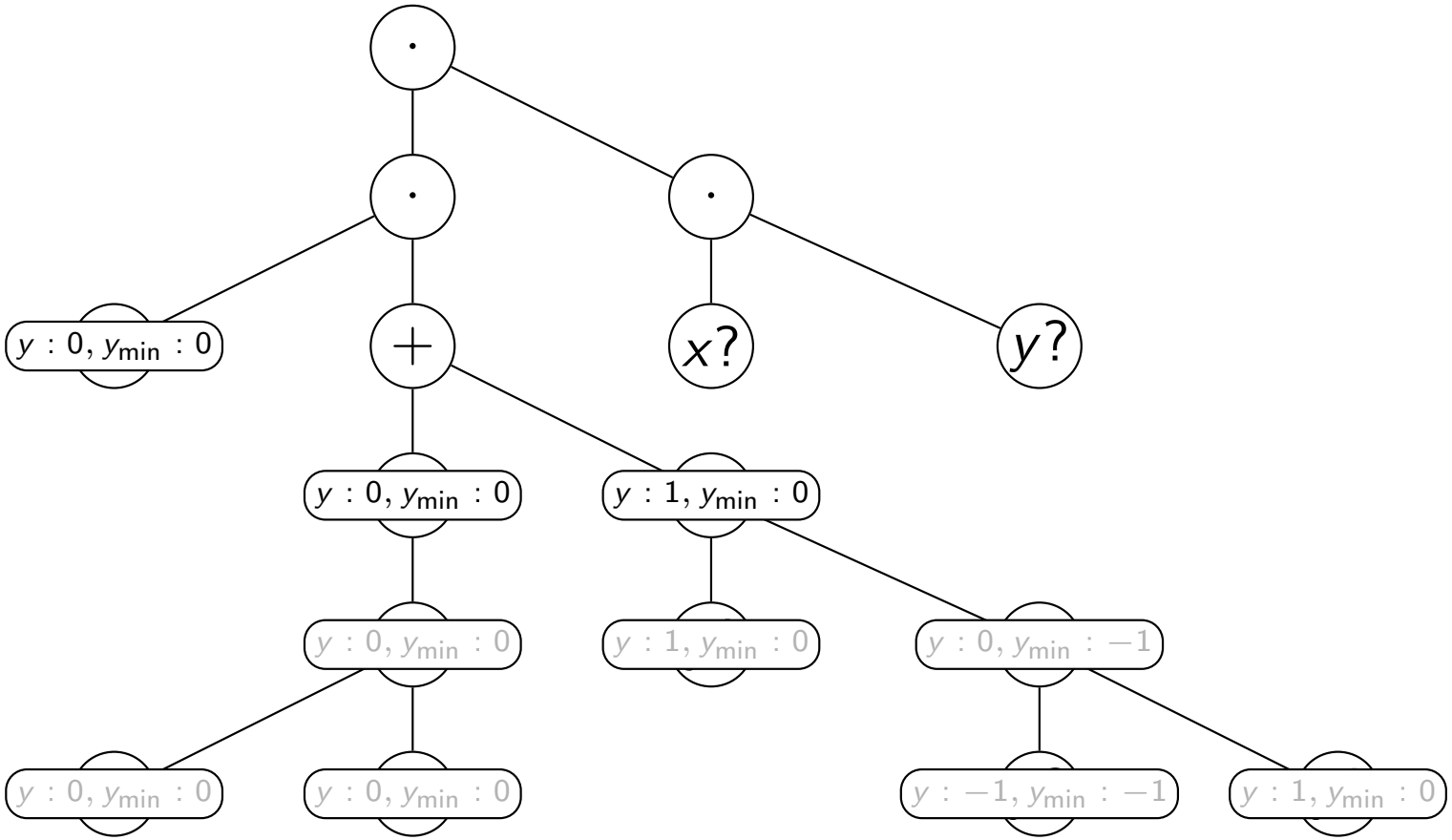
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



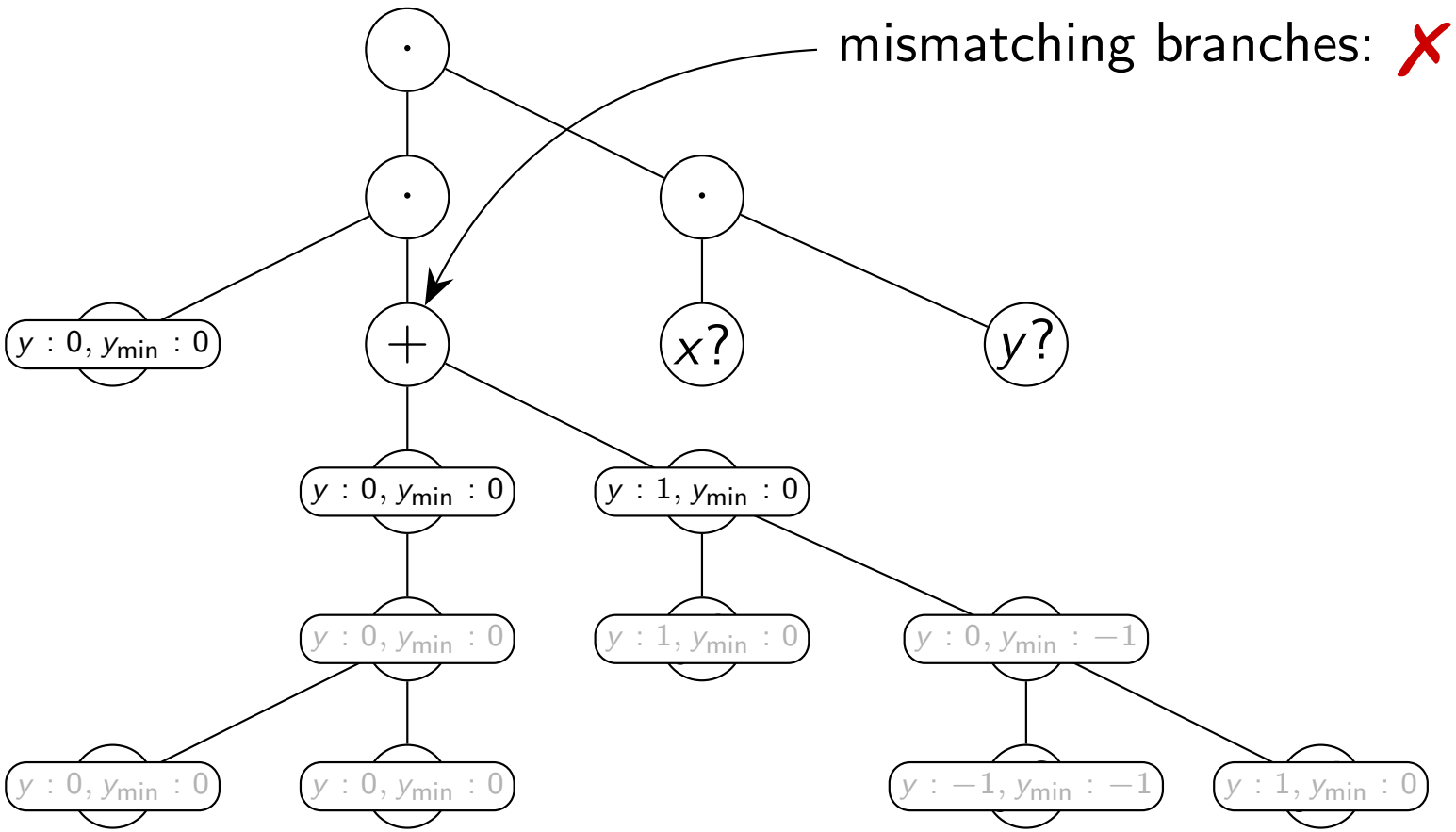
Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



Safety in regular expressions

$$e = x!((x?x!)^* + y!y?y!)x?y?$$



Constructing safe expressions?

Constructing safe expressions?

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^*
\end{aligned}
$$

Constructing safe expressions?

Not safe

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^*
\end{aligned}
$$

Constructing safe expressions?

Not safe

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid \cancel{x!} \mid x? \mid y! \mid y? \mid \ \ldots \\
\mid \cancel{e + e} \mid e \cdot e \mid e^*
\end{aligned}
$$

Constructing safe expressions?

Not safe

$$e ::= \varnothing \mid \lambda \mid \cancel{x!} \mid x? \mid y! \mid y? \mid \ldots$$
$$\mid \cancel{e + e} \mid e \cdot e \mid e^*$$

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

Constructing safe expressions?

Not safe

$$e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

Not enough control

Constructing safe expressions?

Not safe

$$e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

Not enough control

Constructing safe expressions?

Not safe

$$e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

Not enough control

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$
$$\mid e \parallel e \mid e \llcorner e \mid e \lrcorner e \mid \ldots$$

Constructing safe expressions?

Not safe

$$e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

Not enough control

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$
$$\mid e \parallel e \mid e \llcorner e \mid e \lrcorner e \mid \ldots$$

Still not enough control

Constructing safe expressions?

Not safe

$$e ::= \varnothing \mid \lambda \mid x! \mid x? \mid y! \mid y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$

Not enough control

$$e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ldots$$
$$\mid e + e \mid e \cdot e \mid e^*$$
$$\mid e \parallel e \mid e \llcorner e \mid e \lrcorner e \mid \ldots$$

Still not enough control

Constructing safe expressions!

Solution: parametrise the shuffle operator

Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

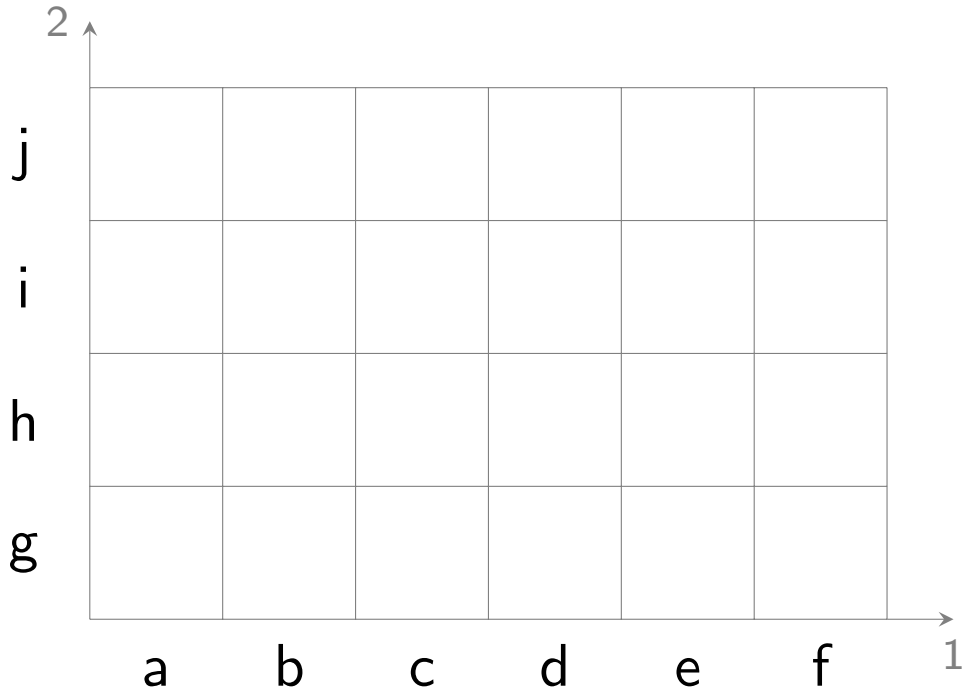
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\mathsf{abcdef}, \mathsf{ghij})$

$=$



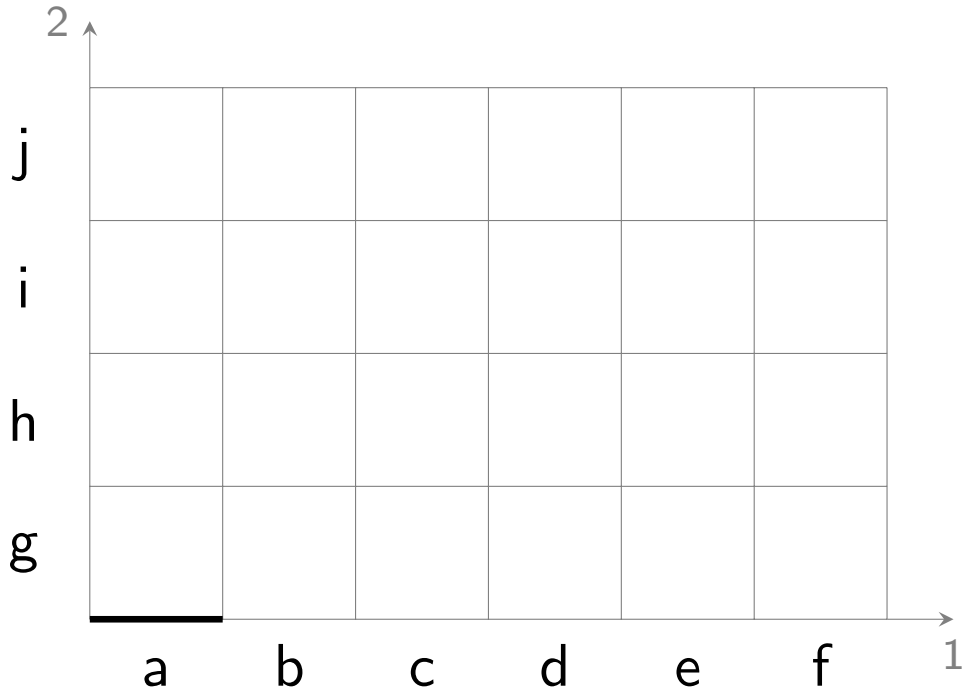
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$$\sqcup\!\sqcup_{1221112112}(\mathrm{abcdef}, \mathrm{ghij})$$

$$= \mathrm{a}$$



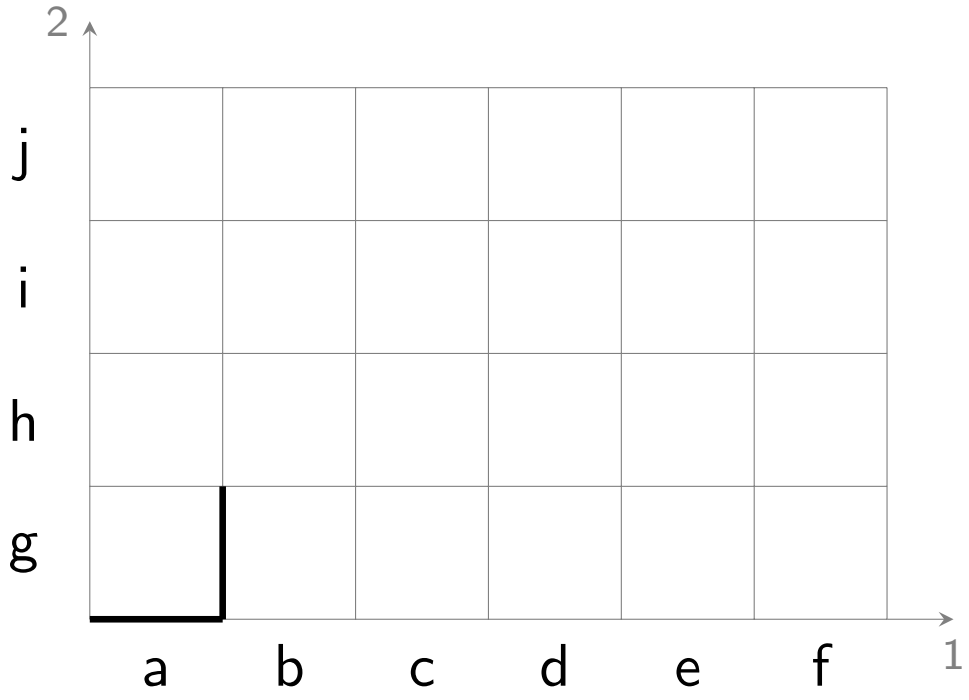
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{12211121 12}(\mathrm{abcdef}, \mathrm{ghij})$

$= \mathrm{ag}$



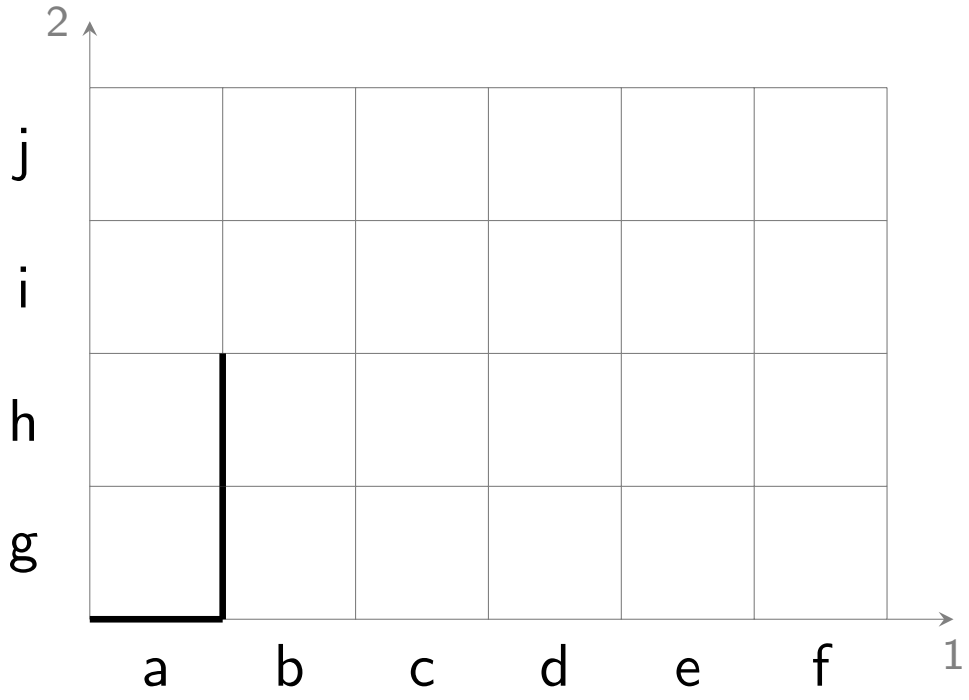
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\mathrm{abcdef}, \mathrm{ghij})$

$= \mathrm{agh}$



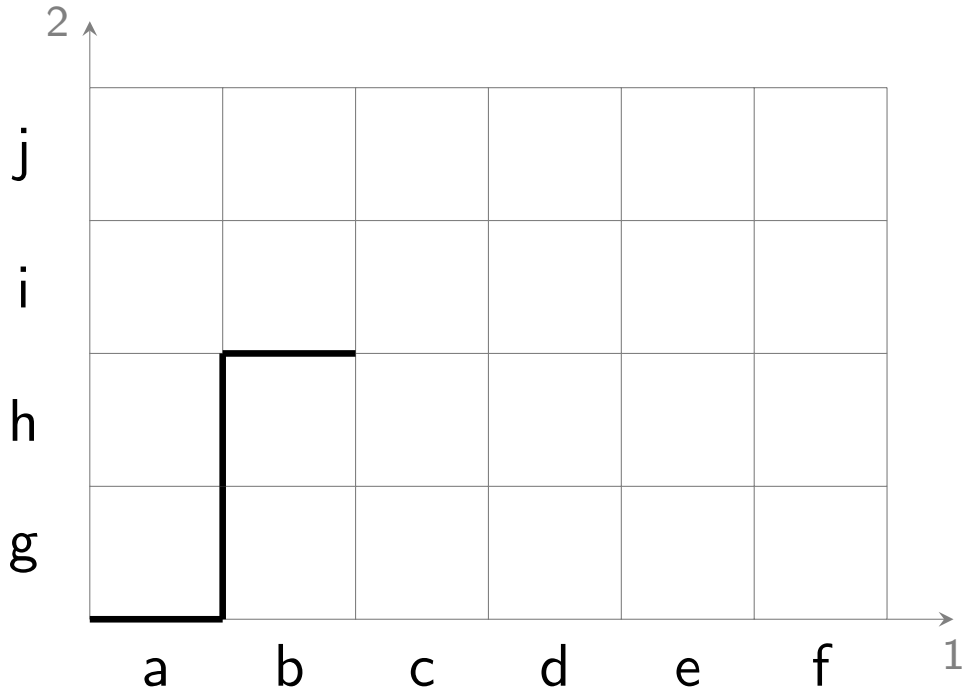
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\mathrm{abcdef}, \mathrm{ghij})$

$= \mathrm{aghb}$



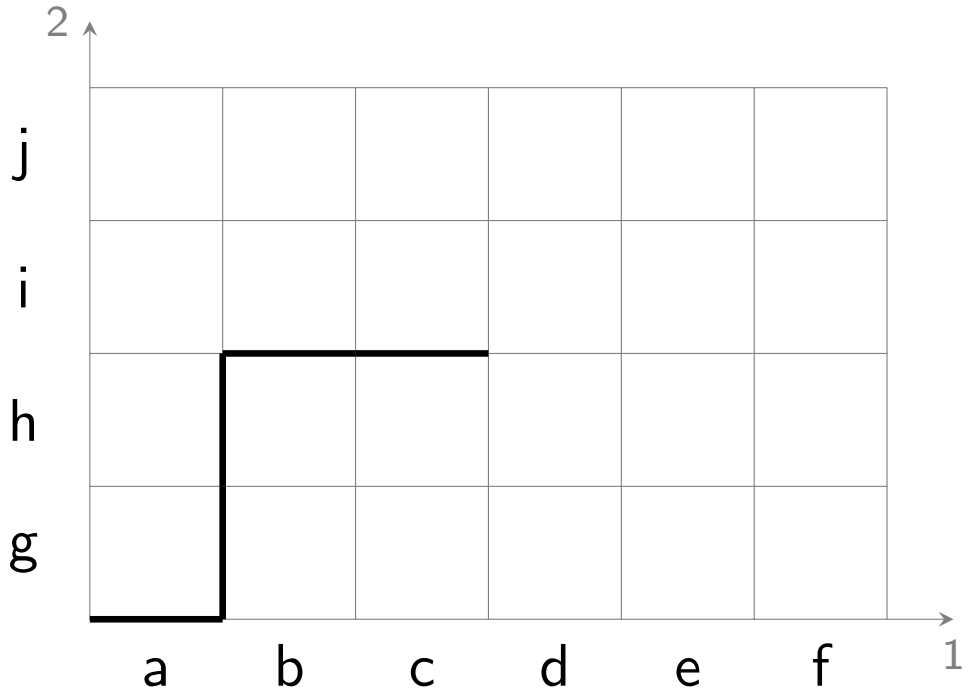
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\text{abcdef}, \text{ghij})$

$= \text{aghbc}$



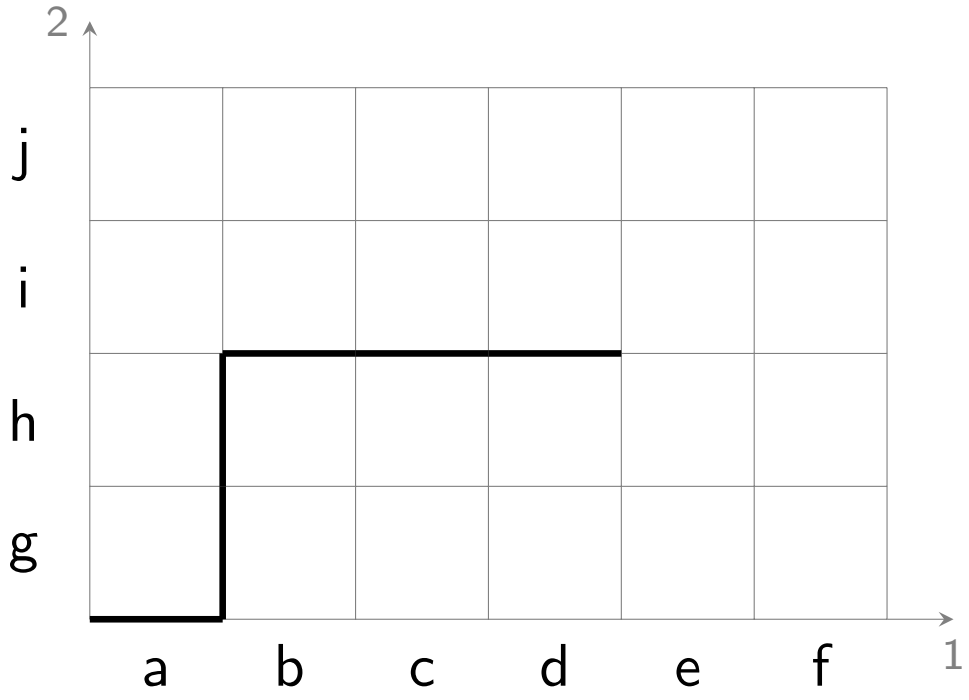
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\mathrm{abcdef}, \mathrm{ghij})$

$= \mathrm{aghbcd}$



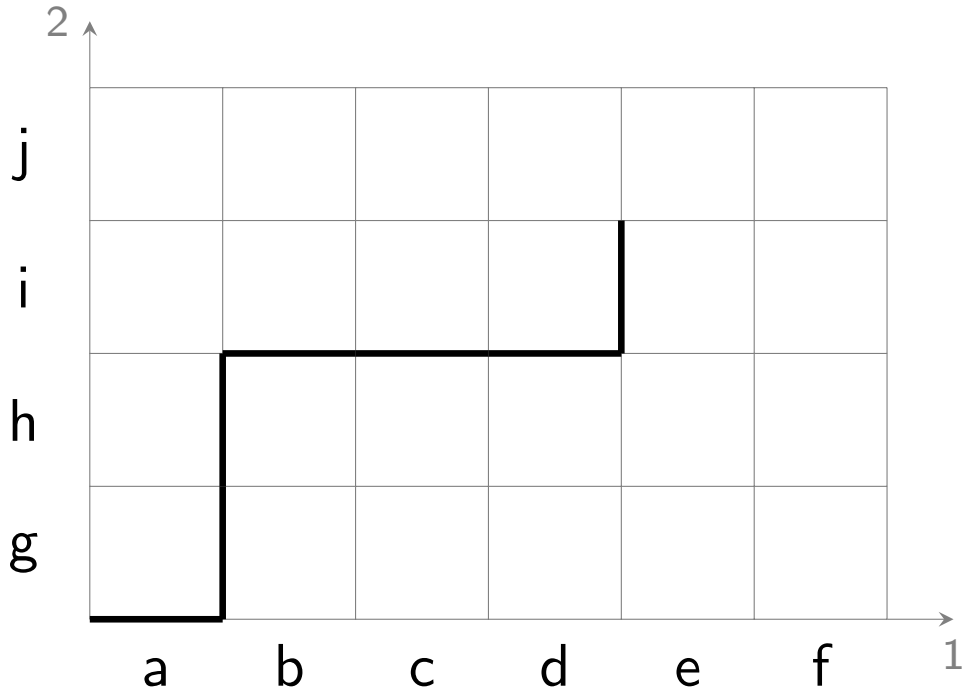
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\mathrm{abcdef}, \mathrm{ghij})$

$= \mathrm{aghbcdi}$



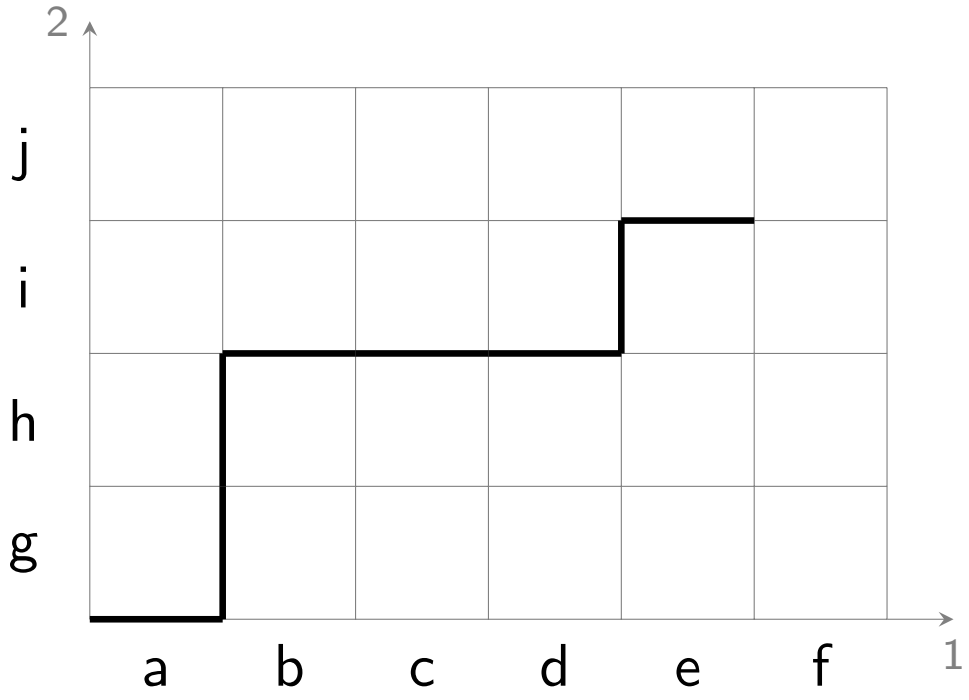
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\text{abcdef}, \text{ghij})$

$= \text{aghbcdie}$



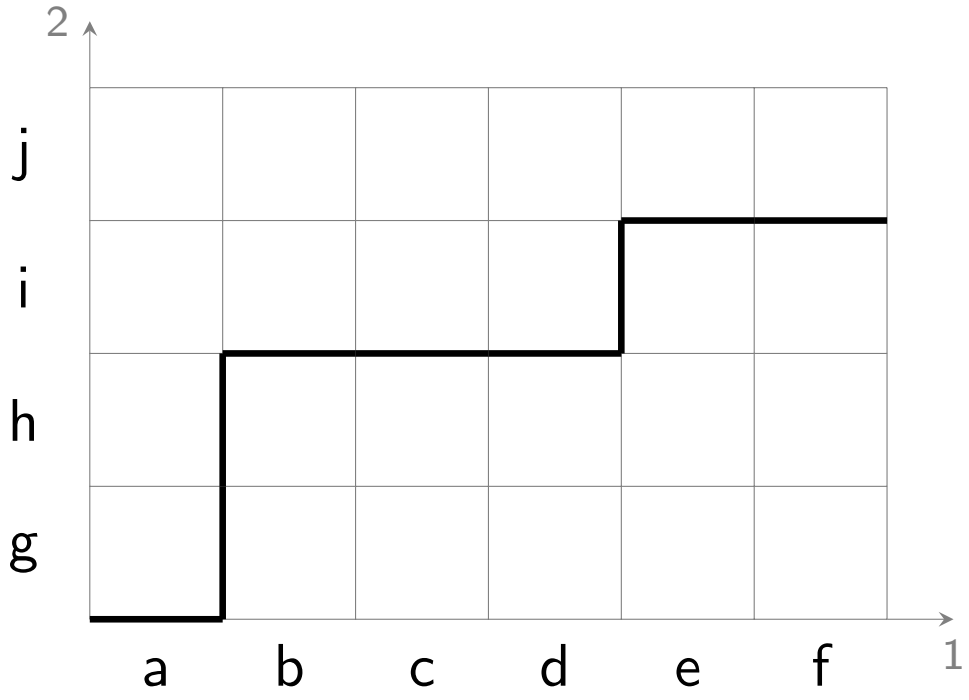
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\text{abcdef}, \text{ghij})$

$= \text{aghbcdief}$



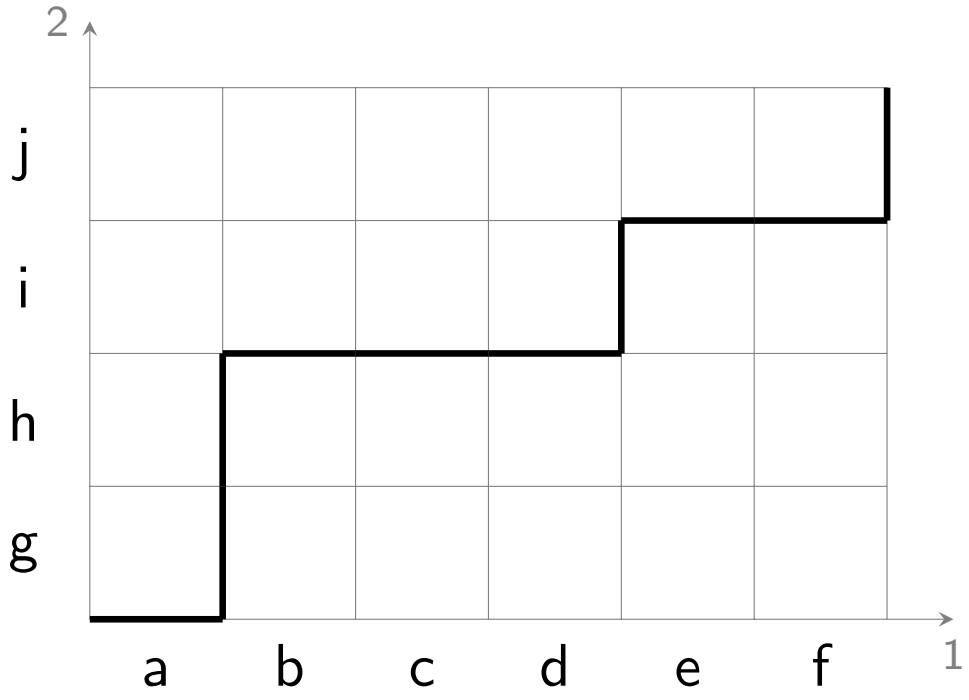
Constructing safe expressions!

Solution: parametrise the shuffle operator

"Shuffle on trajectories" (Mateescu et al., 1998)

$\sqcup\!\sqcup_{1221112112}(\text{abcdef}, \text{ghij})$

$= \text{aghbcdiefj}$



Constructing safe expressions!

Undefined (does not 'fit'):

$$\sqcup_{1122}(\mathsf{aaaa}, \mathsf{bbbb}) = \mathsf{abef}\ldots ?$$

Constructing safe expressions!

Undefined (does not 'fit'):

$$\sqcup\!\sqcup_{1122}(\mathsf{aaaa}, \mathsf{bbbb}) = \mathsf{abef}\ldots ?$$

$n > 2$ dimensions:

$$\sqcup\!\sqcup_{123132}(\mathsf{aa}, \mathsf{bb}, \mathsf{cc}) = \mathsf{abcacb}$$

Constructing safe expressions!

Undefined (does not 'fit'):

$$\sqcup\!\sqcup_{1122}(\mathsf{aaaa}, \mathsf{bbbb}) = \mathsf{abef}\ldots?$$

$n > 2$ dimensions:

$$\sqcup\!\sqcup_{123132}(\mathsf{aa}, \mathsf{bb}, \mathsf{cc}) = \mathsf{abcacb}$$

Languages:

$$\sqcup\!\sqcup_{\{1122,2211\}}(\{\mathsf{aa}, \mathsf{bb}\}, \{\mathsf{cc}, \mathsf{ddd}\}) = \{\mathsf{aacc}, \mathsf{bbcc}, \mathsf{ccaa}, \mathsf{ccbb}\}$$

Constructing safe expressions!

Undefined (does not 'fit'):

$$\sqcup\!\sqcup_{1122}(\mathsf{aaaa}, \mathsf{bbbb}) = \mathsf{abef}\ldots ?$$

$n > 2$ dimensions:

$$\sqcup\!\sqcup_{123132}(\mathsf{aa}, \mathsf{bb}, \mathsf{cc}) = \mathsf{abcacb}$$

Languages:

$$\sqcup\!\sqcup_{\{1122,2211\}}(\{\mathsf{aa}, \mathsf{bb}\}, \{\mathsf{cc}, \mathsf{ddd}\}) = \{\mathsf{aacc}, \mathsf{bbcc}, \mathsf{ccaa}, \mathsf{ccbb}\}$$

Expressions: same as languages

Constructing safe expressions!

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^* \\
\mid \sqcup_\theta(e) \mid \sqcup_\theta(e, e) \mid \ \ldots
\end{aligned}
$$

$$
\begin{aligned}
\theta ::= \varnothing \mid \lambda \mid 1 \mid 2 \mid \ \ldots \\
\mid \theta + \theta \mid \theta \cdot \theta \mid \theta^*
\end{aligned}
$$

Constructing safe expressions!

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^* \\
\mid \sqcup_\theta(e) \mid \sqcup_\theta(e, e) \mid \ \ldots
\end{aligned}
$$

$$
\begin{aligned}
\theta ::= \varnothing \mid \lambda \mid 1 \mid 2 \mid \ \ldots \\
\mid \theta + \theta \mid \theta \cdot \theta \mid \theta^*
\end{aligned}
$$

Safe:

Constructing safe expressions!

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^* \\
\mid \sqcup_\theta(e) \mid \sqcup_\theta(e, e) \mid \ \ldots
\end{aligned}
$$

Safe: ✓

$$
\begin{aligned}
\theta ::= \varnothing \mid \lambda \mid 1 \mid 2 \mid \ \ldots \\
\mid \theta + \theta \mid \theta \cdot \theta \mid \theta^*
\end{aligned}
$$

Constructing safe expressions!

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^* \\
\mid \sqcup_\theta(e) \mid \sqcup_\theta(e, e) \mid \ \ldots
\end{aligned}
$$

$$
\begin{aligned}
\theta ::= \varnothing \mid \lambda \mid 1 \mid 2 \mid \ \ldots \\
\mid \theta + \theta \mid \theta \cdot \theta \mid \theta^*
\end{aligned}
$$

Safe: ✓

Expressive:

Constructing safe expressions!

$$
\begin{aligned}
e ::= \varnothing \mid \lambda \mid x! \cdot x? \mid y! \cdot y? \mid \ \ldots \\
\mid e + e \mid e \cdot e \mid e^* \\
\mid \sqcup_\theta(e) \mid \sqcup_\theta(e, e) \mid \ \ldots
\end{aligned}
$$

$$
\begin{aligned}
\theta ::= \varnothing \mid \lambda \mid 1 \mid 2 \mid \ \ldots \\
\mid \theta + \theta \mid \theta \cdot \theta \mid \theta^*
\end{aligned}
$$

Safe: ✓

Expressive: 100% ✓

Constructing safe expressions!

$$x!(x!x? + x?x!)^* x!x?(x? + x?x!x?)$$

Constructing safe expressions!

$$x!(x!x? + x?x!)^* x!x?(x? + x?x!x?)$$
$$= x!((x!x?)^*(x?x!)^*)^* x!x?(x? + x?x!x?)$$

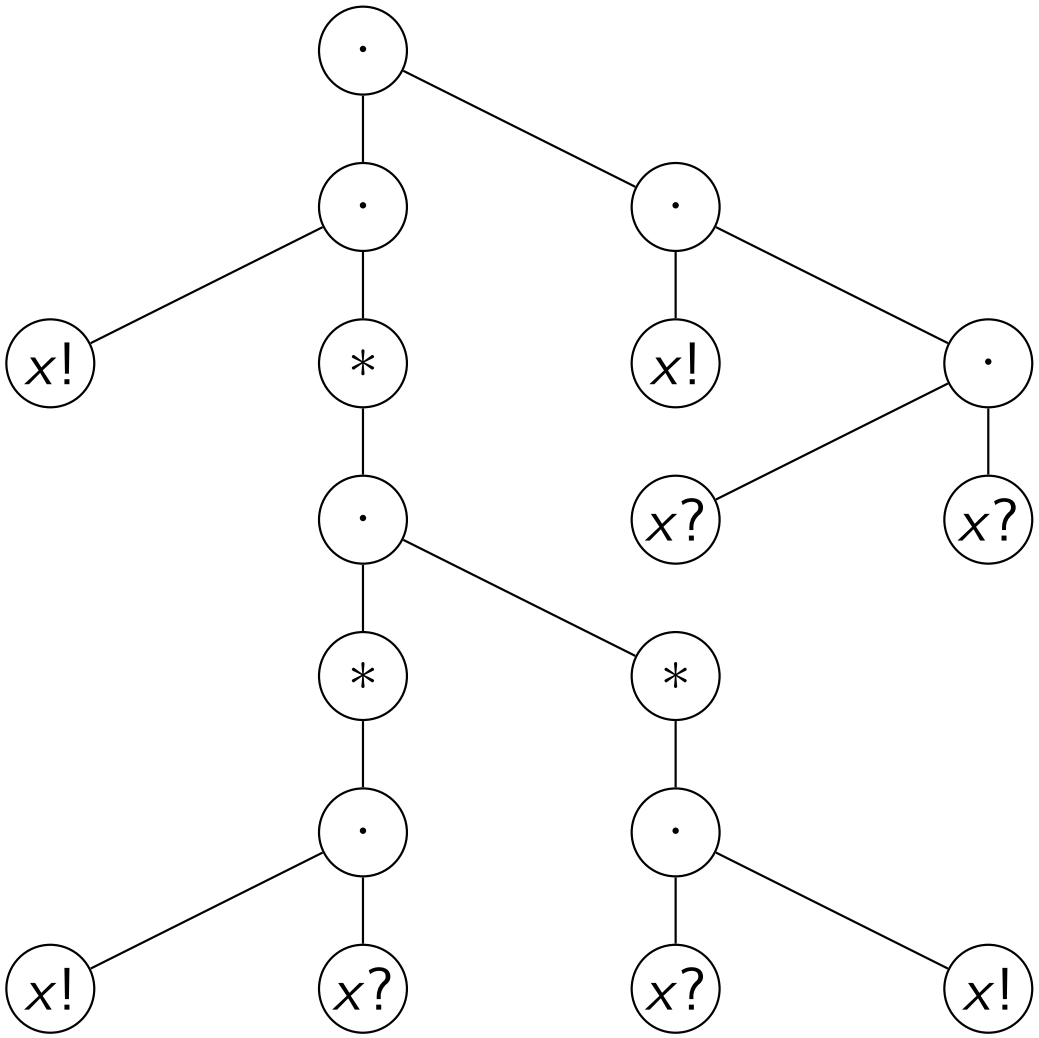
Constructing safe expressions!

$$
\begin{aligned}
& x!(x!x? + x?x!)^* x!x?(x? + x?x!x?) \\
= \; & x!((x!x?)^*(x?x!)^*)^* x!x?(x? + x?x!x?) \\
= \; & x!((x!x?)^*(x?x!)^*)^* x!x?x? \\
& + x!((x!x?)^*(x?x!)^*)^* x!x?x?x!x?
\end{aligned}
$$

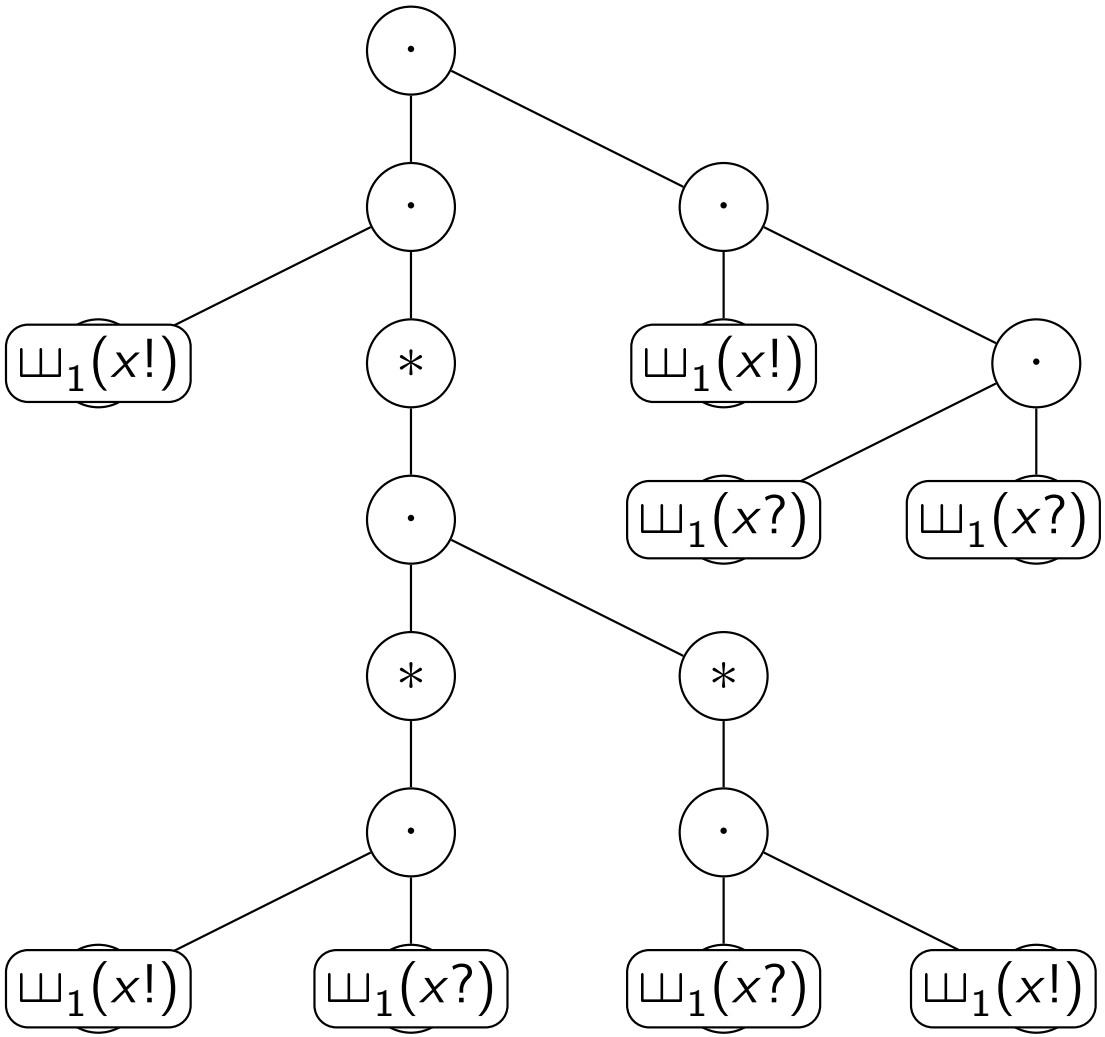
Constructing safe expressions!

$$
\begin{aligned}
& x!(x!x? + x?x!)^* x!x?(x? + x?x!x?) \\
= \; & x!((x!x?)^*(x?x!)^*)^* x!x?(x? + x?x!x?) \\
= \; & \boxed{x!((x!x?)^*(x?x!)^*)^* x!x?x?} \\
& + x!((x!x?)^*(x?x!)^*)^* x!x?x?x!x?
\end{aligned}
$$

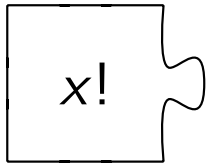
Constructing safe expressions!

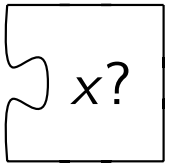


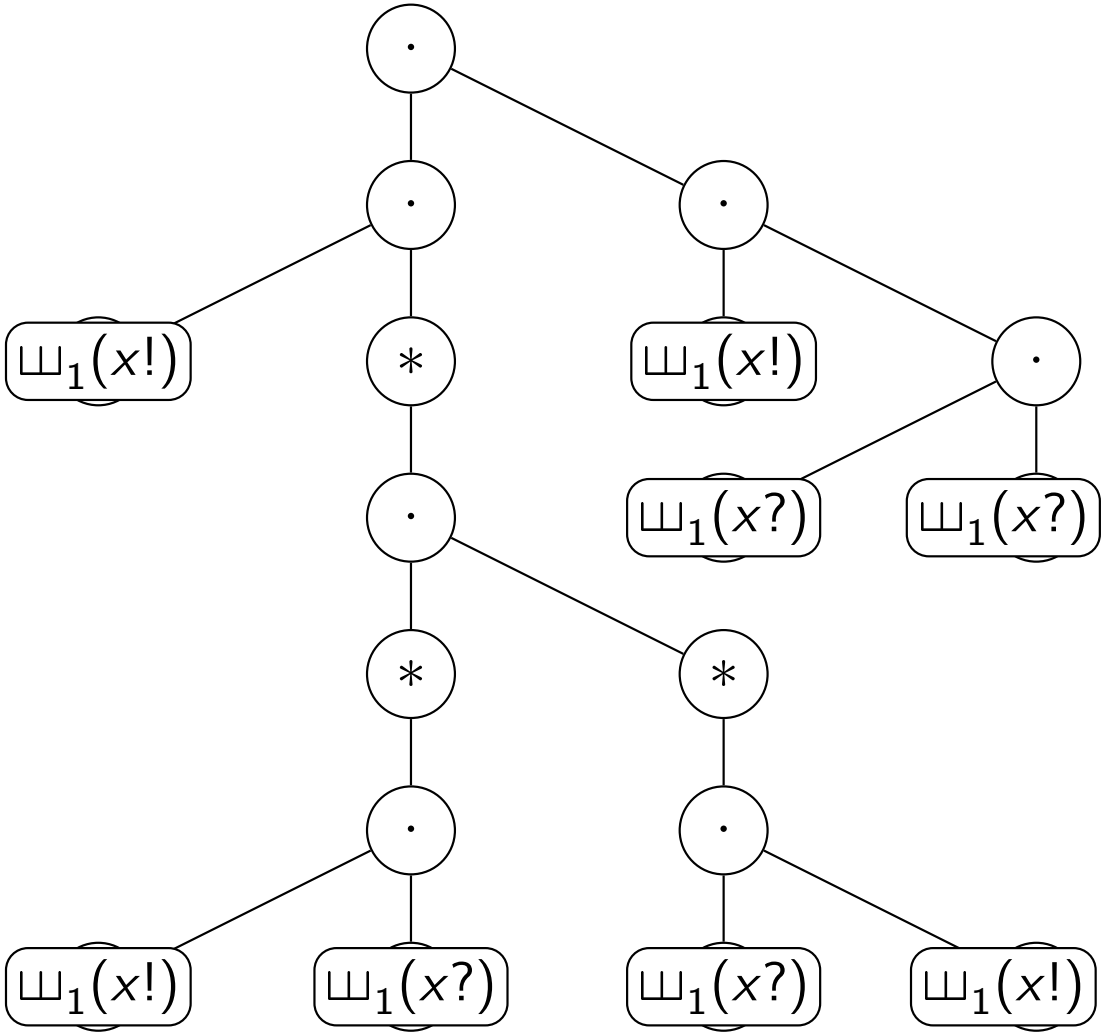
Constructing safe expressions!



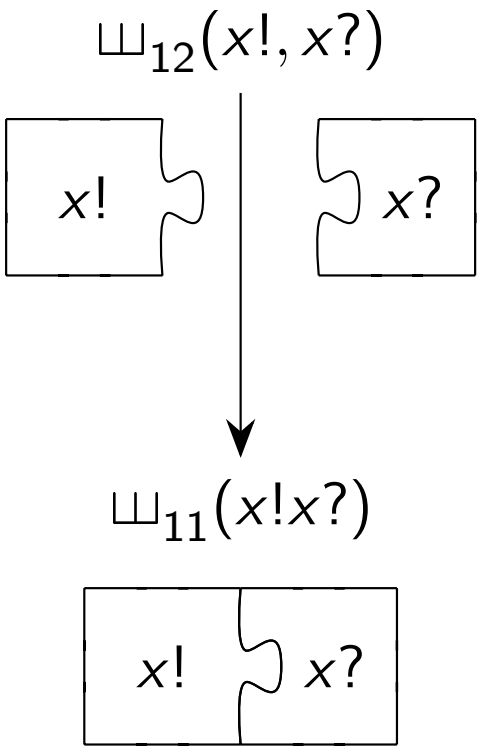
Constructing safe expressions!

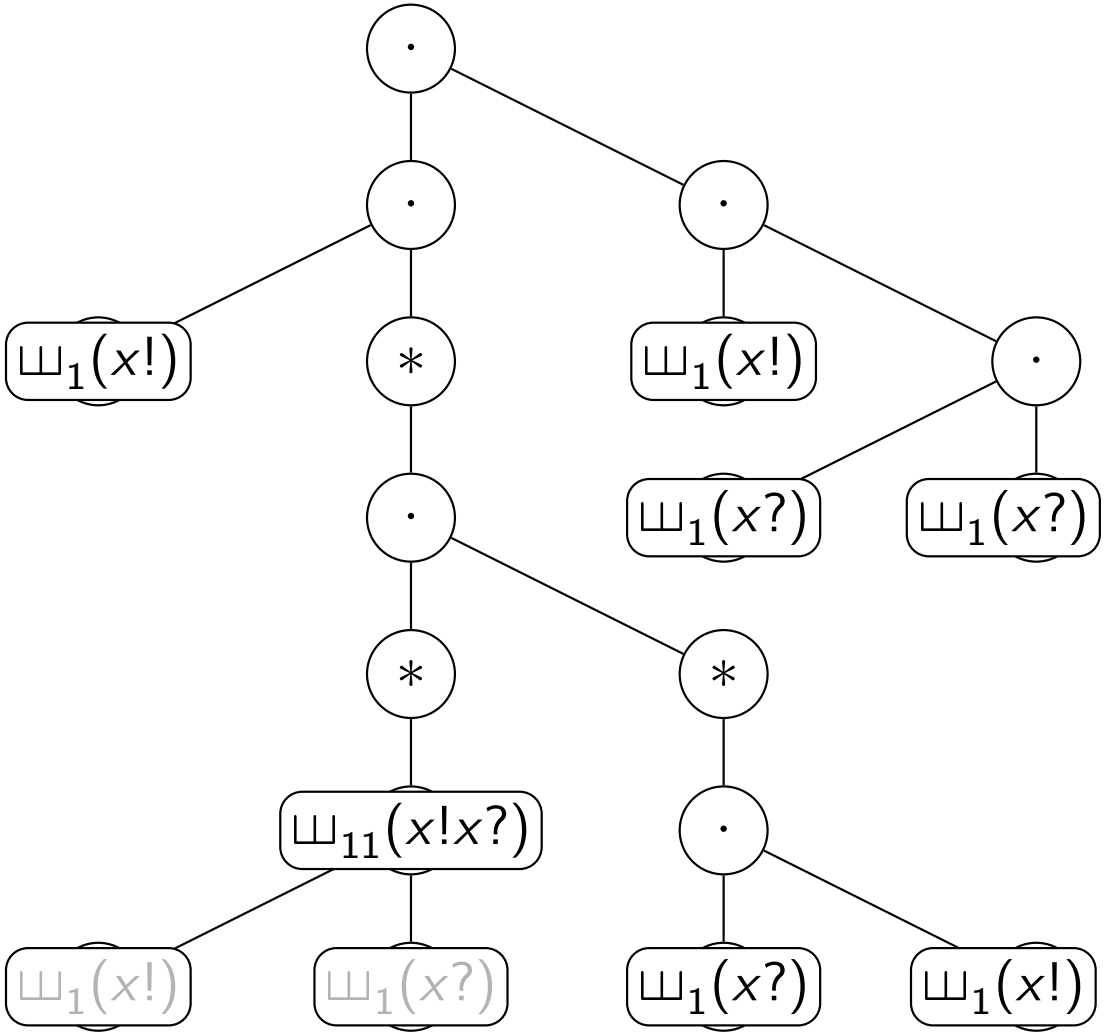




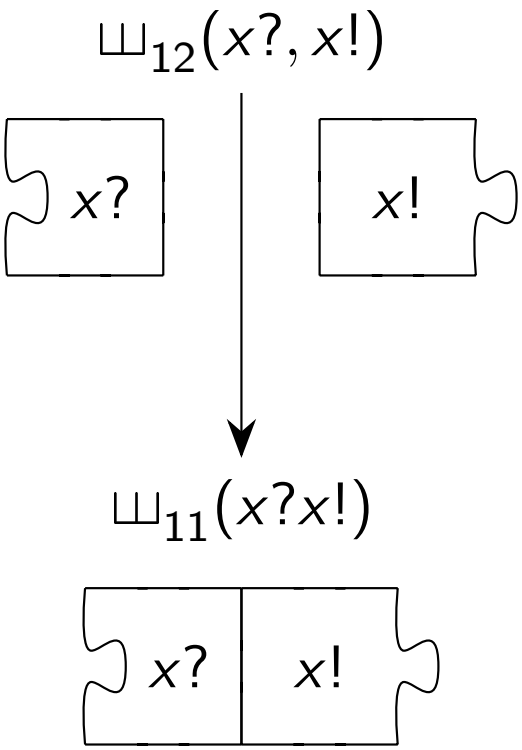


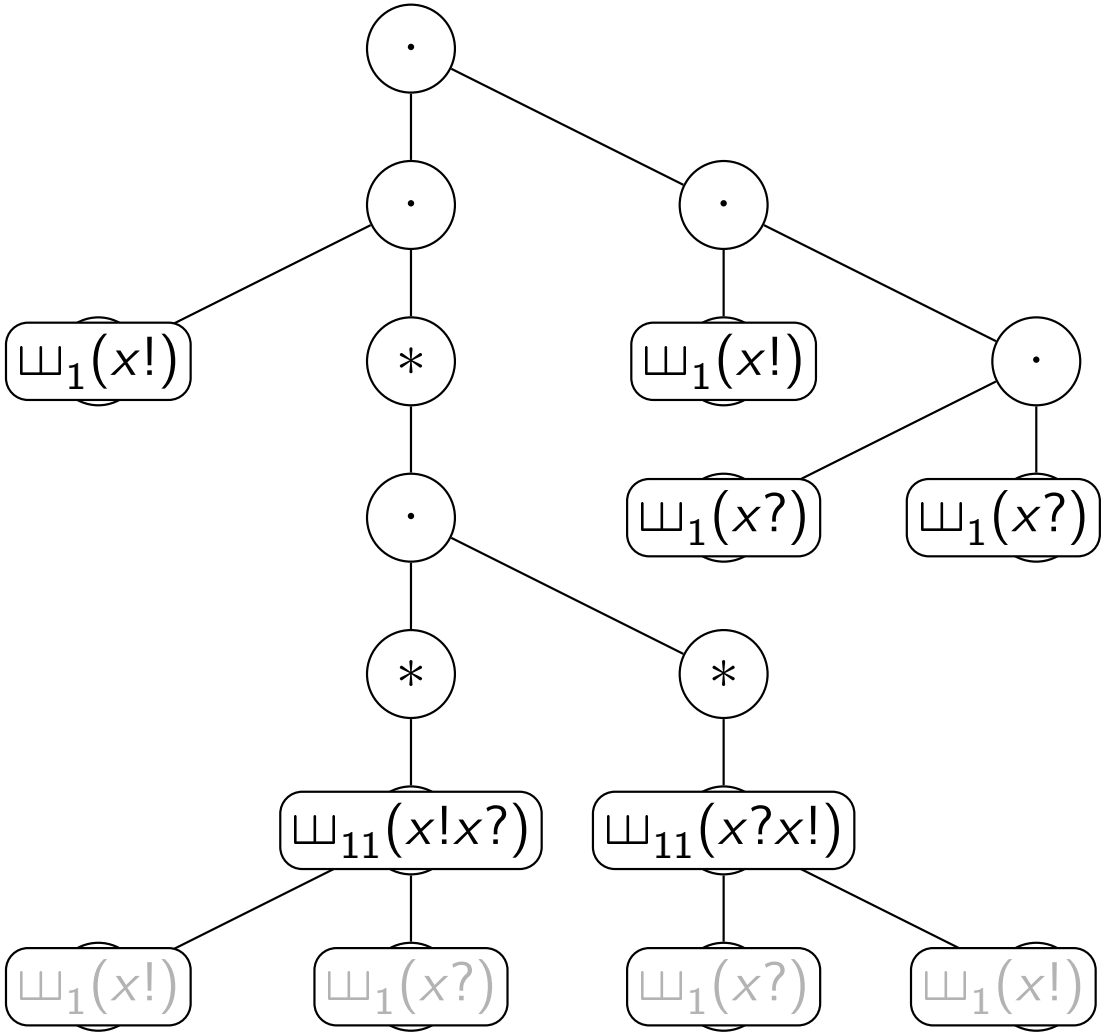
Constructing safe expressions!





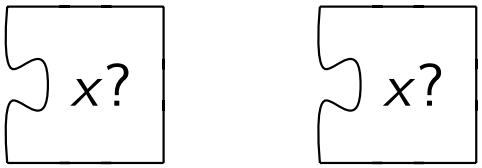
Constructing safe expressions!

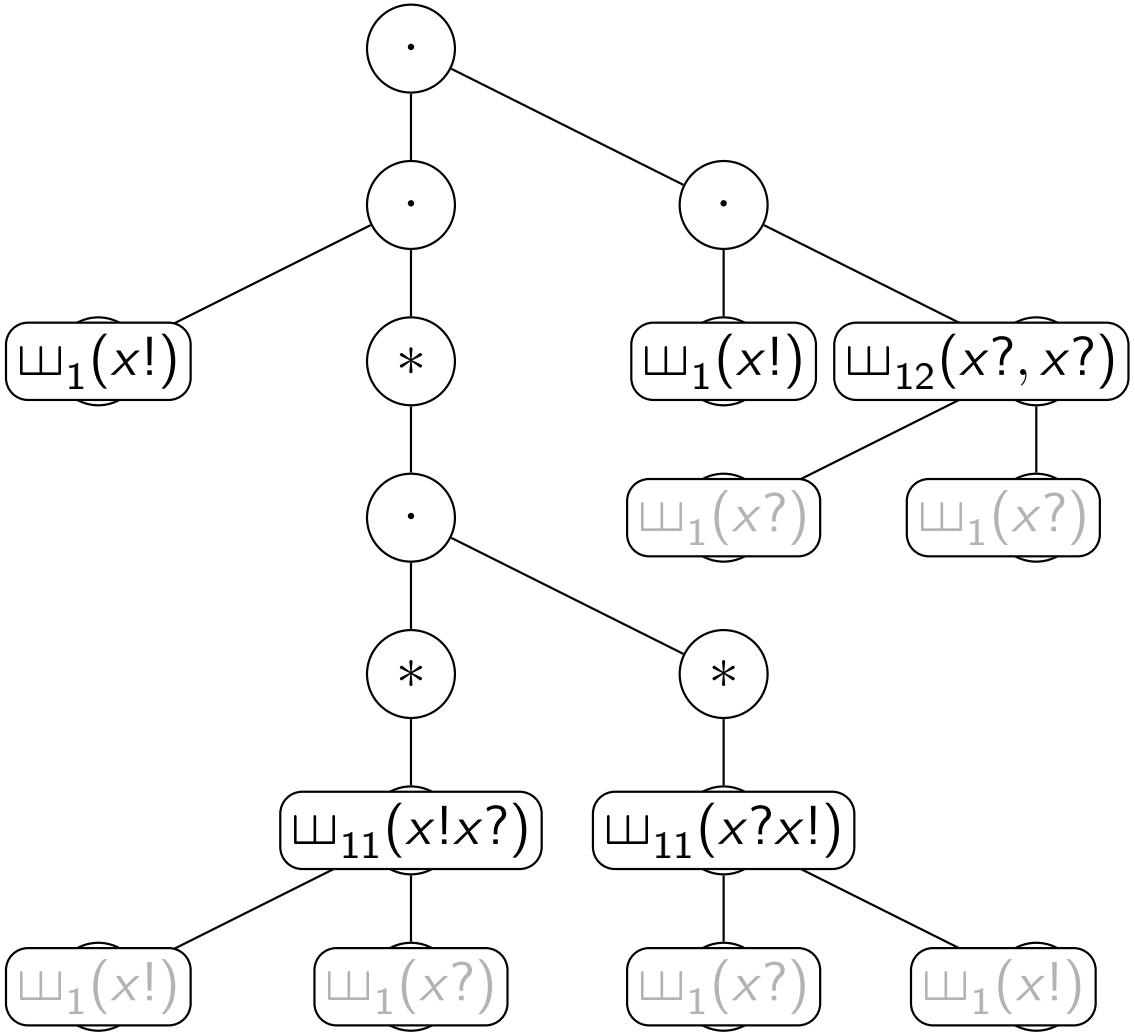




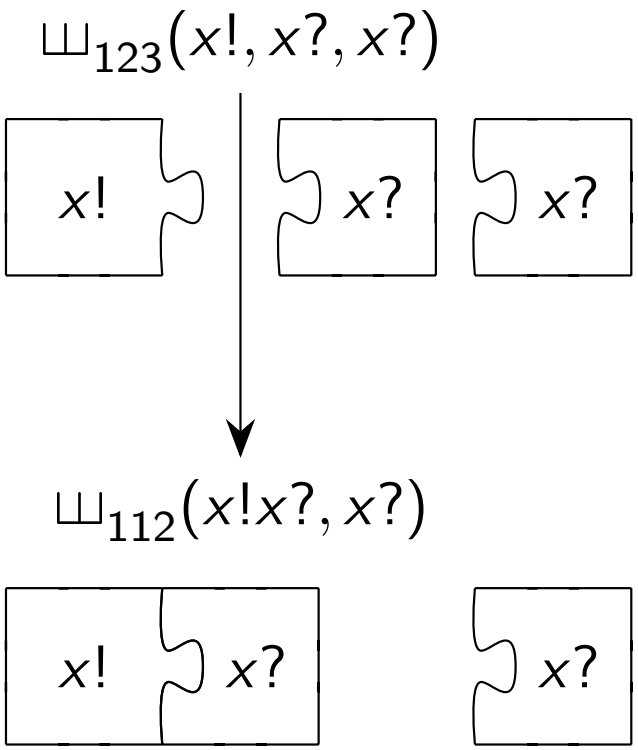
Constructing safe expressions!

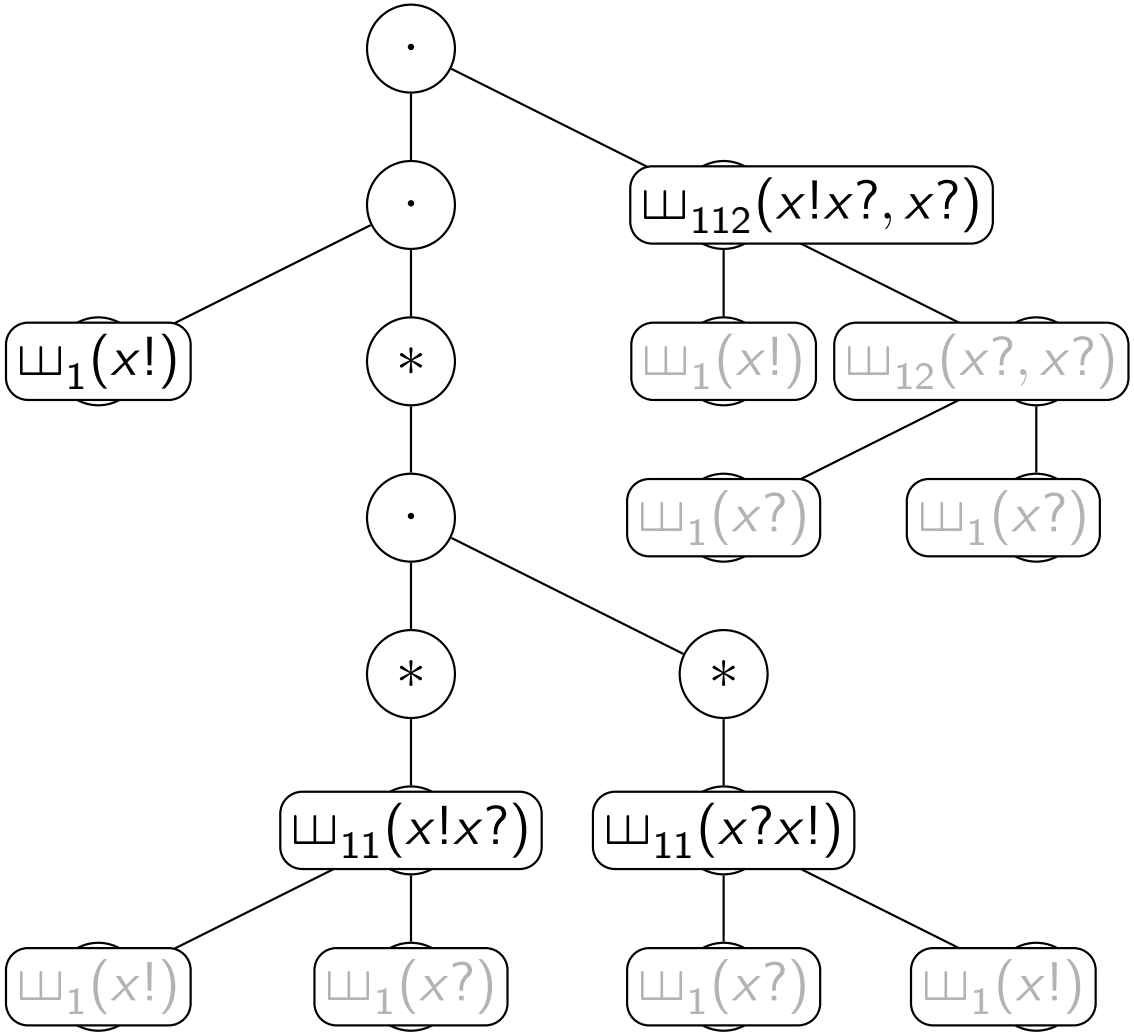
$\sqcup_{12}(x?, x?)$





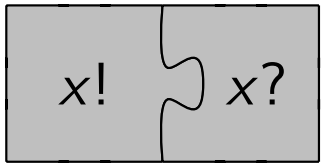
Constructing safe expressions!

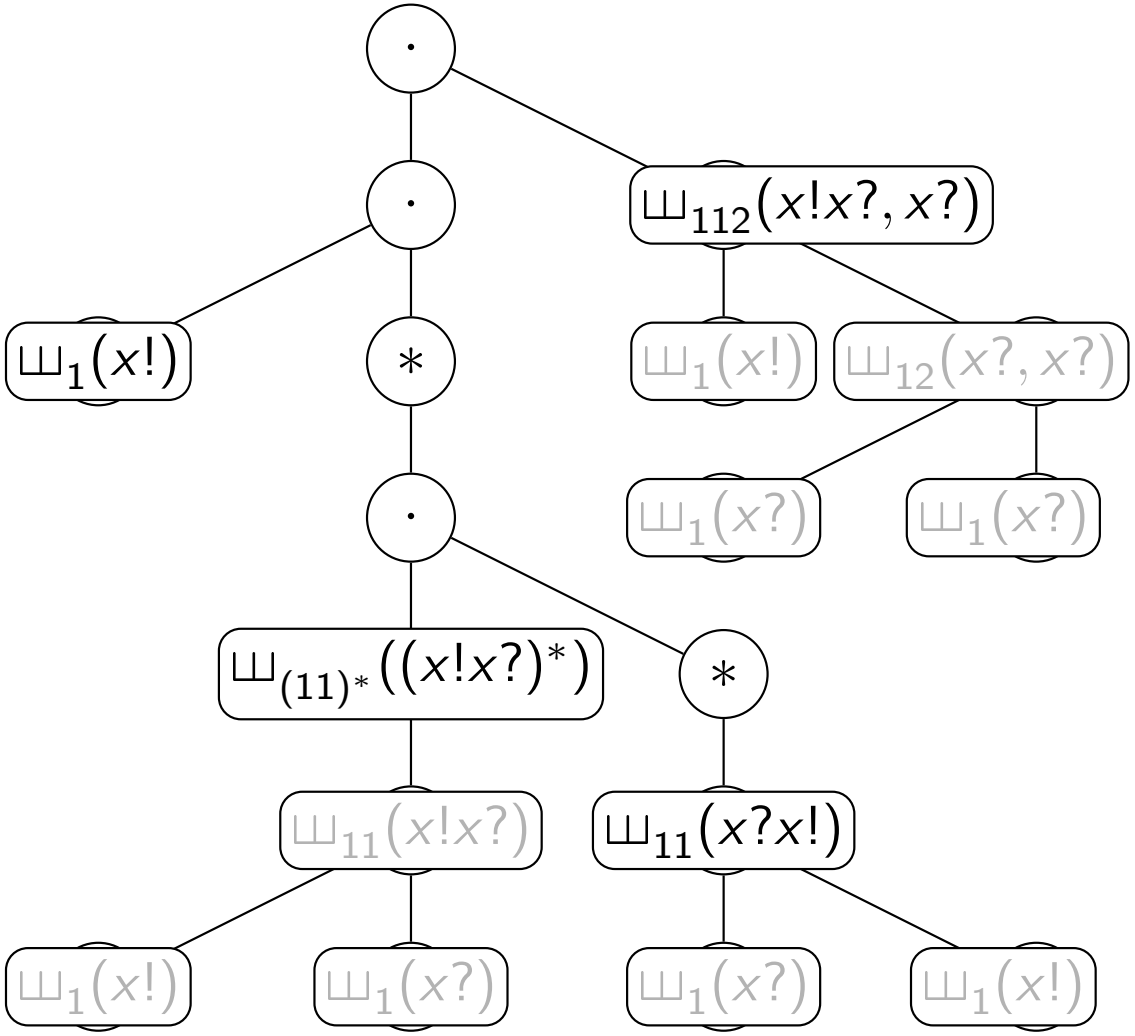




Constructing safe expressions!

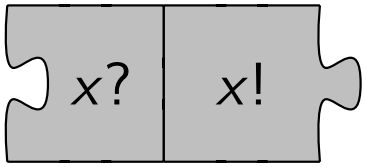
$\sqcup_{(11)^*}((x!x?)^*)$

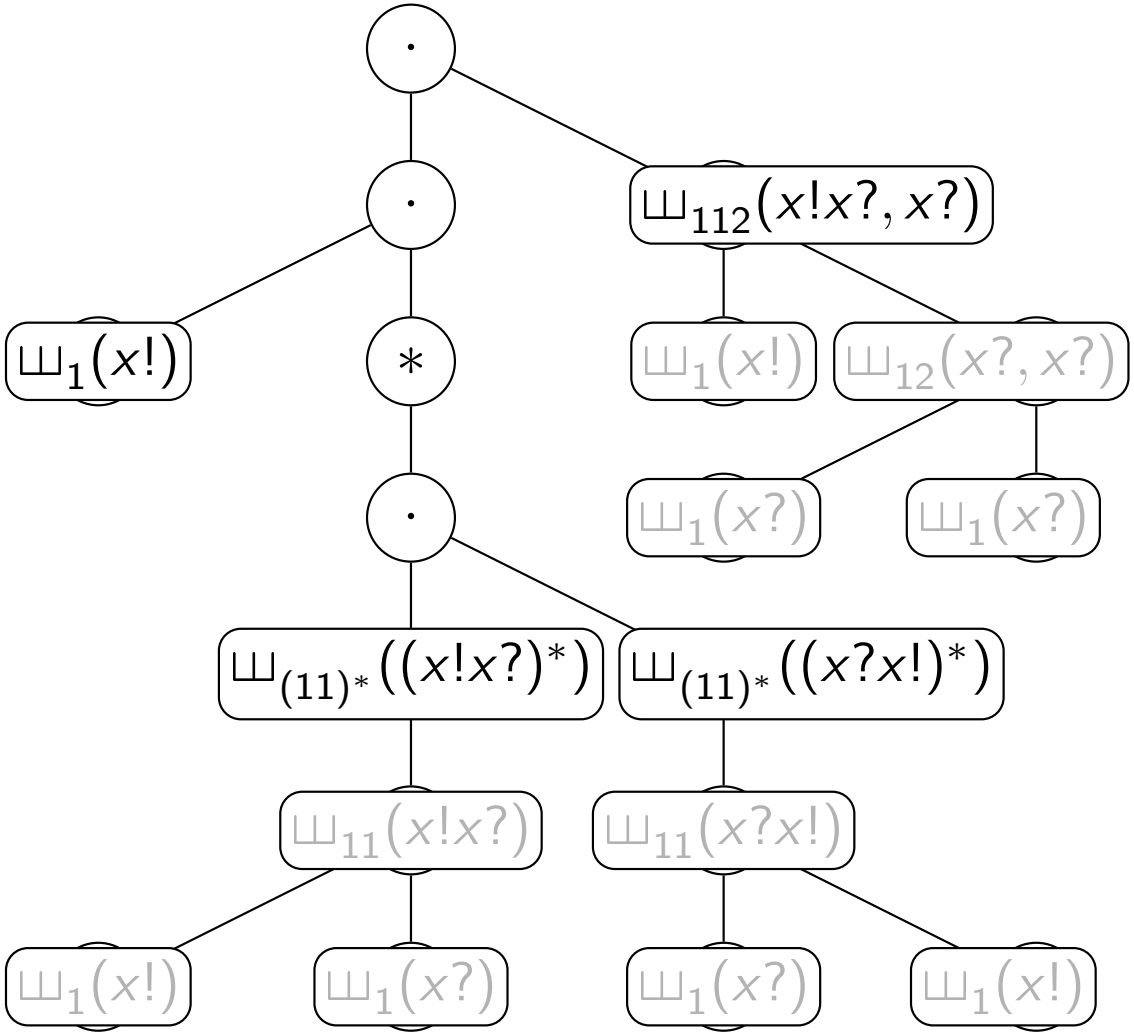




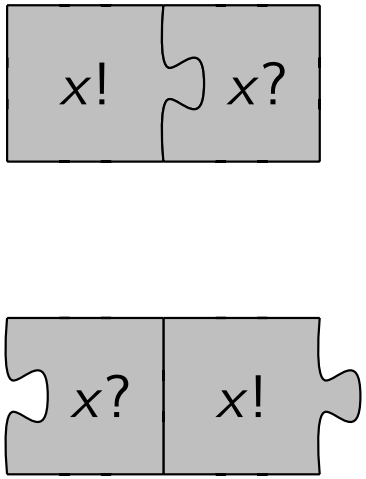
Constructing safe expressions!

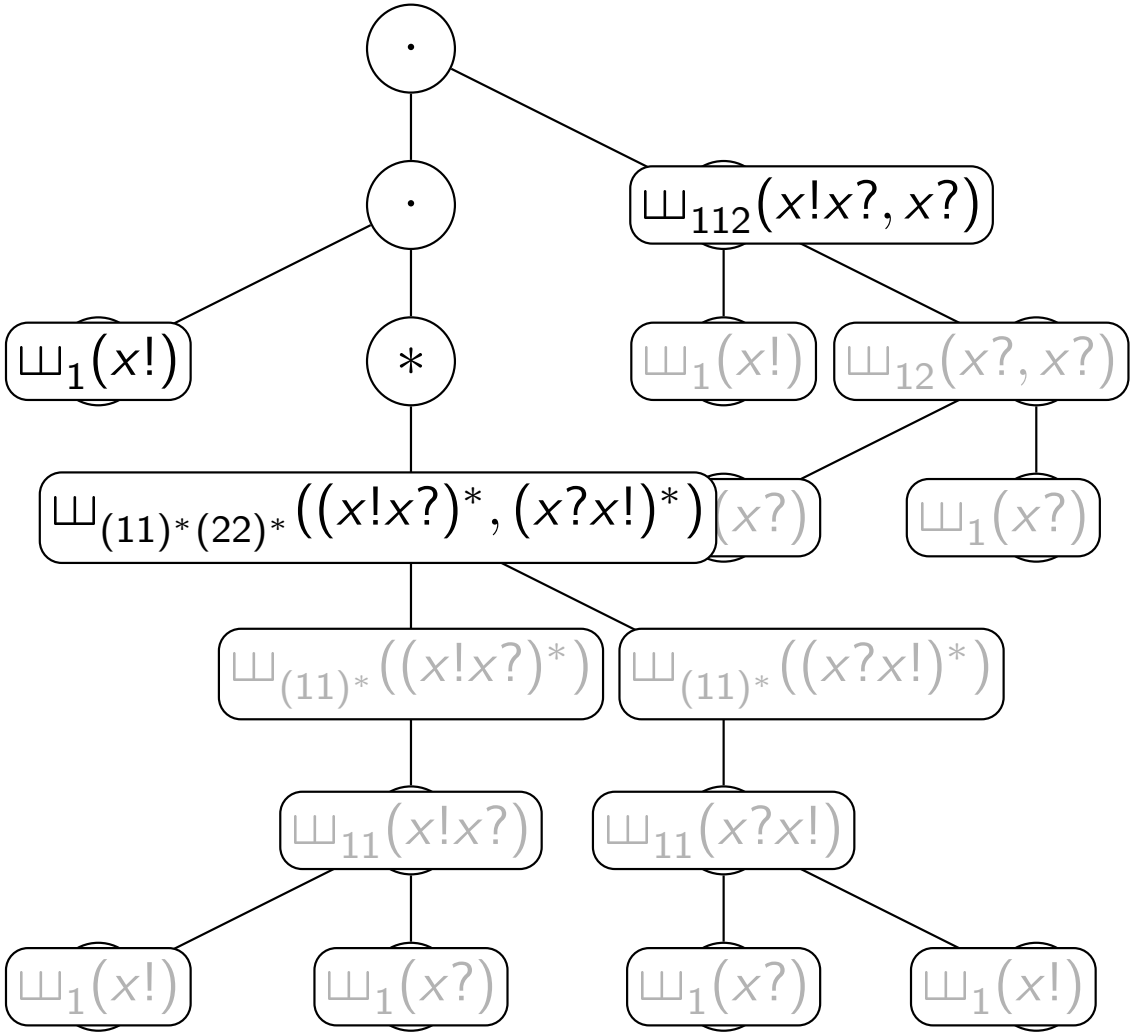
$\sqcup_{(11)^*}((x?x!)^*)$



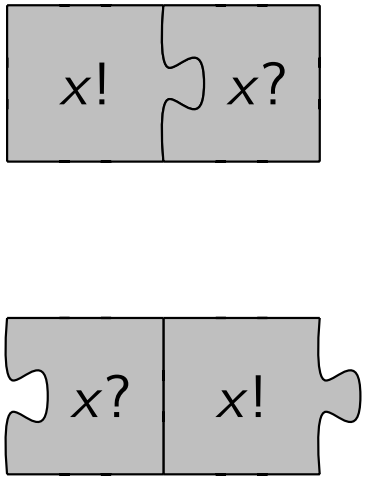


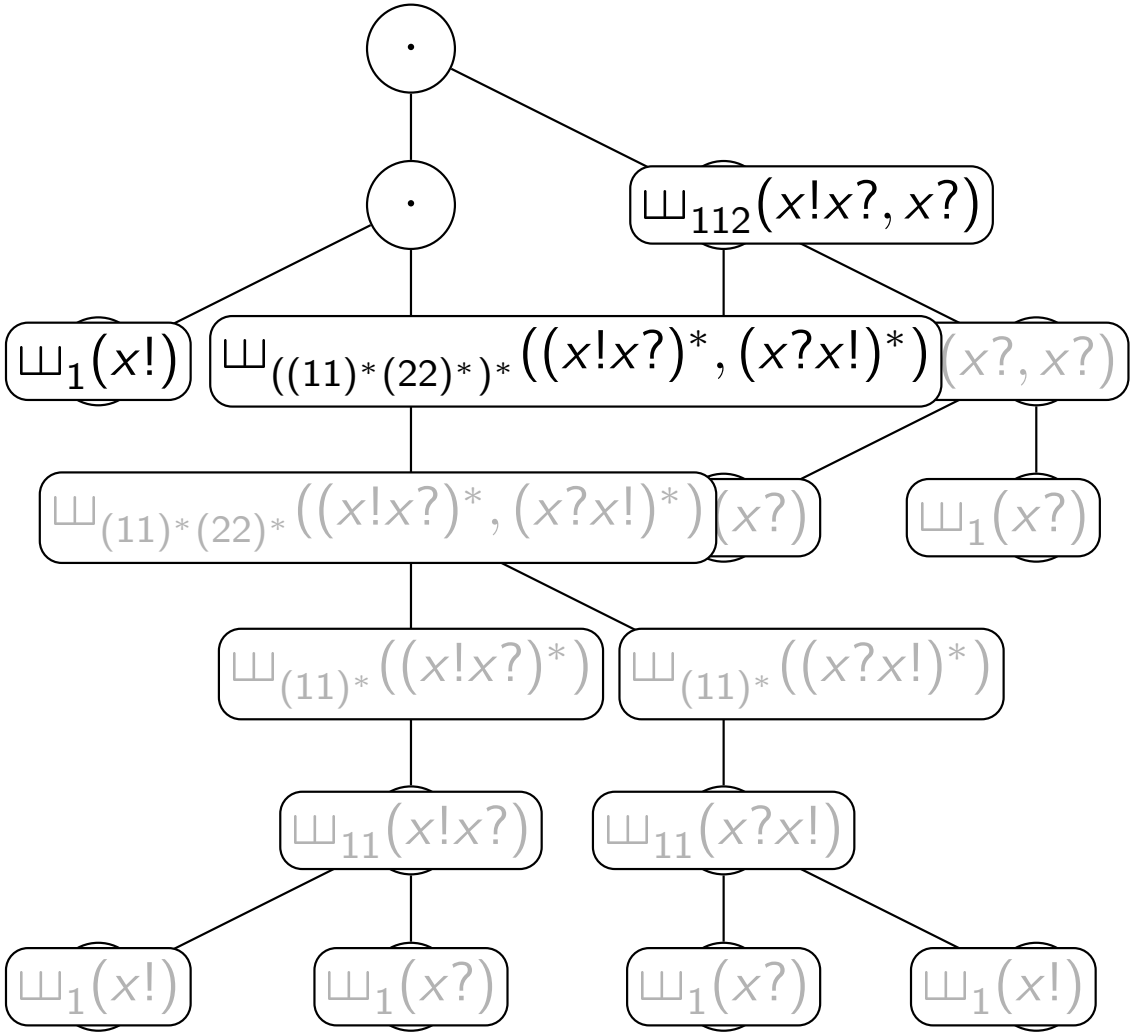
Constructing safe expressions!



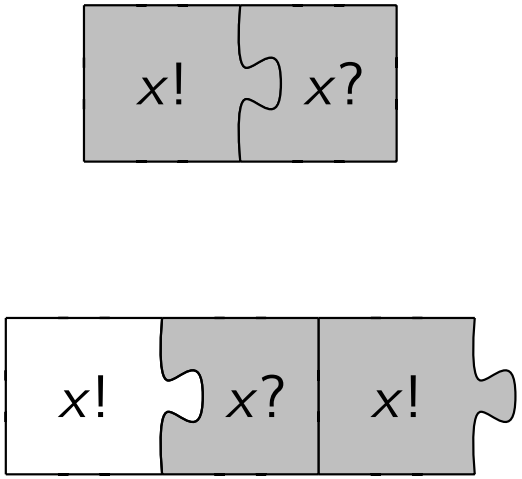


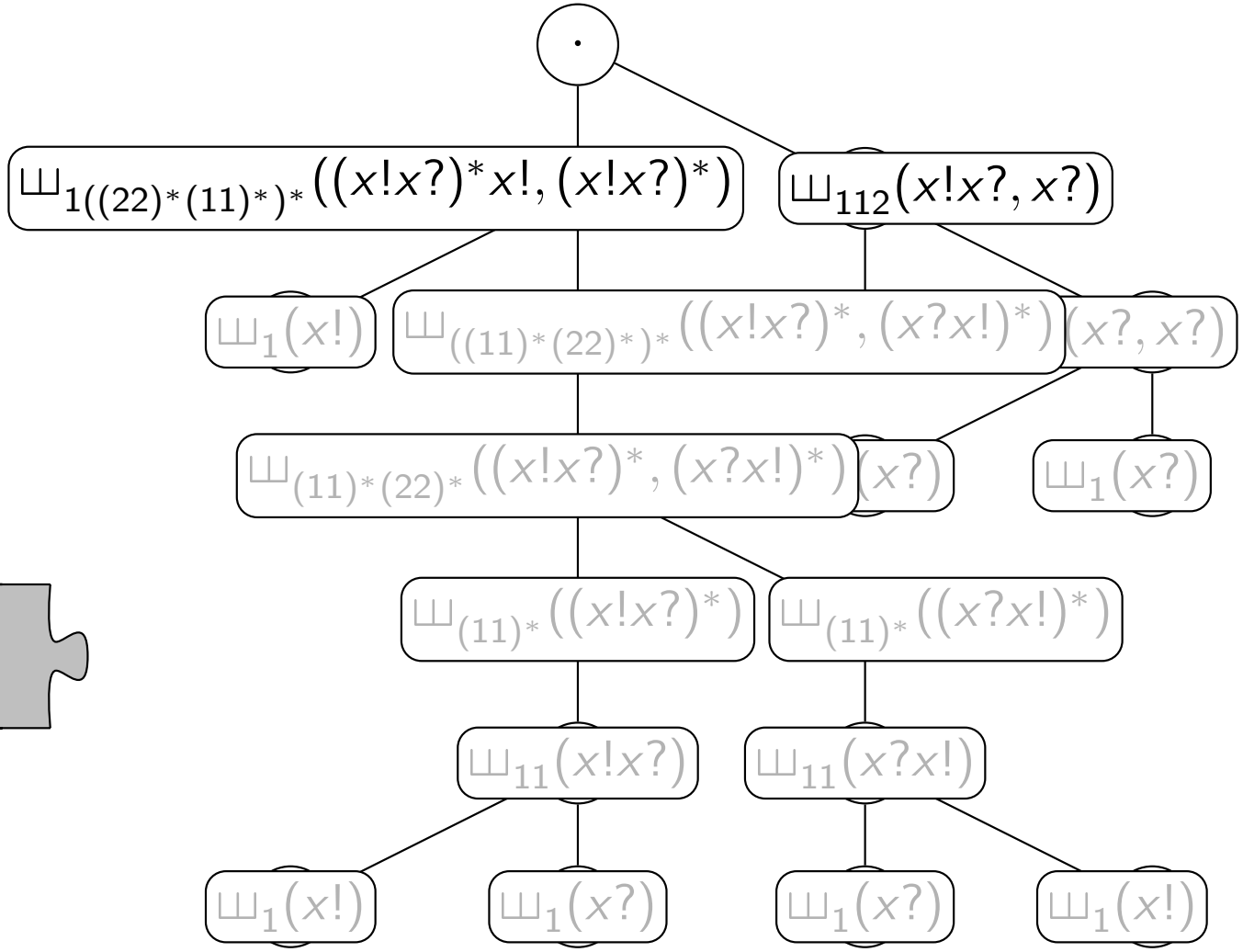
Constructing safe expressions!



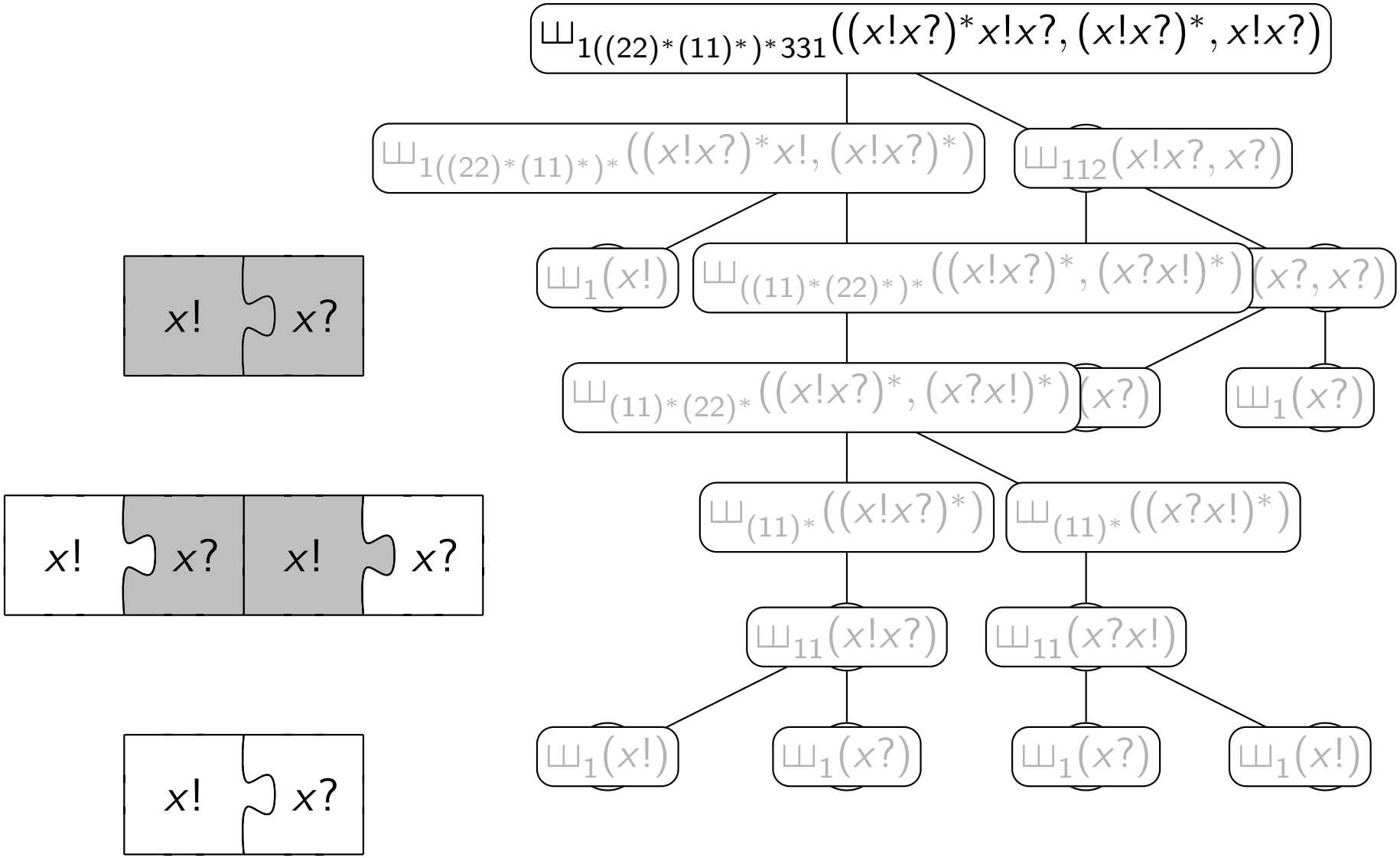


Constructing safe expressions!





Constructing safe expressions!



Beyond regular languages

- ω-regular: ✓
- Context-free languages: not yet
- Message data types
- Realisability

That's all, folks!