Introducing Quantification into a Hierarchical Graph Rewriting Language

Haruto MISHINA, Kazunori UEDA

Waseda University

LOPSTR 2024 @ Milan, Italy September 10, 2024

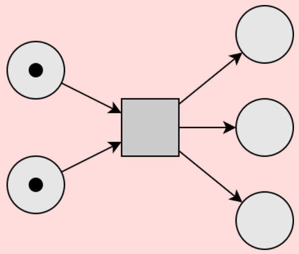
Overview

Graph rewriting languages can model diverse structures in the real world.

Problem

A challenge towards an expressive graph rewriting language is to provide its syntax and semantics with the ability to handle quantities of graph elements.

Petri Nets



Repotting all the geraniums of broken pots [1]

Contribution

We introduced into LMNtal (a hierarchical graph rewriting language) quantification for matching and rewriting that support

- ✓ Cardinality
- ✓ Non-existence
- ✓ Universal Quantification

in an integrated manner.

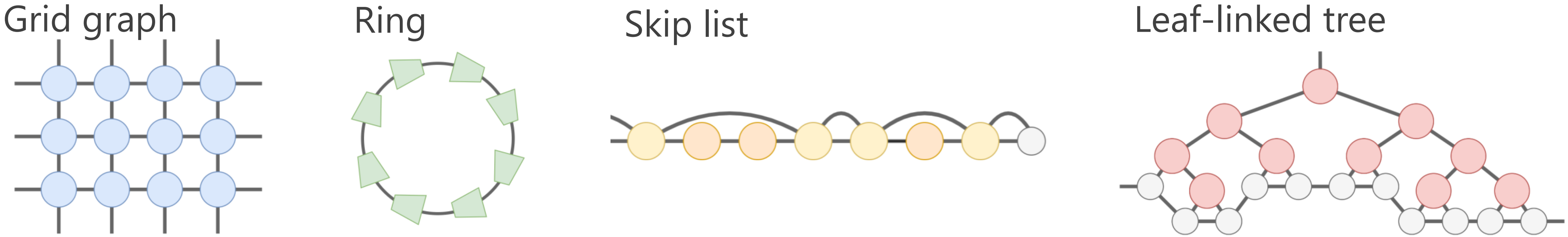
[1] Rensink, A., Kuperus, J.H.: Repotting the geraniums: On nested graph transformation rules. Electronic Communications of the EASST 18 (2009).

Contents

Contents

Graph Rewriting Languages

- express computation as rewriting of graphs,
- handle complex data structures safely and clearly, and
- can model diverse structures in the real world.

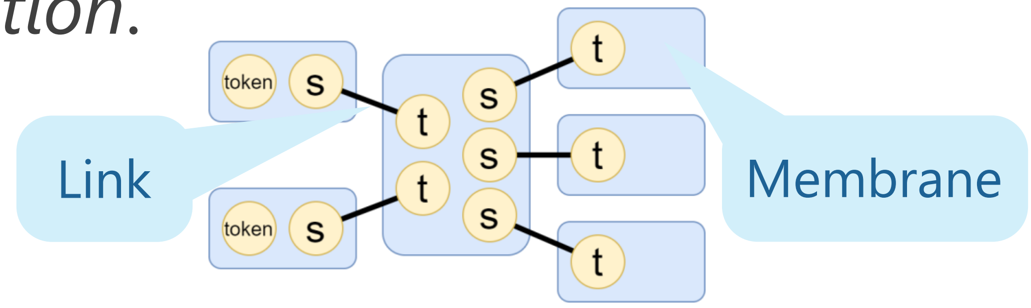


Existing tools have proposed various methods to handle "quantities" (e.g., *all* G's, *no* G's), but how to provide those features by the formal syntax and semantics of programming languages has been an open question.

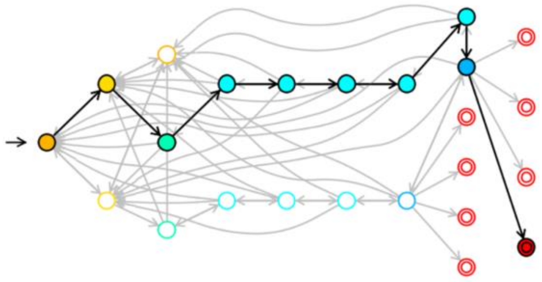
LMNtal [2] is ...

Toolchain available from https://github.com/lmntal

- a hierarchical graph rewriting language,
- suitable for modelling consisting of *connectivity* and *hierarchy*,
- based on *term-based syntax*, and the semantics consists of *structural congruence* and *(small-step) reduction relation*, and
- our implementation provides a *model checker* with *state space visualization*.



LMNtal graph



State space visualization

[2] Ueda, K.: LMNtal as a hierarchical logic programming language. Theoretical Computer Science 410(46), 4784–4800 (2009).

Syntax of LMNtal

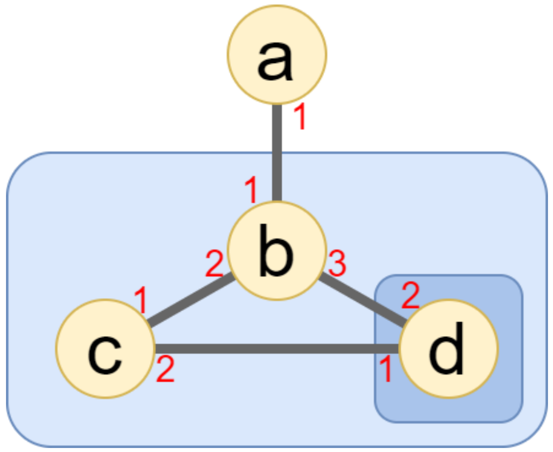
process	$P ::= \mathbf{0}$	null
	$\mid p(X_1, \ldots, X_m)$	atom
	$\mid P, P$	molucule
	$\mid \{P\}$	membrane

links

rule	$R ::= T\ \texttt{:-}\ T$	
template	$T ::= \mathbf{0}$	null
	$\mid p(X_1, \ldots, X_m)$	atom
	$\mid T, T$	molucule
	$\mid \{T\}$	membrane
	$\mid \$p$	context

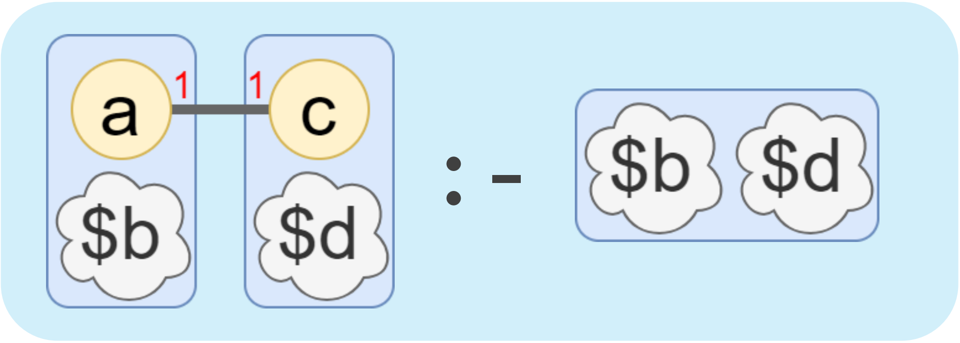
Wildcard of Processes

`a(W),{b(W,X,Y),c(X,Z),{d(Z,Y)}}`



LMNtal Graph (Process)

`{a(X),$b},{c(X),$d} :- {$b,$d}`



LMNtal Rewrite Rule

Structural Congruence of LMNtal

Structurally congruent LMNtal terms represent the same graph.

Characterizing atoms as multisets

(E1)	$0, P$	$\equiv$	P
(E2)	P, Q	$\equiv$	Q, P
(E3)	$P, (Q, R)$	$\equiv$	$(P, Q), R$

α-conversion of local link names

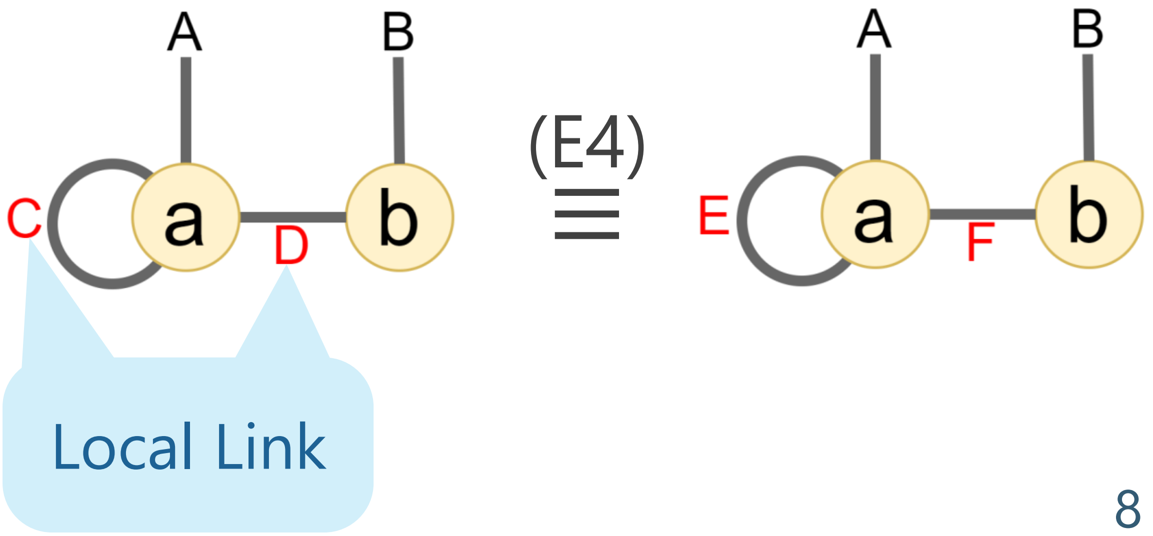
(E4)	P	$\equiv$	$P[Y/X]$ (if X is a local link of P)

Structural rules

(E5)	$P \equiv P'$	$\Rightarrow$	$P, Q \equiv P', Q$
(E6)	$P \equiv P'$	$\Rightarrow$	$\{P\} \equiv \{P'\}$

`a(A,C,C,D),b(B,D)` `b(B,D),a(A,C,C,D)`

`a(A,C,C,D),b(B,D)` `a(A,E,E,F),b(B,F)`



Reduction Relation of LMNtal

This is called the small-step semantics.

Structural rules

$$\text{(R1)}\ \frac{P \xrightarrow{R} P'}{P,Q \xrightarrow{R} P',Q}$$

$$\text{(R3)}\ \frac{Q \equiv P \quad P \xrightarrow{R} P' \quad P' \equiv Q'}{Q \xrightarrow{R} Q'}$$

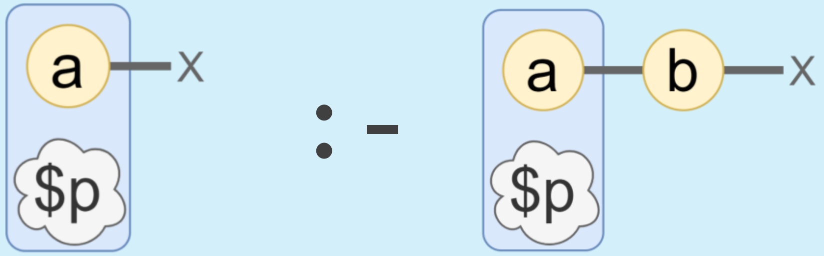
Rewriting

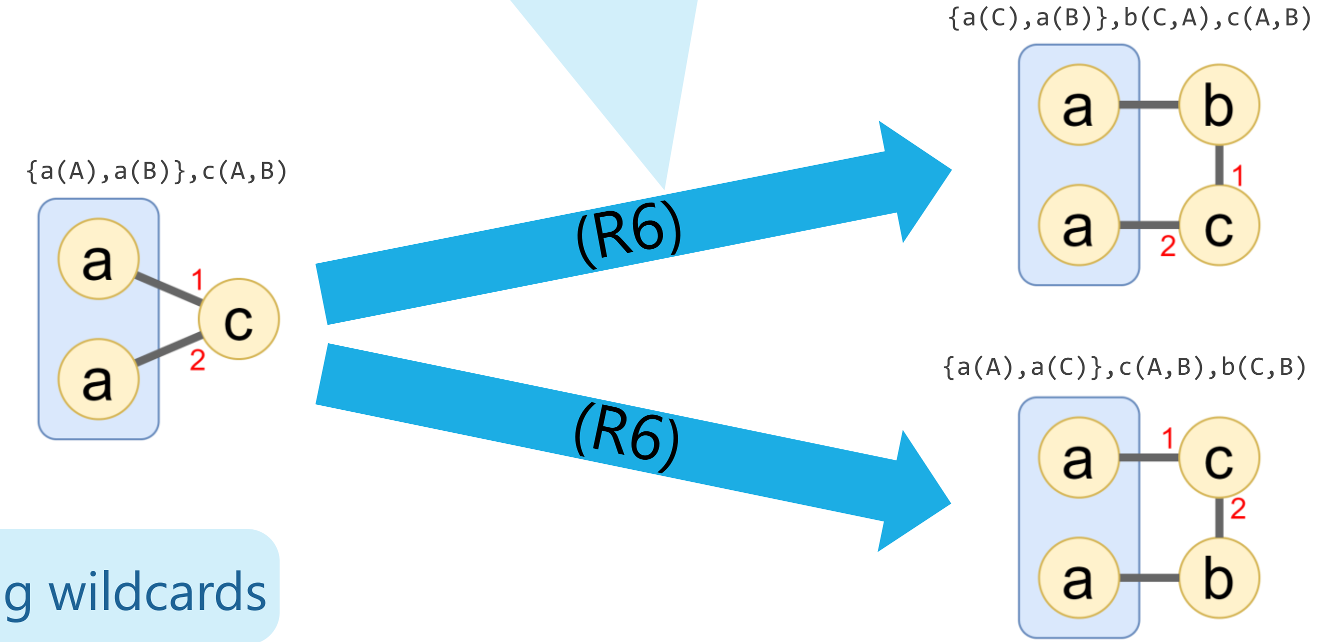
$$\text{(R6)}\ T\theta \xrightarrow{T\ \texttt{:-}\ U} U\theta$$

Instantiating wildcards

"Insert a binary **b** next to a unary **a** in a membrane."

```
{a(X),$p} :- {a(Y),$p},b(Y,X)
```





Contents

QLMNtal (LMNtal with Quantification)

To enhance the usefulness of hierarchical graph rewriting for high-level modelling, we extended LMNtal by introducing quantifiers into both matching and rewriting.

Key features of QLMNtal

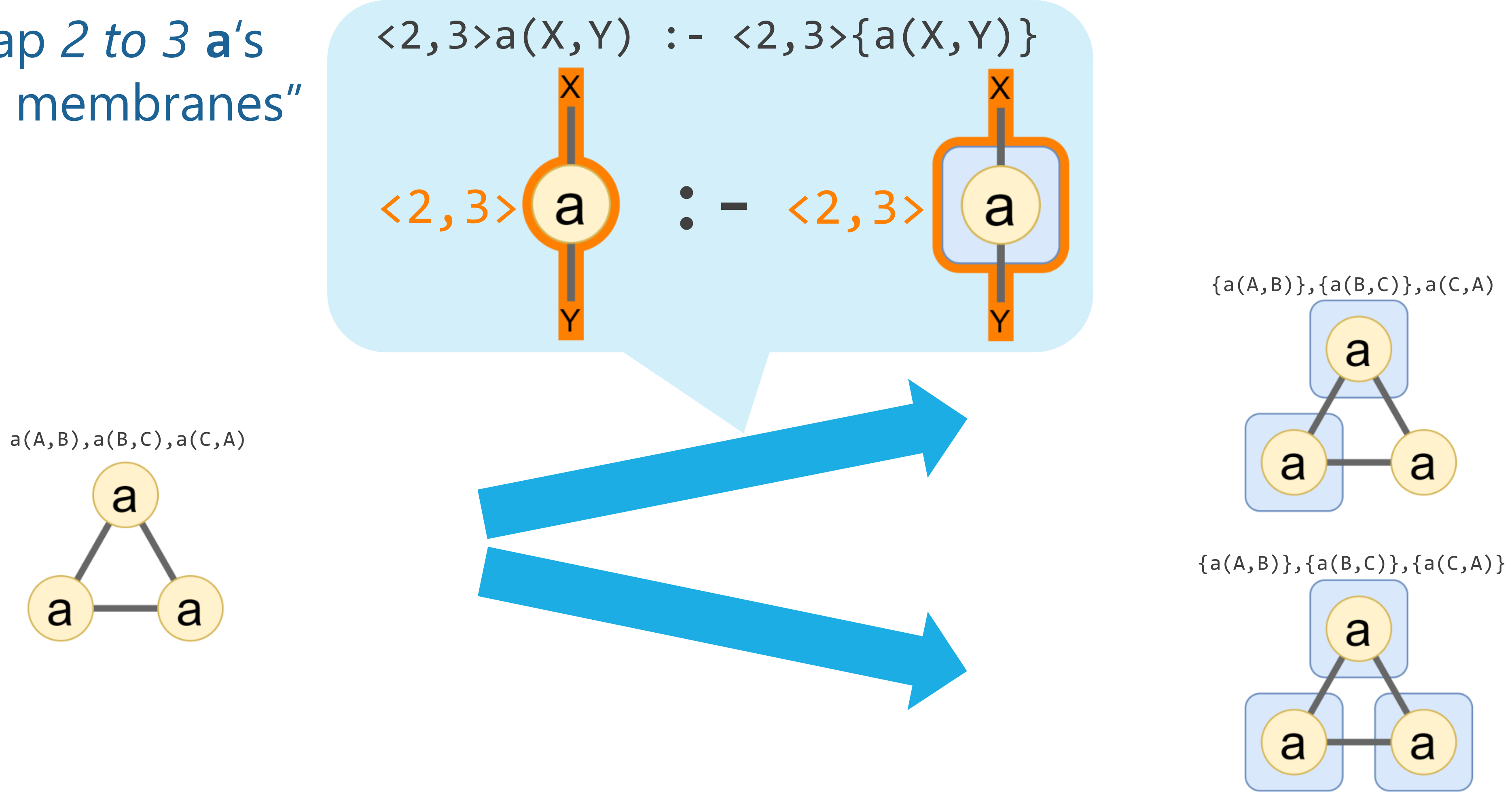
1. Introducing
    a. *cardinality*,
    b. *non-existence* (negative application condition, NAC), and
    c. *universal quantification*

    in an integrated manner
2. Relating different quantification by *labelling*
3. *Combination and nesting* of quantification

1a : Cardinality Quantification

Specifies the minimum and the maximum numbers of processes and rewrite them in a single step.

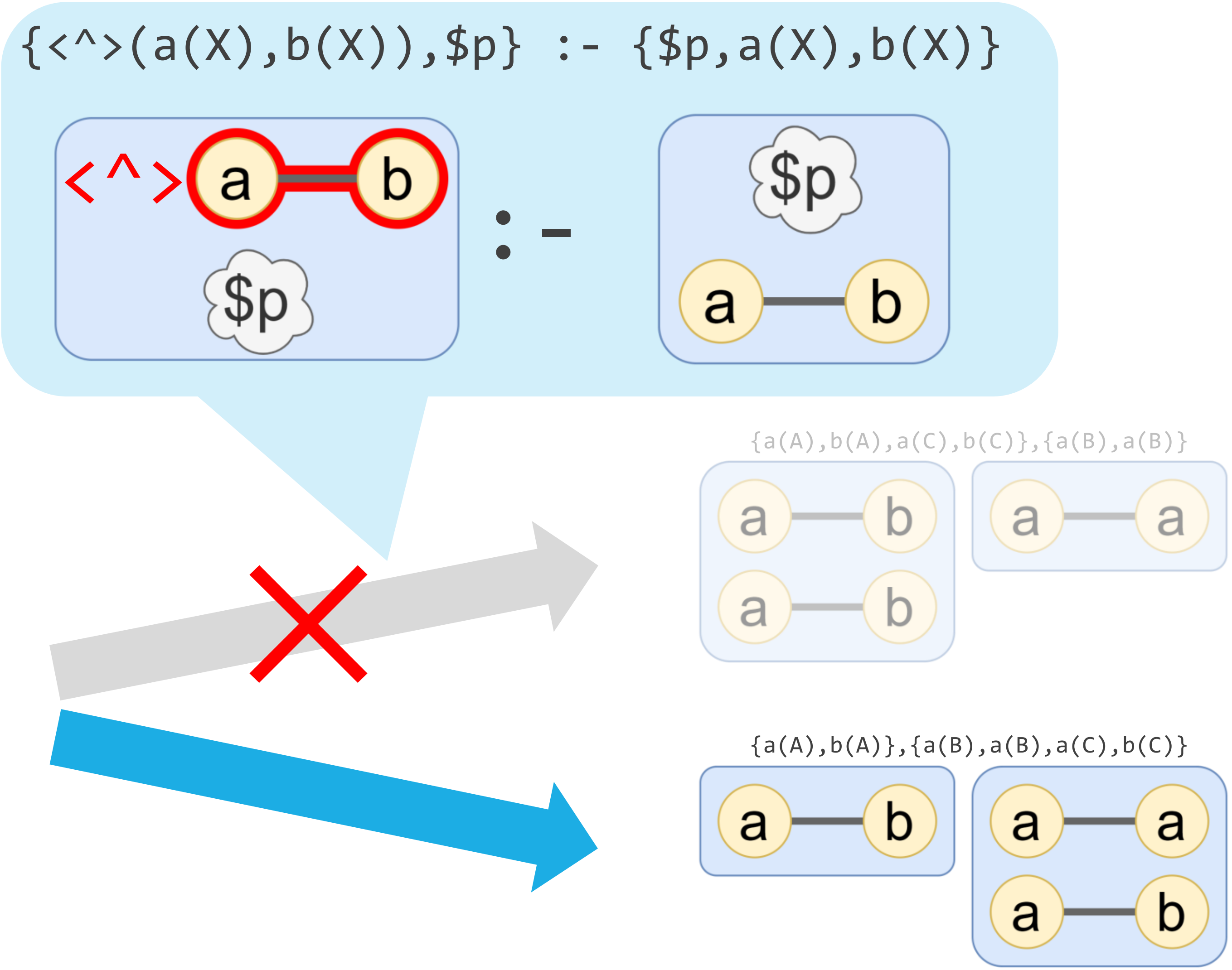
"Wrap *2 to 3* **a**'s with membranes"

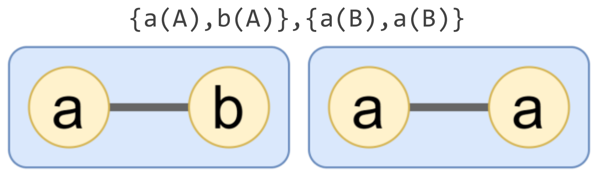


1b : Non-existence Quantification

Ensures that specified processes don't exist.

"Add connected **a** and **b** to a membrane that does *not* contain them."



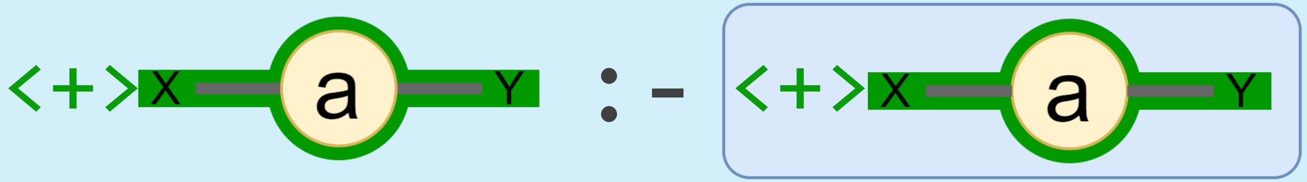


1c : Universal Quantification

Finds all specified process greedily and rewrite them in a single step.

"Wrap *all* binary **a**'s with a membrane."

```
<+>a(X,Y) :- {<+>a(X,Y)}
```




```
a(A,B),b(B,C),a(C,D),a(D,E),b(E,A)
```

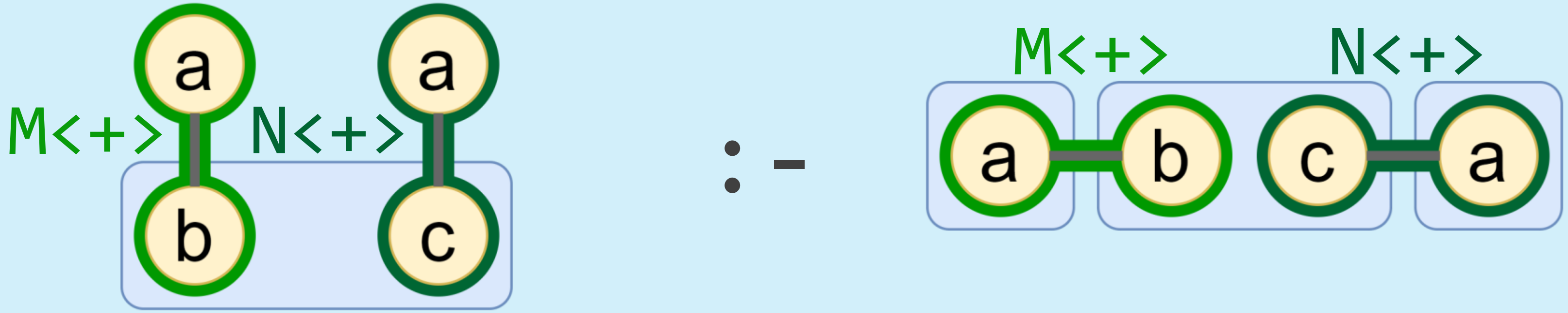
```
{a(A,B),a(C,D),a(D,E)},b(B,C),b(E,A)
```

2 : Relating Different Quantification by Labelling

Labels control the (in)dependence of quantification.

"Wrap *all* **a**'s connected to **b**'s with a membrane and
all **a**'s connected to **c**'s with another membrane."

```
M<+>a(X),N<+>a(Y),{M<+>b(X),N<+>c(Y)} :- {M<+>a(X)},{N<+>a(Y)},{M<+>b(X),N<+>c(Y)}
```

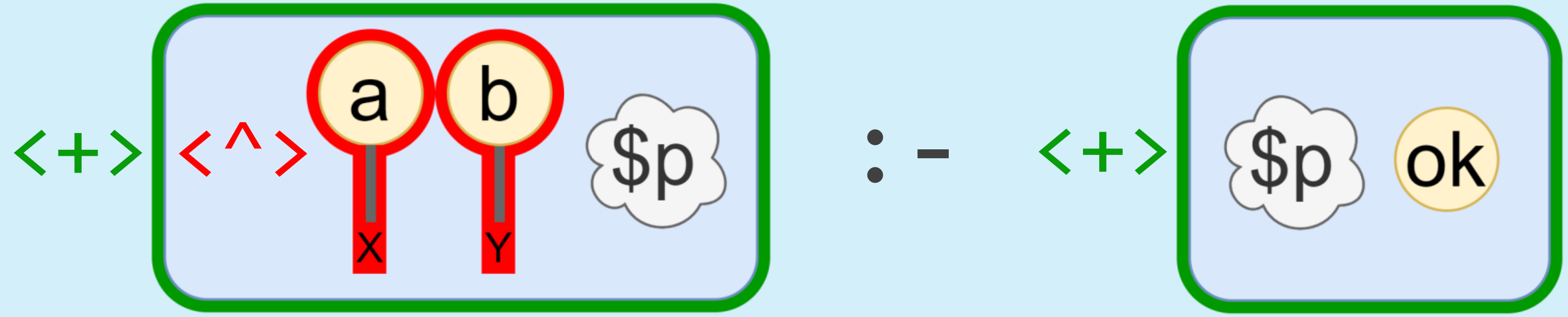


3 : Combination and Nesting of Quantification

The approach based on structural operational semantics allows combined and nested use of quantification without restrictions.

“Add **ok** to *all* membranes that do *not* contain *both* **a** and **b**.”

```
<+>{<^>(a(X),b(Y)),$p} :- <+>{$p,ok}
```



{a(A),a(B)},{a(B),b(C)},{b(C),b(A)}

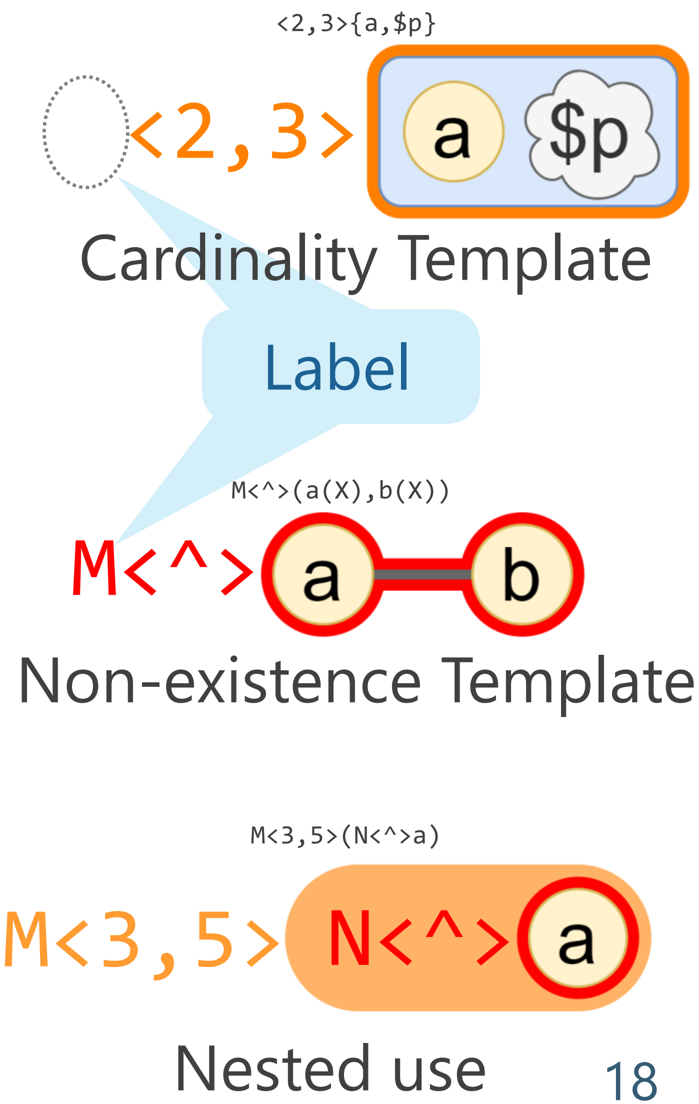
{a(A),a(B),ok},{a(B),b(C)},{b(C),b(A),ok}

Contents

Syntax of QLMNtal

We added cardinality and non-existence quantifier for templates.

rule	$R ::= T \;\texttt{:-}\; T$	
template	$T ::= \mathbf{0}$	null
	$\mid p(X_1, \ldots, X_m)$	atom
	$\mid QT$	quantified template
	$\mid T, T$	molucule
	$\mid \{T\}$	membrane
	$\mid \$p$	context
quantifier	$Q ::= l\langle z, z\rangle$	cardinality
	$\mid l\langle \hat{}\,\rangle$	non-existence



Representation of Universal Quantification

Universal quantification of QLMNtal (two versions) is not a primitive; it can be represented by combining cardinality quantification and non-existence quantification:

disallows zero T's

$$l\langle + \rangle T \;\equiv\; l\langle 1, \infty \rangle T \,,\, l\langle \wedge \rangle T'$$

allows zero T's

$$\underbrace{l\langle * \rangle T}_{\text{grabs all } T\text{'s}} \;\equiv\; \underbrace{l\langle 0, \infty \rangle T}_{\text{grabs arbitrary \# of } T\text{'s}} \,,\, \underbrace{l\langle \wedge \rangle T'}_{\text{ensures no remaining } T\text{'s}}$$

T's variant with renamed links, contexts and labels

Structural Congruence of QLMNtal

We added the following equivalence relation (EQ) :

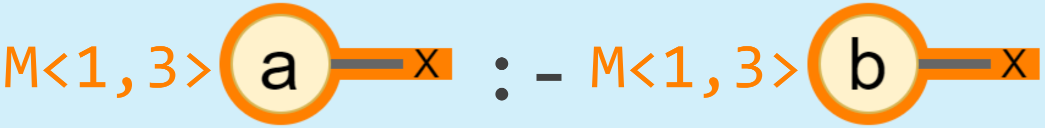
replace

$$\text{(EQ)}\ T\ \texttt{:-}\ U \equiv (T\ \texttt{:-}\ U)\left[\overrightarrow{(l\langle \underbrace{z_1-1, z_2-1}_{\text{decrement}}\rangle_0 T_i, \underbrace{T_i'}_{\text{unroll}})/l\langle z_1, z_2\rangle_0 T_i}^{\,i}\right]$$

That is, *outermost* cardinality quantified templates can be *unrolled* (or *expanded*) by decrementing the cardinalities.

"Replace 1 to 3 **a**'s with **b**'s."

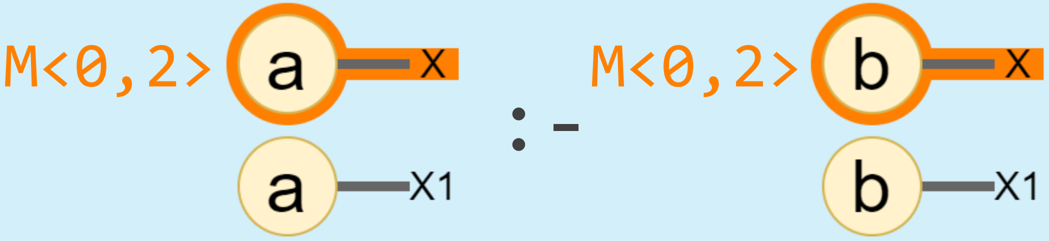
```
M<1,3>a(X) :- M<1,3>b(X)
```



$\equiv$

"Replace 0 to 2 **a**'s with **b**'s, and **a** with **b**."

```
M<0,2>a(X),a(X1) :- M<0,2>b(X),b(X1)
```



Reduction Relation of QLMNtal

We replaced the rule (R6) with the following (RQ) :

Have all top-level cardinalities been unrolled the right number of times by (EQ) ? *(Cardcond)*

Are there no processes that match templates quantified by top-level negation quantifiers? *(Negcond)*

$$\text{(RQ)}\quad \frac{\forall l\langle z_1, z_2\rangle_0 (z_1 \leq 0 \land z_2 \geq 0) \;\land\; \forall l\langle \hat{} \rangle_0 (cxt(\{T, \$\gamma\})\theta \overset{neg(l, \{T, \$\gamma\})\ :\text{-}}{\not\longrightarrow})}{(\underbrace{simp(T)}_{\text{unquantified part of } T}, \underbrace{\$\gamma}_{\text{global context of } simp(T)})\theta \xrightarrow{T\ :\text{-}\ U} (simp(U), \$\gamma)\theta}$$

unquantified part of T

global context of $simp(T)$

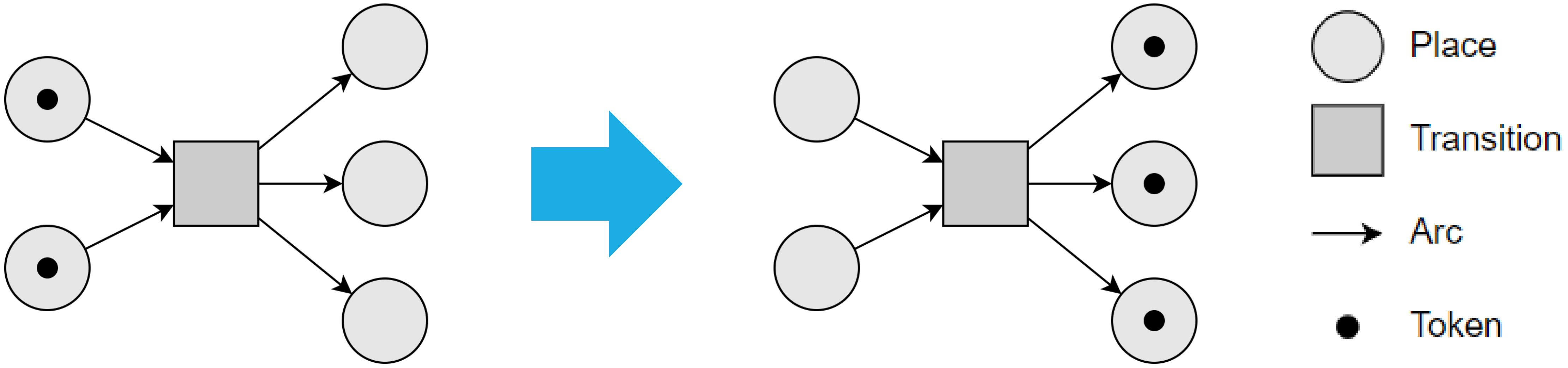
rewriting by $T :\text{-} U$ (RQ')

When *(Cardcond)* ∧ *(Negcond)* is satisfied, rewriting takes place by (RQ').

Contents

Petri Nets [3]

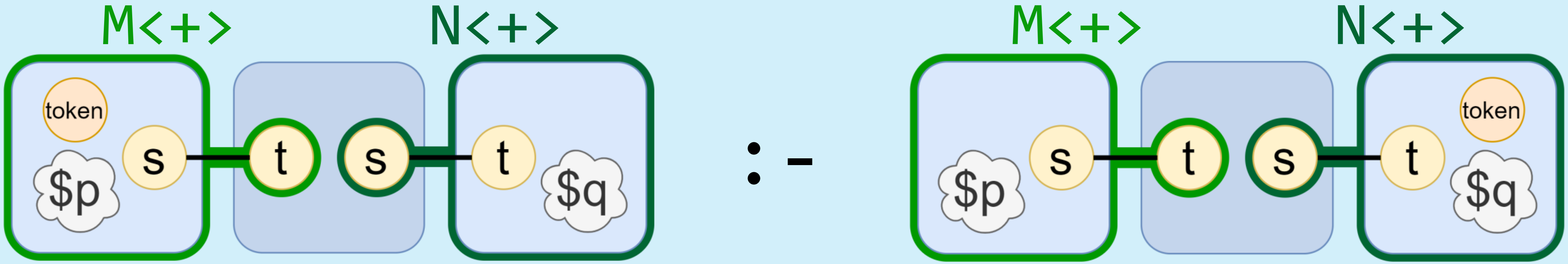
If *all* inputs of a transition contain *at least* one token, delete *one* token from *each* input place and create *one* token in *each* output place.



[3] Desel, J., Reisig, W.: Place/transition petri nets. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models: Advances in Petri Nets, pp. 122–173. Springer Berlin Heidelberg (1998).

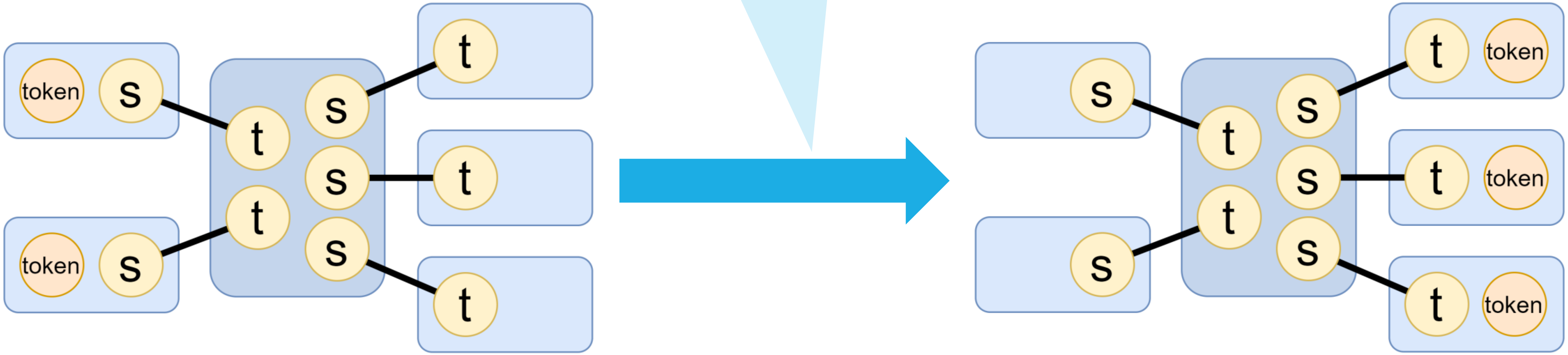
Petri Nets

```
M<+>{token,s(X),$p}, {M<+>t(X),N<+>s(Y)}, N<+>{t(Y),$q}
 :- M<+>{s(X),$p}, {M<+>t(X),N<+>s(Y)}, N<+>{token,t(Y),$q}
```



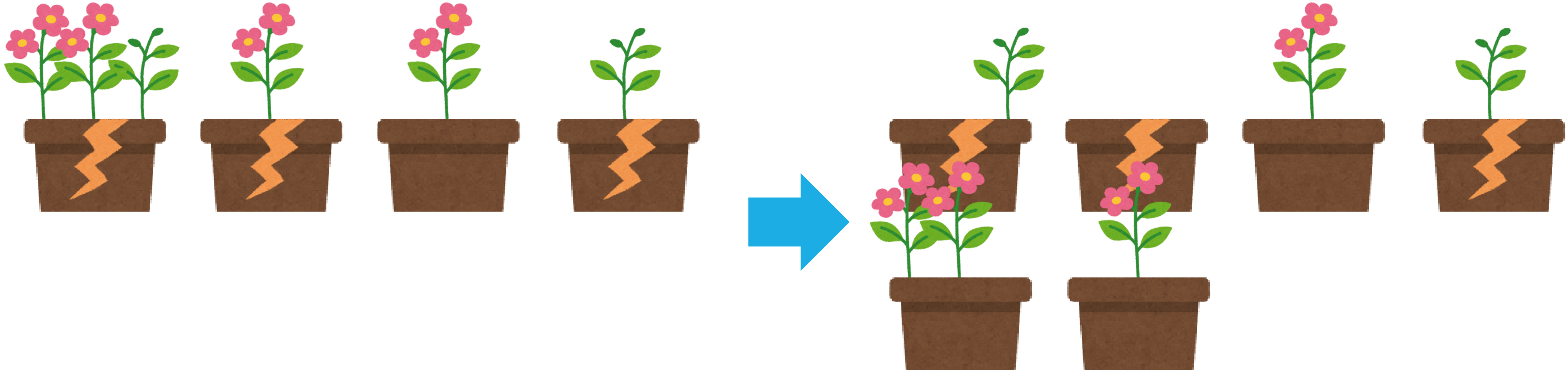

```
{t(X1),t(X2),s(Y1),s(Y2),s(Y3)},
{token,s(X1)}, {token,s(X2)},
{t(Y1)}, {t(Y1)}, {t(Y1)}
```

```
{t(X1),t(X2),s(Y1),s(Y2),s(Y3)},
{s(X1)}, {s(X2)},
{t(Y1),token}, {t(Y1),token}, {t(Y1),token}
```



Repotting the Geraniums[1]

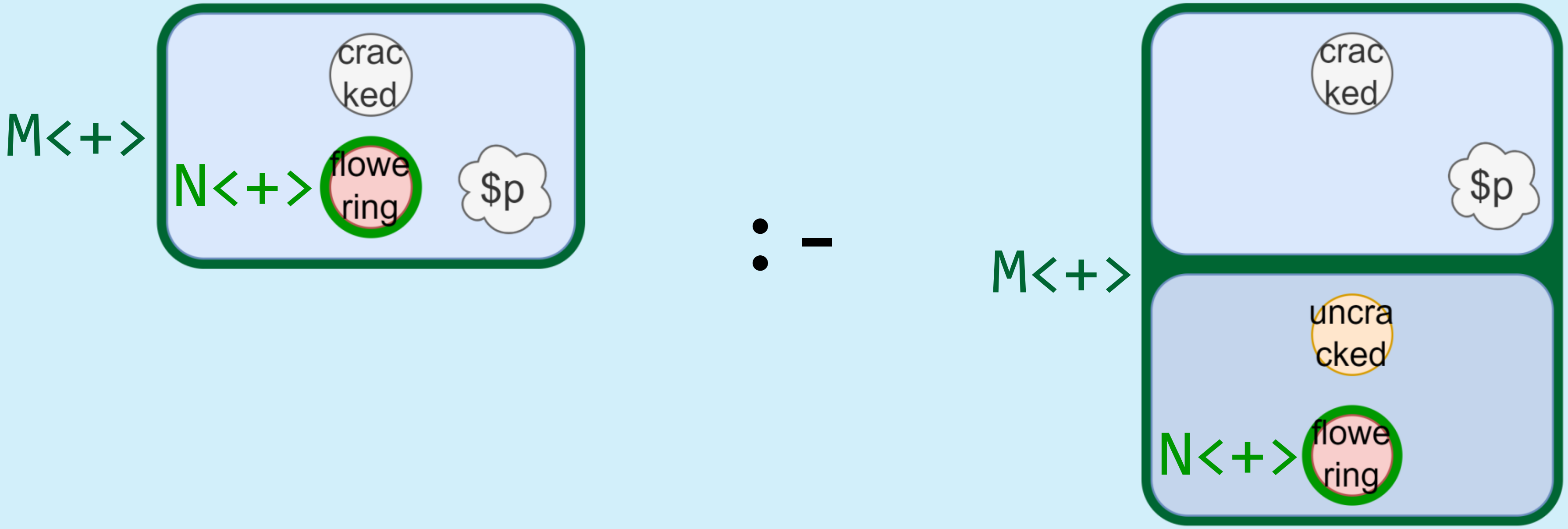
"There are several pots, each with several geranium plants. Some pots were broken because the geraniums filled the space with their roots. New pots are prepared for the broken pots with flowering geraniums and all the flowering geraniums are moved to the new pots."



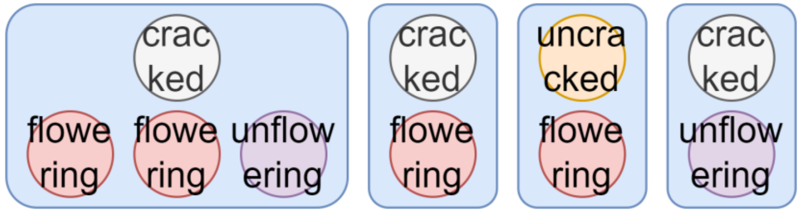
[1] Rensink, A., Kuperus, J.H.: Repotting the geraniums: On nested graph transformation rules. Electronic Communications of the EASST 18 (2009).

Repotting the Geraniums

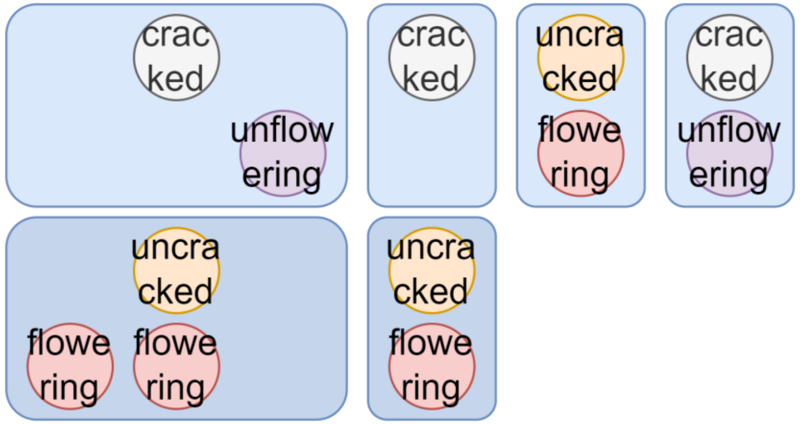
```
M<+>{cracked,N<+>flowering,$p} :- M<+>({cracked,$p},{uncracked,N<+>flowering})
```




```
{cracked,flowering,flowering,unflowering},
{cracked,flowering},
{uncracked,flowering},
{cracked,unflowering}
```




```
{cracked,unflowering},{uncracked,flowering,flowering}
{cracked,flowering},{uncracked,flowering},
{uncracked,flowering},
{cracked,unflowering}
```



Contents

Related Work

- Most graph rewriting tools
  - are based on *algebraic* (Double/Single-Pushout) approaches,
  - specify rewriting steps *visually*,
  - provide *sublanguages for execution control* (e.g., GP 2[5], PORGY),

  unlike LMNtal defined in a *(concurrent) programming language setting*.
- Some tools (e.g., GROOVE[4]) provide nested quantification within a single rule, whereas we use *inductively defined* syntax and semantics to allow nested quantification in a natural manner.
- Languages/tools that feature cardinality include in Answer Set Programming[6](ASP), practical regular expressions, and graph databases.

[4] Ghamarian, A., de Mol, M., Rensink, A., Zambon, E., Zimakova, M.: Modelling and analysis using GROOVE. Int. J. Softw. Tools. Technol. Transfer 14, 15–40 (2012).
[5] Plump, D.: The design of GP 2. Electronic Proceedings in Theoretical Computer Science 82, 1–16 (2012).
[6] Gebser, M., Kaminski, R., Kaufmann, B., Shaub, T.: Answer Set Solving in Practice. Springer Cham (2013).

Summary and Future Work

- Quantification has been well studied in the (mainstream) algebraic approach to graph rewriting. However,
- QLMNtal is the first attempt to *formalize quantification in the framework of (concurrent) programming languages* based on (i) *term-based syntax*, (ii) *structural congruence*, and (iii) *small-step semantics*.
- *Non-existence* quantification, *cardinality* quantification and *labeling* of quantification play important role in enabling us to express *universal* quantification, *nested* quantification, and *quantified rewriting*.
- Future work includes supporting the full expressive power of QLMNtal by extending SLIM (= LMNtal VM) and the LMNtal compiler.