On the Power and Universality of Biologically-inspired Models of Computation

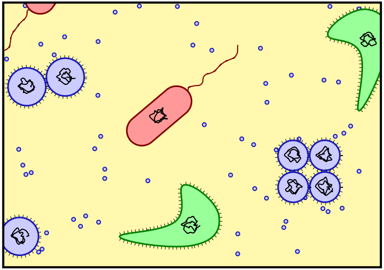
PhD Thesis

Sergiu IVANOV

Supervisor: Serghei VERLAN

LACL, Université Paris Est

June 23, 2015

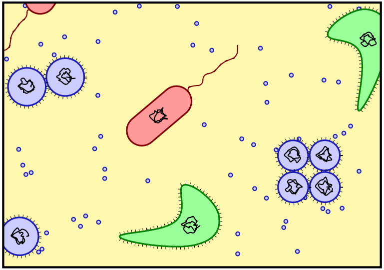


Biologically-inspired Models

Mimic biological processes

Biologically-inspired Models

Mimic biological processes



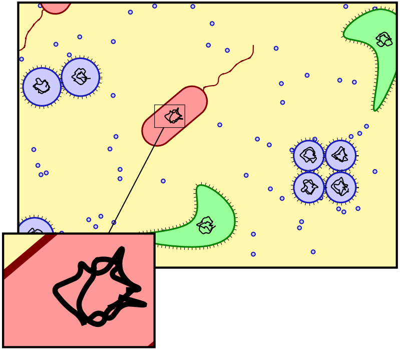
Biologically-inspired Models

Mimic biological processes

DNA/RNA operations

⇓

string rewriting



DNA/RNA

Biologically-inspired Models

Mimic biological processes

DNA/RNA operations

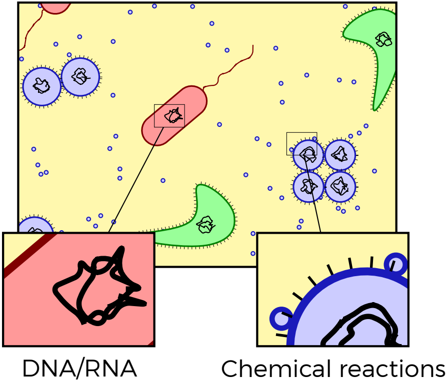
⇓

string rewriting

Chemical reactions

⇓

multiset rewriting



Biologically-inspired Models

Mimic biological processes

DNA/RNA operations

⇓

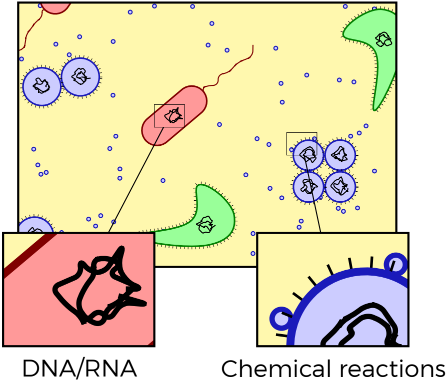
string rewriting

Chemical reactions

⇓

multiset rewriting

Focus on formal models



Biologically-inspired Models

Mimic biological processes

DNA/RNA operations

⇓

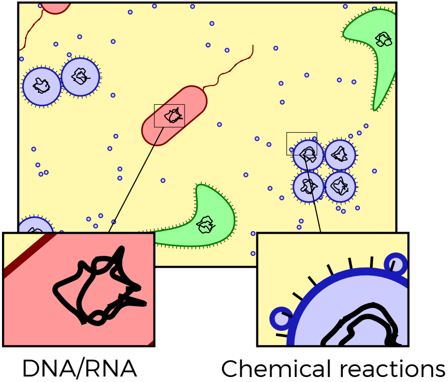
string rewriting

Chemical reactions

⇓

multiset rewriting

Focus on formal models



- Better understanding of complexity

Biologically-inspired Models

Mimic biological processes

DNA/RNA operations

⇓

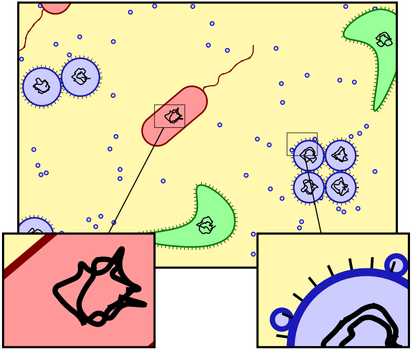
string rewriting

Chemical reactions

⇓

multiset rewriting

Focus on formal models



- Better understanding of complexity
- New models of computation

Biologically-inspired Models

Mimic biological processes

DNA/RNA operations

⇓

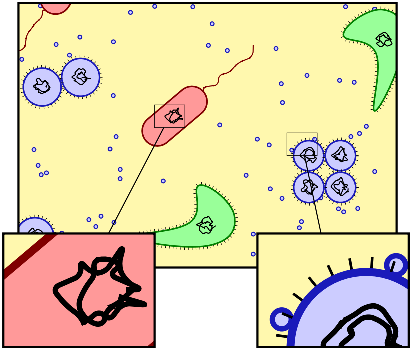
string rewriting

Chemical reactions

⇓

multiset rewriting

Focus on formal models



DNA/RNA

Chemical reactions

- Better understanding of complexity
- New models of computation

Presentation Map

Insertion and Deletion

Presentation Map

Insertion and Deletion

Multiset Rewriting

Presentation Map

Insertion and Deletion

Leftist insertion-deletion systems $(u, x, v)_{ins/del}$

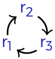
Multiset Rewriting

Presentation Map

Insertion and Deletion

Leftist insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control



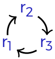
Multiset Rewriting

Presentation Map

Insertion and Deletion

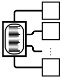
Leftist insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control



Multiset Rewriting

Small universal register machines

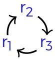


Presentation Map

Insertion and Deletion

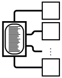
Leftist insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control



Multiset Rewriting

Small universal register machines



Small universal Petri nets

Presentation Map

Insertion and Deletion

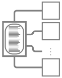
Leftist insertion-deletion systems

$(u, x, v)_{ins/del}$

Insertion-deletion systems with control

Multiset Rewriting

Small universal register machines



Small universal Petri nets

Presentation Map

Insertion and Deletion

Leftist insertion-deletion systems

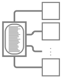
$(u, x, v)_{ins/del}$

- Introduction and motivation
- One-sided insertion-deletion systems
- Systems of sizes $(1, m, 0; 1, q, 0)$
- Derivation graphsi for $(1, 1, 0; 1, 1, 0)$

Insertion-deletion systems with control

Multiset Rewriting

Small universal register machines



Small universal Petri nets

Insertion-deletion Systems

$(u, x, v)_{ins}$

Insertion-deletion Systems

$$(u, x, v)_{\text{ins}}$$

$$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$$

Insertion-deletion Systems

$(u, x, v)_{\mathrm{ins}}$ $\qquad$ $(u, x, v)_{\mathrm{del}}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$ $\qquad$ $\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion Systems

$(u, x, v)_{ins}$

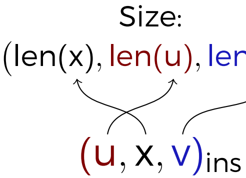
$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$

$(u, x, v)_{del}$

$\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

Insertion-deletion Systems



Size:

$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$

$(u, x, v)_{\mathrm{ins}}$

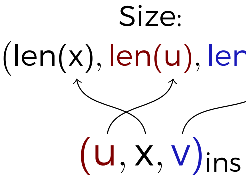
$(u, x, v)_{\mathrm{del}}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u\, x\, v \cdots$

$\cdots u\, x\, v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

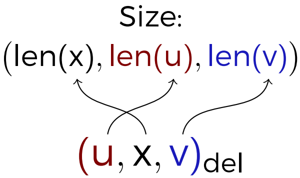
Insertion-deletion Systems



Size:

$$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$$

$$(u, x, v)_{\mathrm{ins}}$$



Size:

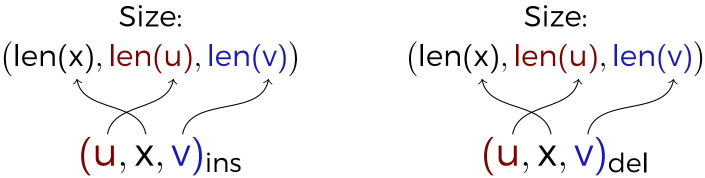
$$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$$

$$(u, x, v)_{\mathrm{del}}$$

$$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots \qquad \cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

Insertion-deletion Systems



Size:

$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$

$(u, x, v)_{\mathrm{ins}}$

Size:

$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$

$(u, x, v)_{\mathrm{del}}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$

$\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

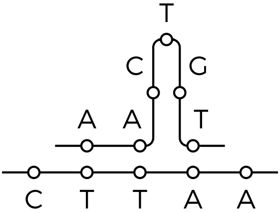
$$\text{System size} = (\underbrace{n, m, m'}_{\text{max insertion rule size}};\ \underbrace{p, q, q'}_{\text{max deletion rule size}})$$

Biological Motivation

Mismatched DNA annealing

Biological Motivation

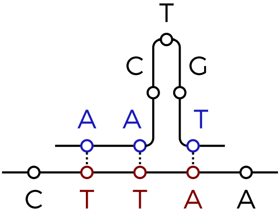
Mismatched DNA annealing



Biological Motivation

Mismatched DNA annealing

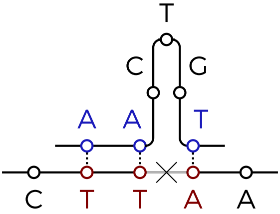
- two DNA strands bind
  - A – T
  - C – G



Biological Motivation

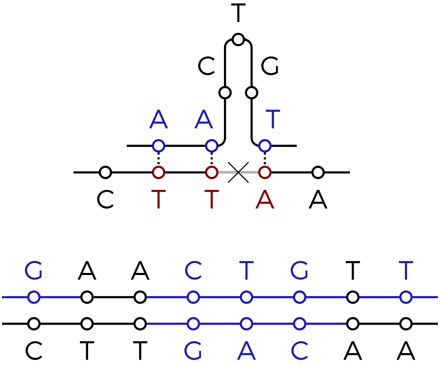
Mismatched DNA annealing

- two DNA strands bind
    - A – T
    - C – G
- the strands are cleft
    - enzymes



Biological Motivation

Mismatched DNA annealing

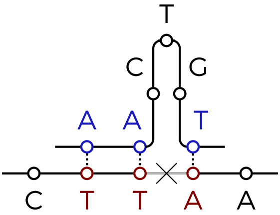


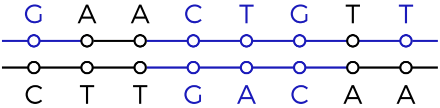
- two DNA strands bind
  - A – T
  - C – G
- the strands are cleft
  - enzymes
- both strands are filled in
  - complementarity

Biological Motivation

Mismatched DNA annealing

- two DNA strands bind
  - A – T
  - C – G
- the strands are cleft
  - enzymes
- both strands are filled in
  - complementarity





Context-free insertions and deletions on DNA strands

Formal Language Motivation

Context-free insertion = generalised concatenation

- insertion of size $(n, 0, 0)$

Formal Language Motivation

Context-free insertion = generalised concatenation

- insertion of size $(n, 0, 0)$

  Concatenation ($\bullet$):

  $\mathrm{abc} \bullet \mathrm{d} = \mathrm{abc\,d}$

Formal Language Motivation

Context-free insertion = generalised concatenation

- insertion of size $(n, 0, 0)$

Concatenation ($\bullet$):

$$abc \bullet d = abc\,d$$

Context-free insertion ($\leftarrow$):

$$ab\,c \leftarrow d = ab\,d\,c$$

Formal Language Motivation

Context-free insertion = generalised concatenation

- insertion of size $(n, 0, 0)$

Concatenation ($\bullet$):

$$abc \bullet d = abc\,d$$

Context-free insertion ($\leftarrow$):

$$ab\,c \leftarrow d = ab\,d\,c$$

Context-free deletion = generalised quotient

- deletion of size $(p, 0, 0)$

Formal Language Motivation

Context-free insertion = generalised concatenation

- insertion of size $(n, 0, 0)$

Concatenation ($\bullet$):

$$abc \bullet d = abc\,d$$

Context-free insertion ($\leftarrow$):

$$ab\,c \leftarrow d = ab\,d\,c$$

Context-free deletion = generalised quotient

- deletion of size $(p, 0, 0)$

Quotient ($/$):

$$abc\,\cancel{d} / d = abc$$

Context-free deletion ($\rightarrow$):

$$ab\,\cancel{d}\,c \rightarrow d = abc$$

Known Results on Insertion-deletion Systems

Context-free systems

- completeness

  $(3, 0, 0;\ 3, 0, 0) = \mathrm{RE}$
  $(3, 0, 0;\ 2, 0, 0) = \mathrm{RE}$
  $(2, 0, 0;\ 3, 0, 0) = \mathrm{RE}$

- incompleteness

  $(2, 0, 0;\ 2, 0, 0) \subsetneq \mathrm{CF}$
  $(m, 0, 0;\ 1, 0, 0) \subsetneq \mathrm{CF}$
  $(1, 0, 0;\ p, 0, 0) \subsetneq \mathrm{REG}$

Known Results on Insertion-deletion Systems

Context-free systems

- completeness

  $(3,0,0;\ 3,0,0) = \mathrm{RE}$
  $(3,0,0;\ 2,0,0) = \mathrm{RE}$
  $(2,0,0;\ 3,0,0) = \mathrm{RE}$

- incompleteness

  $(2,0,0;\ 2,0,0) \subsetneq \mathrm{CF}$
  $(m,0,0;\ 1,0,0) \subsetneq \mathrm{CF}$
  $(1,0,0;\ p,0,0) \subsetneq \mathrm{REG}$

Unconstrained systems

- completeness

  $(1,1,1;\ 2,0,0) = \mathrm{RE}$
  $(2,0,0;\ 1,1,1) = \mathrm{RE}$
  $(1,1,1;\ 1,1,0) = \mathrm{RE}$

Known Results on Insertion-deletion Systems

Context-free systems

- completeness

$$(3,0,0;\ 3,0,0) = \mathrm{RE}$$
$$(3,0,0;\ 2,0,0) = \mathrm{RE}$$
$$(2,0,0;\ 3,0,0) = \mathrm{RE}$$

- incompleteness

$$(2,0,0;\ 2,0,0) \subsetneq \mathrm{CF}$$
$$(m,0,0;\ 1,0,0) \subsetneq \mathrm{CF}$$
$$(1,0,0;\ p,0,0) \subsetneq \mathrm{REG}$$

Unconstrained systems

- completeness

$$(1,1,1;\ 2,0,0) = \mathrm{RE}$$
$$(2,0,0;\ 1,1,1) = \mathrm{RE}$$
$$(1,1,1;\ 1,1,0) = \mathrm{RE}$$

One-sided systems

- completeness

$$(1,1,2;\ 1,1,0) = \mathrm{RE}$$
$$(1,1,0;\ 1,1,2) = \mathrm{RE}$$

$$(2,0,2;\ 1,1,0) = \mathrm{RE}$$
$$(1,1,0;\ 2,0,2) = \mathrm{RE}$$

$$(2,0,1;\ 2,0,0) = \mathrm{RE}$$
$$(2,0,0;\ 2,0,1) = \mathrm{RE}$$

- incompleteness

$$(1,1,1;\ 1,1,0) \subsetneq \mathrm{RE}$$
$$(1,1,0;\ 1,1,1) \subsetneq \mathrm{RE}$$

$$(n,m,m';p,q,q')$$

- either $m = 0$ or $m' = 0$, not both
- either $q = 0$ or $q' = 0$, not both

Presentation Map

Insertion and Deletion

One-sided insertion-deletion systems

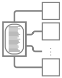
$(u, x, v)_{ins/del}$

- Introduction and motivation
- Leftist insertion-deletion systems
- Systems of sizes $(1, m, 0; 1, q, 0)$
- Derivation graphs for $(1, 1, 0; 1, 1, 0)$

Insertion-deletion systems with control

Multiset Rewriting

Small universal register machines

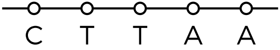


Small universal Petri nets

One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis



messenger RNA

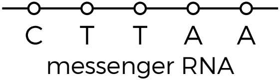
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA



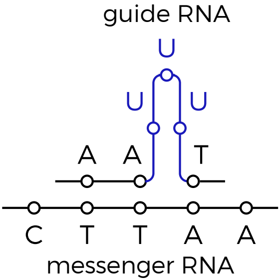
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA



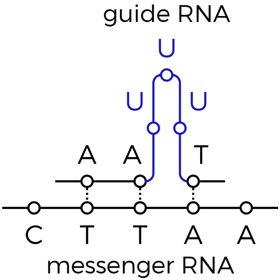
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA



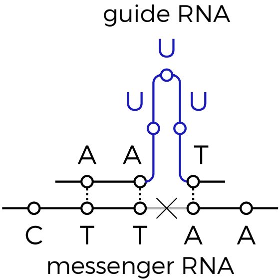
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA



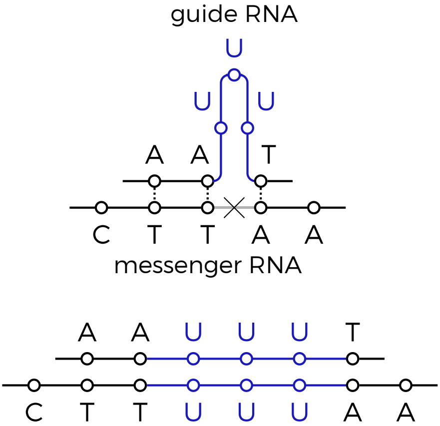
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA



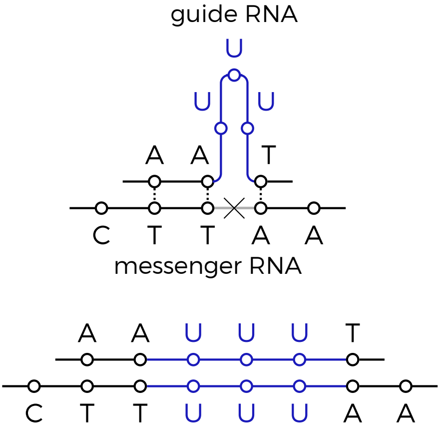
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA
- guide not modified



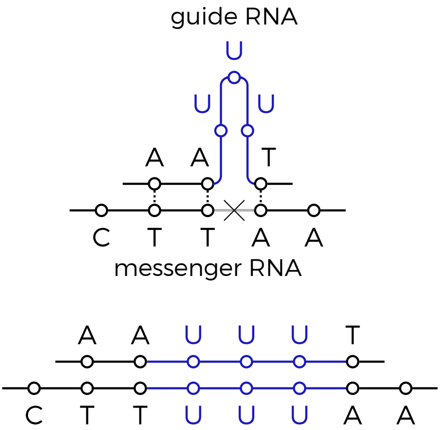
One-sided Systems: RNA Editing

RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA
- guide not modified
- only sequences of U inserted/deleted



One-sided Systems: RNA Editing

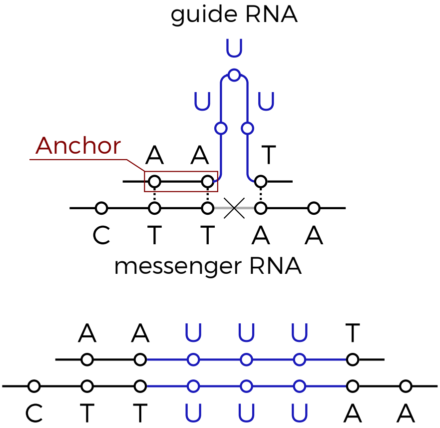
RNA: copy of DNA

- matrix for protein synthesis

RNA editing

- similar to mismatched annealing of DNA
- guide not modified
- only sequences of U inserted/deleted

Anchor always on same side



One-sided Systems: Accessibility Problems

Accessibility graphs

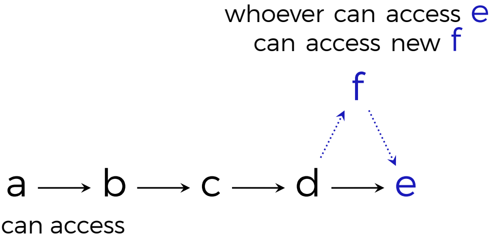
- can access

$$a \underset{\text{can access}}{\longrightarrow} b \longrightarrow c \longrightarrow d \longrightarrow e$$

One-sided Systems: Accessibility Problems

Accessibility graphs

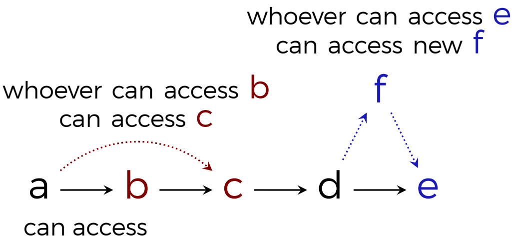
- can access
- give



One-sided Systems: Accessibility Problems

Accessibility graphs

- can access
- give
- get



whoever can access e
can access new f

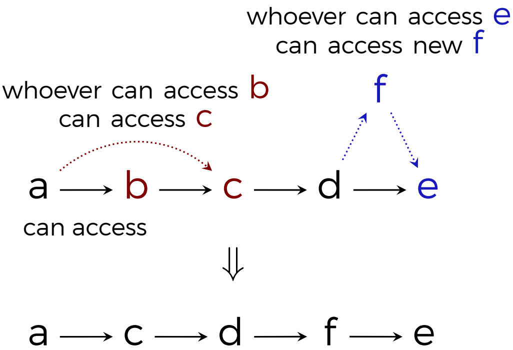
whoever can access b
can access c

can access

One-sided Systems: Accessibility Problems

Accessibility graphs

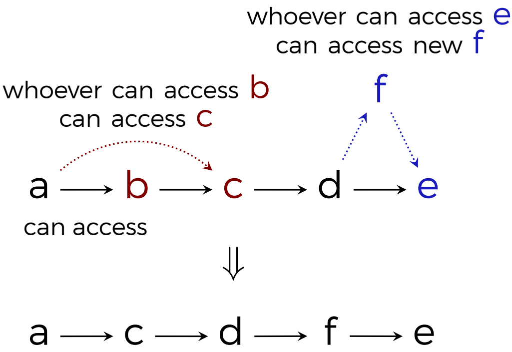
- can access
- give
- get



One-sided Systems: Accessibility Problems

Accessibility graphs

- can access
- give = insertion
- get = deletion



whoever can access e
can access new f

whoever can access b
can access c

$a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e$

can access

$\Downarrow$

$a \longrightarrow c \longrightarrow d \longrightarrow f \longrightarrow e$

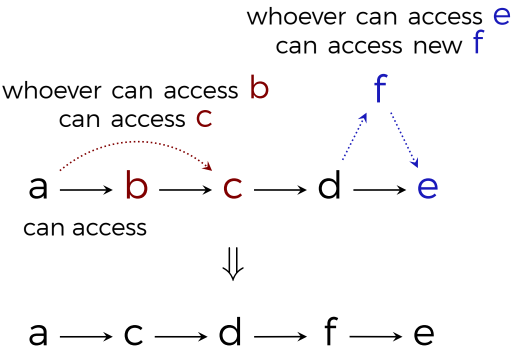
One-sided Systems: Accessibility Problems

Accessibility graphs

- can access
- give = insertion
- get = deletion

Leftist grammars

$(1, 1, 0; 1, 1, 0)$



One-sided Systems: Accessibility Problems

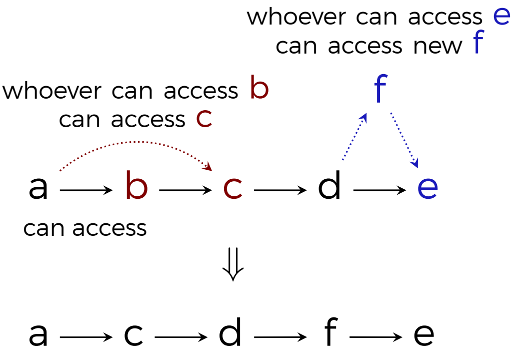
Accessibility graphs

- can access
- give = insertion
- get = deletion

Leftist grammars

$(1, 1, 0; 1, 1, 0)$

- $\not\ni (ba)^+$



whoever can access e
can access new f

whoever can access b
can access c

$a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e$

can access

$\Downarrow$

$a \longrightarrow c \longrightarrow d \longrightarrow f \longrightarrow e$

One-sided Systems: Accessibility Problems

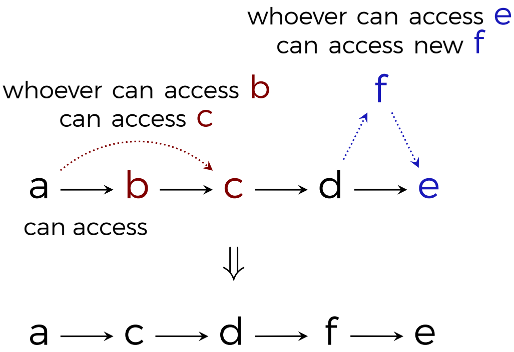
Accessibility graphs

- can access
- give = insertion
- get = deletion

Leftist grammars

$(1, 1, 0; 1, 1, 0)$

- $\not\ni (ba)^+$
- $\ni$ some CS languages



whoever can access e
can access new f

whoever can access b
can access c

$a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e$

can access

$\Downarrow$

$a \longrightarrow c \longrightarrow d \longrightarrow f \longrightarrow e$

One-sided Systems: Accessibility Problems

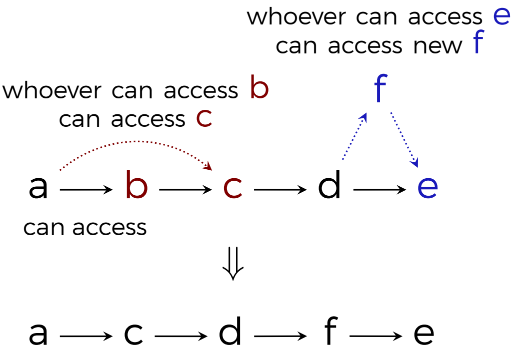
Accessibility graphs

- can access
- give = insertion
- get = deletion

Leftist grammars

$(1, 1, 0; 1, 1, 0)$

- $\not\ni (ba)^+$
- $\ni$ some CS languages



whoever can access e
can access new f

whoever can access b
can access c

$a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e$

can access

$\Downarrow$

$a \longrightarrow c \longrightarrow d \longrightarrow f \longrightarrow e$

We are interested in $(1, m, 0; 1, q, 0)$

Presentation Map

Insertion and Deletion

One-sided insertion-deletion systems

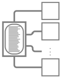
$(u, x, v)_{ins/del}$

- Introduction and motivation
- Leftist insertion-deletion systems
- Systems of sizes $(1, m, 0; 1, q, 0)$
- Derivation graphs for $(1, 1, 0; 1, 1, 0)$

Insertion-deletion systems with control

Multiset Rewriting

Small universal register machines



Small universal Petri nets

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

▶ ◀

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

▶ S ◀

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

▶ S S ◀

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

▶ u s s ◀

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

▶ a u s s ◀

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

▶ g a u s s ◀

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

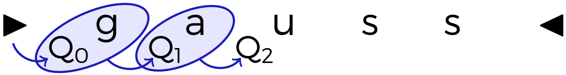
Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

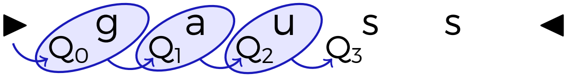
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



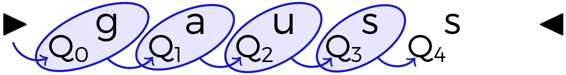
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



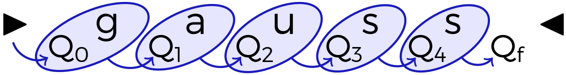
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



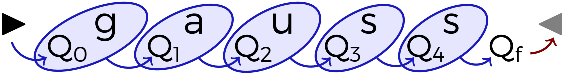
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



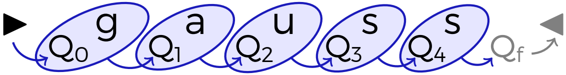
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



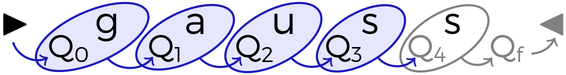
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



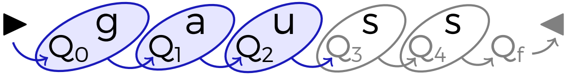
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



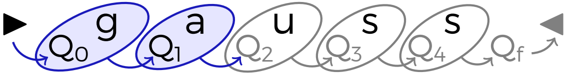
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



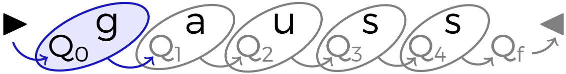
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



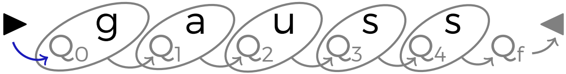
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



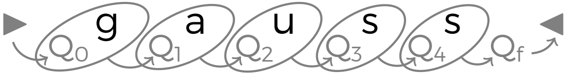
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



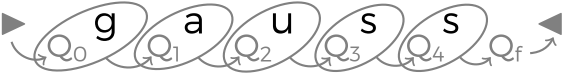
Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

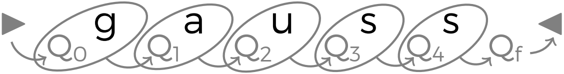
Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



- similarly for $(1, 1, 0; 1, 2, 0)$

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$

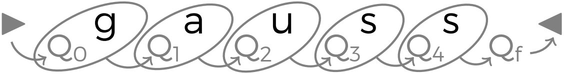


- similarly for $(1, 1, 0; 1, 2, 0)$

Simulate intersection with a REG language

Systems of Sizes $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$

Generation of regular languages (REG): $(1, 2, 0; 1, 1, 0)$



- similarly for $(1, 1, 0; 1, 2, 0)$

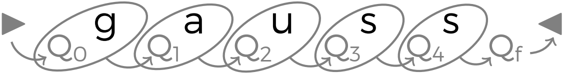
Simulate intersection with a REG language

$(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$ generate

$$L_{2^n} = \{(F_1F_0)^n(a_0a_1)^m \mid n \geq 2^{2m-2}\}$$

Systems of Sizes $(1,2,0;1,1,0)$ and $(1,1,0;1,2,0)$

Generation of regular languages (REG): $(1,2,0;1,1,0)$



- similarly for $(1,1,0;1,2,0)$

Simulate intersection with a REG language

$(1,2,0;1,1,0)$ and $(1,1,0;1,2,0)$ generate

$$L_{2^n} = \{(F_1F_0)^n(a_0a_1)^m \mid n \geq 2^{2m-2}\}$$

- $(1,1,0;1,1,0)$ intersected with a REG language generate L_{2^n}

$(1, m, 0; 1, q, 0)$: Longer Contexts

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{del}$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{del}$ $(k = 2)$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{del}$ $(k = 2)$

$\cdots a\, b\, c \cdots$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab\overset{\times}{,} c, \lambda)_{\mathrm{del}}$ $(k = 2)$

$$\cdots \underbrace{a\,b}\,c \cdots \xRightarrow{(ab, X, \lambda)_{\mathrm{ins}}} \cdots a\,b\,X\,c \cdots$$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; \underset{\rightleftarrows}{1, k}, 0)$

let $(ab, c, \lambda)_{del}$ $(k = 2)$

$$\cdots \underbrace{ab}\,c \cdots \xRightarrow{(ab, X, \lambda)_{ins}} \cdots ab\,\underbrace{X}\,\cancel{c} \cdots \xRightarrow{(X, c, \lambda)_{del}} \cdots ab\,X \cdots$$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{\mathrm{del}}$ $(k = 2)$

$$\cdots \underbrace{a\,b}_{}\,c \cdots \xRightarrow{(ab, X, \lambda)_{\mathrm{ins}}} \cdots a\,b\,\underbrace{X}_{}\,\cancel{c} \cdots \xRightarrow{(X, c, \lambda)_{\mathrm{del}}} \cdots a\,b\,X \cdots \xRightarrow{(\lambda, X, \lambda)_{\mathrm{del}}} \cdots a\,b \cdots$$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{\mathrm{del}}$ $(k = 2)$

$$\cdots \underbrace{a\,b}c\cdots \xRightarrow{(ab, X, \lambda)_{\mathrm{ins}}} \cdots a\,b\underbrace{X}\cancel{c}\cdots \xRightarrow{(X, c, \lambda)_{\mathrm{del}}} \cdots a\,b\,X\cdots \xRightarrow{(\lambda, X, \lambda)_{\mathrm{del}}} \cdots a\,b\cdots$$

- Similarly for other simulations

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{del}$ $(k = 2)$

$$\cdots \underbrace{ab}c \cdots \xRightarrow{(ab, X, \lambda)_{ins}} \cdots ab\underbrace{X}\cancel{c} \cdots \xRightarrow{(X, c, \lambda)_{del}} \cdots abX \cdots \xRightarrow{(\lambda, X, \lambda)_{del}} \cdots ab \cdots$$

- Similarly for other simulations

In fact, $(1, k, 0; 1, k, 0) \sim (1,\ k + 1,\ 0;\ 1,\ k + 1,\ 0)$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{del}$ $(k = 2)$

$$\cdots \underbrace{a\,b}_{}c \cdots \xRightarrow{(ab, X, \lambda)_{ins}} \cdots a\,b\,\underbrace{X}_{}\cancel{c} \cdots \xRightarrow{(X, c, \lambda)_{del}} \cdots a\,b\,X \cdots \xRightarrow{(\lambda, X, \lambda)_{del}} \cdots a\,b \cdots$$

- Similarly for other simulations

In fact, $(1, k, 0; 1, k, 0) \sim (1,\ k+1,\ 0;\ 1,\ k+1,\ 0)$

Therefore $(1, 2, 0; 1, 1, 0) \sim (1, 1, 0; 1, 2, 0) \sim (1, m, 0; 1, q, 0)$

$(1, m, 0; 1, q, 0)$: Longer Contexts

generate the same languages

$$(1, k, 0; 1, 1, 0) \sim (1, 1, 0; 1, k, 0) \sim (1, k, 0; 1, k, 0)$$

$(1, k, 0; 1, 1, 0)$ simulates $(1, 1, 0; 1, k, 0)$

let $(ab, c, \lambda)_{del}$ $(k = 2)$

$$\cdots \underline{a\,b}\,c \cdots \xRightarrow{(ab, X, \lambda)_{ins}} \cdots a\,b\,\underline{X}\,\cancel{c} \cdots \xRightarrow{(X, c, \lambda)_{del}} \cdots a\,b\,X \cdots \xRightarrow{(\lambda, X, \lambda)_{del}} \cdots a\,b \cdots$$

- Similarly for other simulations

In fact, $(1, k, 0; 1, k, 0) \sim (1,\ k+1,\ 0;\ 1,\ k+1,\ 0)$

Therefore $(1, 2, 0; 1, 1, 0) \sim (1, 1, 0; 1, 2, 0) \sim (1, m, 0; 1, q, 0)$

Conjecture:
$(1, m, 0; 1, q, 0)$ not computationally complete

Presentation Map

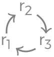
Insertion and Deletion

One-sided insertion-deletion systems

$(u, x, v)_{ins/del}$

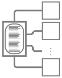
- Introduction and motivation
- One-sided insertion-deletion systems
- Systems of sizes $(1, m, 0; 1, q, 0)$
- Derivation graphs for $(1, 1, 0; 1, 1, 0)$

Insertion-deletion systems with control



Multiset Rewriting

Small universal register machines



Small universal Petri nets

Derivation Graphs for $(1, 1, 0; 1, 1, 0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

Derivation Graphs for $(1, 1, 0; 1, 1, 0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

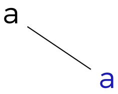
a

a

Derivation Graphs for $(1, 1, 0; 1, 1, 0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$



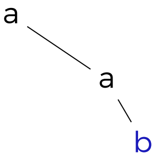
$$a \xRightarrow{r_1} \underset{\curvearrowright}{aa}$$

Derivation Graphs for $(1,1,0;1,1,0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

$$a \overset{r_1}{\Longrightarrow} \underset{\curvearrowright}{aa} \overset{r_2}{\Longrightarrow} a\underset{\curvearrowright}{ab}$$

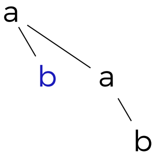


Derivation Graphs for $(1, 1, 0; 1, 1, 0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

$$a \xRightarrow{r_1} \underset{\curvearrowright}{aa} \xRightarrow{r_2} a\underset{\curvearrowright}{ab} \xRightarrow{r_2} \underset{\curvearrowright}{ab}ab$$

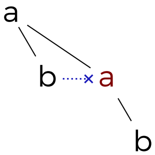


Derivation Graphs for $(1, 1, 0; 1, 1, 0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

$$a \xRightarrow{r_1} aa \xRightarrow{r_2} aab \xRightarrow{r_2} abab \xRightarrow{r_3} abb$$

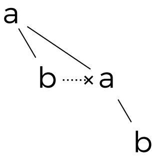


Derivation Graphs for $(1,1,0;1,1,0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

$$a \xRightarrow{r_1} \underset{\curvearrowright}{aa} \xRightarrow{r_2} a\underset{\curvearrowright}{ab} \xRightarrow{r_2} \underset{\curvearrowright}{ab}\overset{\cancel{\curvearrowright}}{ba}b \xRightarrow{r_3} abb$$



The system generating $L_{2^n} = \{(F_1F_0)^n(a_0a_1)^m \mid n \geq 2^{2m-2}\}$:

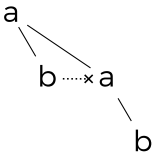
- intersection with $(F_1F_0)^*(a_0a_1)^*$

Derivation Graphs for $(1,1,0;1,1,0)$

$r_1 : (a, a, \lambda)_{ins}$, $r_2 : (a, b, \lambda)_{ins}$,

$r_3 : (b, a, \lambda)_{del}$

$$a \xRightarrow{r_1} aa \xRightarrow{r_2} aab \xRightarrow{r_2} abab \xRightarrow{r_3} abb$$



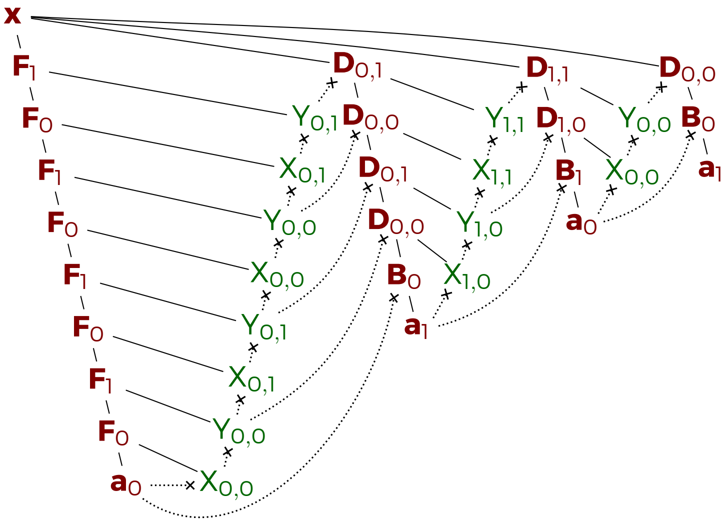
The system generating $L_{2^n} = \{(F_1F_0)^n(a_0a_1)^m \mid n \geq 2^{2m-2}\}$:

- intersection with $(F_1F_0)^*(a_0a_1)^*$

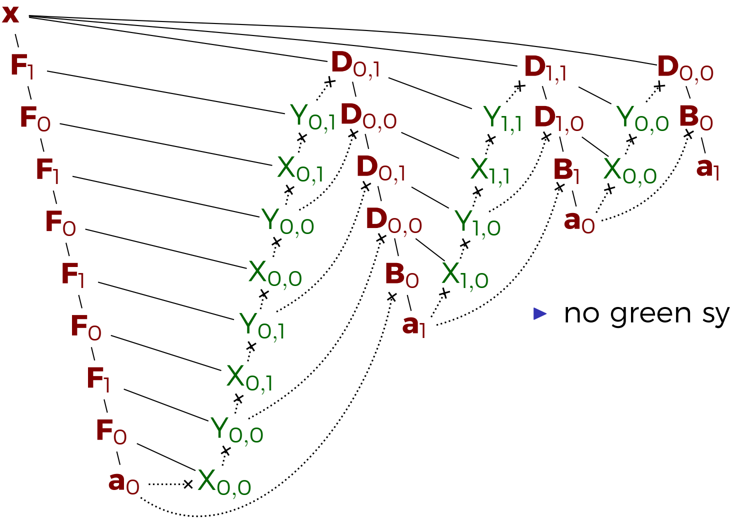
$(a_i, B_i, \lambda)_{del}$,	$(B_i, a_{1-i}, \lambda)_{ins}$,	$(F_0, a_0, \lambda)_{ins}$,
$(a_i, X_{i,0}, \lambda)_{del}$,	$(D_{i,j}, B_i, \lambda)_{ins}$,	$(F_0, X_{0,j}, \lambda)_{ins}$,
$(X_{i,j}, Y_{i,j}, \lambda)_{del}$	$(D_{i,1-j}, D_{i,j}, \lambda)_{ins}$	$(F_1, Y_{0,j}, \lambda)_{ins}$,
$(Y_{i,j}, D_{i,j}, \lambda)_{del}$	$(D_{i,0}, X_{1-i,k}, \lambda)_{ins}$	$(F_i, F_{1-i}, \lambda)_{ins}$,
$(Y_{i,j}, X_{i,1-j}, \lambda)_{del}$	$(D_{i,1}, Y_{1-i,k}, \lambda)_{ins}$	$(x, F_1, \lambda)_{ins}$,
		$(x, D_{i,j}, \lambda)_{ins}$,

where $i, j, k \in \{0, 1\}$

Derivation Graphs for Generation of L_{2^n}

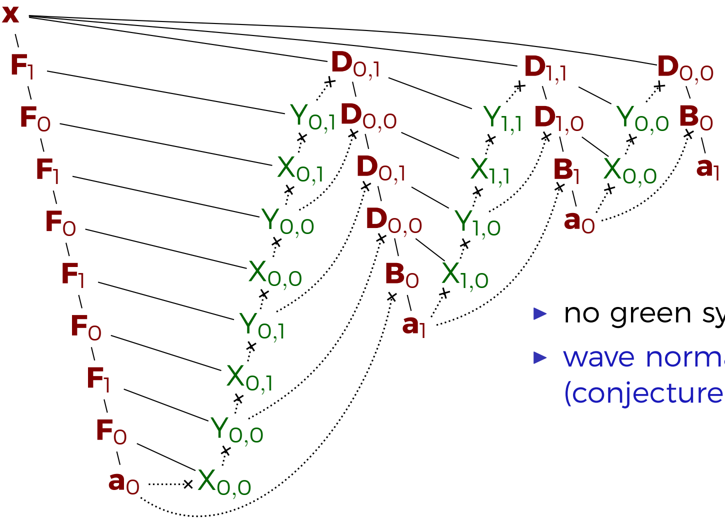


Derivation Graphs for Generation of L_{2^n}



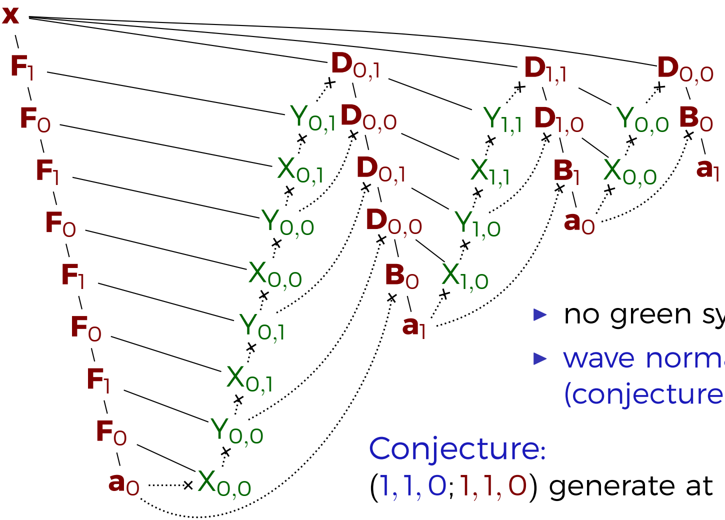
- no green symbols

Derivation Graphs for Generation of L_{2^n}



- no green symbols
- wave normal form (conjecture)

Derivation Graphs for Generation of L_{2^n}



- no green symbols
- wave normal form (conjecture)

Conjecture:
(1, 1, 0; 1, 1, 0) generate at most CS

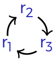
Presentation Map

Insertion and Deletion

One-sided insertion-deletion systems

$(u, x, v)_{ins/del}$

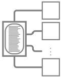
Insertion-deletion systems with control



- Introduction and motivation
- Graph-controlled systems
- Semi-conditional and random context systems
- Networks of evolutionary processors

Multiset Rewriting

Small universal register machines



Small universal Petri nets

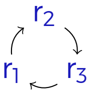
Control Mechanisms

Add pre-conditions to rules

Control Mechanisms

Add pre-conditions to rules

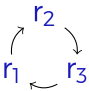
- prescribe rule application
  language



Control Mechanisms

Add pre-conditions to rules

- prescribe rule application language

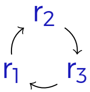


- prescribe language of valid strings
  - require the string to be of a certain form

Control Mechanisms

Add pre-conditions to rules

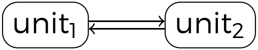
- prescribe rule application language



r_2

r_1 r_3

- prescribe language of valid strings
  - require the string to be of a certain form
- distributed control

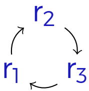


unit_1 unit_2

Control Mechanisms

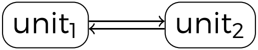
Add pre-conditions to rules

- prescribe rule application language

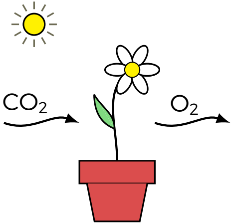


r_2 r_1 r_3

- prescribe language of valid strings
  - require the string to be of a certain form
- distributed control



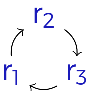
unit_1 unit_2



Control Mechanisms

Add pre-conditions to rules

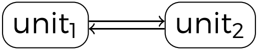
- prescribe rule application language



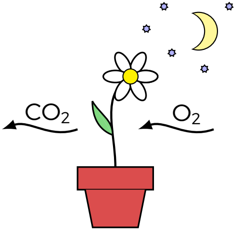
r_2 r_1 r_3

- prescribe language of valid strings
  - require the string to be of a certain form

- distributed control



unit_1 unit_2

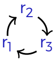


Presentation Map

Insertion and Deletion

One-sided insertion-deletion systems $(u, x, v)_{ins/del}$

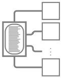
Insertion-deletion systems with control



- Introduction and motivation
- Graph-controlled systems
- Semi-conditional and random context systems
- Networks of evolutionary processors

Multiset Rewriting

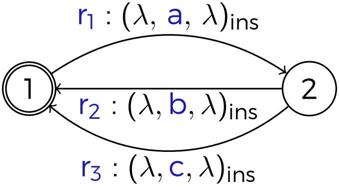
Small universal register machines



Small universal Petri nets

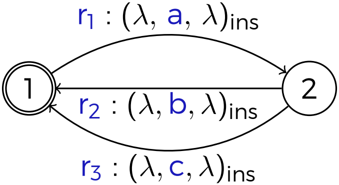
Graph-controlled Insertion-deletion Systems

Consider the system of size $(1, 1, 0; 1, 1, 0)$:



Graph-controlled Insertion-deletion Systems

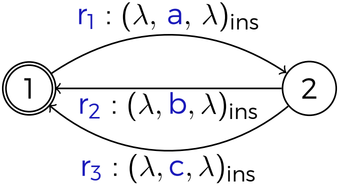
Consider the system of size $(1, 1, 0; 1, 1, 0)$:



$$L = \{w : |w|_a = |w|_b + |w|_c\}$$

Graph-controlled Insertion-deletion Systems

Consider the system of size $(1, 1, 0; 1, 1, 0)$:

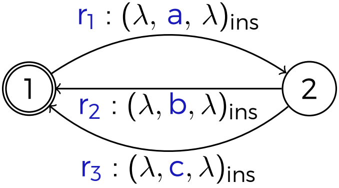


$$L = \{w : |w|_a = |w|_b + |w|_c\}$$

(regular rule application language)

Graph-controlled Insertion-deletion Systems

Consider the system of size $(1, 1, 0; 1, 1, 0)$:



$$L = \{w : |w|_a = |w|_b + |w|_c\}$$

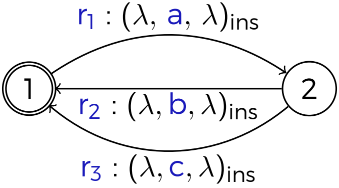
(regular rule application language)

Known fact:

4 nodes + $(1, 1, 0; 1, 1, 0)$ – computationally complete

Graph-controlled Insertion-deletion Systems

Consider the system of size $(1, 1, 0; 1, 1, 0)$:



$$L = \{w : |w|_a = |w|_b + |w|_c\}$$

(regular rule application language)

Known fact:

4 nodes + $(1, 1, 0; 1, 1, 0)$ – computationally complete

We showed that:

3 nodes + $\begin{matrix}(1, 2, 0; 1, 1, 0)\\(1, 1, 0; 1, 2, 0)\end{matrix}$ – computationally complete

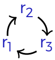
Presentation Map

Insertion and Deletion

One-sided insertion-deletion systems

$(u, x, v)_{ins/del}$

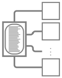
Insertion-deletion systems with control



- Introduction and motivation
- Graph-controlled systems
- Semi-conditional and random context systems
- Networks of evolutionary processors

Multiset Rewriting

Small universal register machines



Small universal Petri nets

Semi-conditional and Random Context Systems

Semi-conditional control

$(\lambda,\ \mathsf{a},\ \lambda)_{\mathsf{ins}}$

Semi-conditional and Random Context Systems

Semi-conditional control

$$\big((\lambda,\ \mathsf{a},\ \lambda)_{\mathsf{ins}},\ \overset{\substack{\text{permitting}\\\text{context condition}}}{\{\mathsf{S}\}},\ \overset{\substack{\text{forbidding}\\\text{context condition}}}{\{\mathsf{ab},\mathsf{ba}\}}\big)$$

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\\text{context condition}}}{1}, \underset{\substack{\text{forbidding}\\\text{context condition}}}{2})$$

$$\big((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big)$$

Semi-conditional and Random Context Systems

Semi-conditional control

degree $= (1\ ,\ 2)$

permitting context condition

forbidding context condition

$$\big((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big) \qquad \big((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big)$$

Semi-conditional and Random Context Systems

Semi-conditional control

degree $= (1\ ,\ 2)$

permitting context condition

forbidding context condition

$$\big((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big) \qquad \big((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big)$$

$$\big((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\}\big)$$

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\\text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\\text{context condition}}}{2})$$

$$\big((\lambda,\ \mathsf{a},\ \lambda)_{\mathsf{ins}},\ \{\mathsf{S}\},\ \{\mathsf{ab},\mathsf{ba}\}\big) \qquad \big((\lambda,\ \mathsf{b},\ \lambda)_{\mathsf{ins}},\ \{\mathsf{S}\},\ \{\mathsf{ab},\mathsf{ba}\}\big)$$

$$\big((\lambda,\ \mathsf{S},\ \lambda)_{\mathsf{del}},\ \varnothing,\ \{\mathsf{ab},\mathsf{ba}\}\big)$$

aSb

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (1\ ,\ 2)$$

permitting context condition

forbidding context condition

$$\big((\lambda,\ \text{a},\ \lambda)_{\text{ins}},\ \{\text{S}\},\ \{\text{ab}, \text{ba}\}\big) \qquad \big((\lambda,\ \text{b},\ \lambda)_{\text{ins}},\ \{\text{S}\},\ \{\text{ab}, \text{ba}\}\big)$$

$$\big((\lambda,\ \text{S},\ \lambda)_{\text{del}},\ \varnothing,\ \{\text{ab}, \text{ba}\}\big)$$

$$\text{aSb} \Longrightarrow \text{aaSb}$$

Semi-conditional and Random Context Systems

Semi-conditional control

degree $= (1\ ,\ 2)$

permitting context condition

forbidding context condition

$$\big((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big) \qquad \big((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big)$$

$$\big((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\}\big)$$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb$$

Semi-conditional and Random Context Systems

Semi-conditional control

degree = (1 , 2)

permitting context condition

forbidding context condition

$$((\lambda,\ \mathsf{a},\ \lambda)_{\mathsf{ins}},\ \{\mathsf{S}\},\ \{\mathsf{ab}, \mathsf{ba}\}) \qquad ((\lambda,\ \mathsf{b},\ \lambda)_{\mathsf{ins}},\ \{\mathsf{S}\},\ \{\mathsf{ab}, \mathsf{ba}\})$$

$$((\lambda,\ \mathsf{S},\ \lambda)_{\mathsf{del}},\ \varnothing,\ \{\mathsf{ab}, \mathsf{ba}\})$$

$$\mathsf{aSb} \Longrightarrow \mathsf{aaSb} \Longrightarrow \mathsf{aaSbb} \Longrightarrow \mathsf{aabb}$$

Semi-conditional and Random Context Systems

Semi-conditional control

degree = (1 , 2)

permitting context condition, forbidding context condition

$$\big((\lambda,\ \mathsf{a},\ \lambda)_{\mathsf{ins}},\ \{\mathsf{S}\},\ \{\mathsf{ab},\mathsf{ba}\}\big) \qquad \big((\lambda,\ \mathsf{b},\ \lambda)_{\mathsf{ins}},\ \{\mathsf{S}\},\ \{\mathsf{ab},\mathsf{ba}\}\big)$$

$$\big((\lambda,\ \mathsf{S},\ \lambda)_{\mathsf{del}},\ \varnothing,\ \{\mathsf{ab},\mathsf{ba}\}\big)$$

$$\mathsf{aSb} \Longrightarrow \mathsf{aaSb} \Longrightarrow \mathsf{aaSbb} \Longrightarrow^{*} \mathsf{a}^{*}\mathsf{b}^{*}$$

Semi-conditional and Random Context Systems

Semi-conditional control

degree = (1 , 2)

permitting context condition — forbidding context condition

$$\big((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big) \qquad \big((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}\big)$$

$$\big((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\}\big)$$

$$\begin{tikzcd}
aSb \arrow[r, Rightarrow] & aaSb \arrow[r, Rightarrow] & aaSbb \arrow[r, Rightarrow, "*"] \arrow[dr, Rightarrow] & a^*b^* \\
 & & & abaSbb
\end{tikzcd}$$

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\ \text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\ \text{context condition}}}{2})$$

$$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}) \qquad ((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$$

$$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

$$\searrow$$

Only terminal strings!

$$ab a\overset{\substack{\text{non-terminal}\\ \downarrow}}{S}bb$$

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\ \text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\ \text{context condition}}}{2})$$

$$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}) \qquad ((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$$

$$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

Only terminal strings! $\Searrow$ $aba\overset{\substack{\text{non-terminal}\\ \downarrow}}{S}bb$

degree $(2, 2)$ + $(1, 0, 0;\ 1, 0, 0)$ – computationally complete

Semi-conditional and Random Context Systems

Semi-conditional control

$\text{degree} = (\underset{\substack{\text{permitting}\\ \text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\ \text{context condition}}}{2})$

$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$ $\quad$ $((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$

$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

$$\Searrow\ aba\overset{\substack{\text{non-terminal}\\ \downarrow}}{S}bb$$

Only terminal strings!

degree $(2, 2)$ + $(1, 0, 0; 1, 0, 0)$ – computationally complete

degree $(2, 2)$ + $(1, 0, 0; 0, 0, 0)$ – contained in CS

Semi-conditional and Random Context Systems

Semi-conditional control

degree = (1 , 2)

permitting context condition forbidding context condition

$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$ $((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$

$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

Only terminal strings!

$$\searrow\ aba\overset{\text{non-terminal}}{\overset{\downarrow}{S}}bb$$

degree $(2, 2)$ + $(1, 0, 0; 1, 0, 0)$ – computationally complete

degree $(2, 2)$ + $(1, 0, 0; 0, 0, 0)$ – contained in CS

Random context control = degree $(1, 1)$

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\ \text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\ \text{context condition}}}{2})$$

$$((\lambda,\ a,\ \lambda)_{\text{ins}},\ \{S\},\ \{ab, ba\}) \qquad ((\lambda,\ b,\ \lambda)_{\text{ins}},\ \{S\},\ \{ab, ba\})$$

$$((\lambda,\ S,\ \lambda)_{\text{del}},\ \varnothing,\ \{ab, ba\})$$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

$$\searrow\ ab a\overset{\substack{\text{non-terminal}\\ \downarrow}}{S}bb$$

Only terminal strings!

degree (2, 2) + (1, 0, 0; 1, 0, 0) – computationally complete

degree (2, 2) + (1, 0, 0; 0, 0, 0) – contained in CS

Random context control = degree (1, 1)

degree (1, 1) + (2, 0, 0; 1, 1, 0) – computationally complete

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\ \text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\ \text{context condition}}}{2})$$

$$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}) \qquad ((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$$

$$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

Only terminal strings!

$$\Searrow\ aba\overset{\substack{\text{non-terminal}\\ \downarrow}}{S}bb$$

degree (2, 2) + (1, 0, 0; 1, 0, 0) – computationally complete

degree (2, 2) + (1, 0, 0; 0, 0, 0) – contained in CS

Random context control = degree (1, 1)

degree (1, 1) + (2, 0, 0; 1, 1, 0) – computationally complete

degree (1, 1) + (1, 1, 0; 2, 0, 0) – not computationally complete

Semi-conditional and Random Context Systems

Semi-conditional control

$$\text{degree} = (\underset{\substack{\text{permitting}\\ \text{context condition}}}{1}\ ,\ \underset{\substack{\text{forbidding}\\ \text{context condition}}}{2})$$

$$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\}) \qquad ((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$$

$$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

$$aaSbb \Searrow ab a \overset{\substack{\text{non-terminal}\\ \downarrow}}{S} bb$$

Only terminal strings!

degree $(2,2)$ + $(1,0,0;\ 1,0,0)$ – computationally complete

degree $(2,2)$ + $(1,0,0;\ 0,0,0)$ – contained in CS

Random context control = degree $(1,1)$

degree $(1,1)$ + $(2,0,0;\ 1,1,0)$ – computationally complete

degree $(1,1)$ + $(1,1,0;\ p,1,1)$ – not computationally complete

Semi-conditional and Random Context Systems

Semi-conditional control

degree = (1 , 2)

permitting context condition

forbidding context condition

$((\lambda,\ a,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$ $((\lambda,\ b,\ \lambda)_{ins},\ \{S\},\ \{ab, ba\})$

$((\lambda,\ S,\ \lambda)_{del},\ \varnothing,\ \{ab, ba\})$

$$aSb \Longrightarrow aaSb \Longrightarrow aaSbb \Longrightarrow^* \overbrace{a^*b^*}^{\text{terminals}}$$

Only terminal strings!

$\Downarrow$ non-terminal $\downarrow$ abaSbb

degree (2, 2) + (1, 0, 0; 1 , 0, 0) – computationally complete

degree (2, 2) + (1, 0, 0; 0, 0, 0) – contained in CS

Random context control = degree (1, 1)

degree (1, 1) + (2, 0, 0 ; 1, 1, 0) – computationally complete

degree (1, 1) + (1, 1, 0 ; p, 1, 1) – not computationally complete

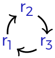
unusual asymmetry

Presentation Map

Insertion and Deletion

One-sided insertion-deletion systems $(u, x, v)_{ins/del}$

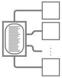
Insertion-deletion systems with control



- Introduction and motivation
- Graph-controlled systems
- Semi-conditional and random context systems
- Networks of evolutionary processors

Multiset Rewriting

Small universal register machines



Small universal Petri nets

Networks of Evolutionary Processors (NEPs)

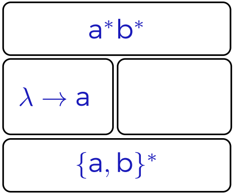
Distributed control

- insertion
- deletion
- substitution

Networks of Evolutionary Processors (NEPs)

Distributed control

- insertion
- deletion
- substitution



a^*b^*

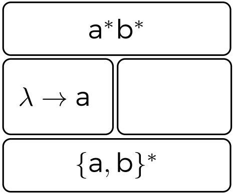
$\lambda \to a$

$\{a, b\}^*$

Networks of Evolutionary Processors (NEPs)

Distributed control

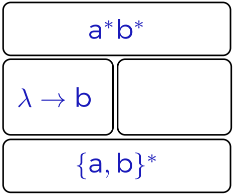
- insertion
- deletion
- substitution



a^*b^*

$\lambda \to a$

$\{a, b\}^*$



a^*b^*

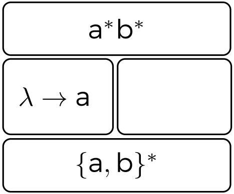
$\lambda \to b$

$\{a, b\}^*$

Networks of Evolutionary Processors (NEPs)

Distributed control

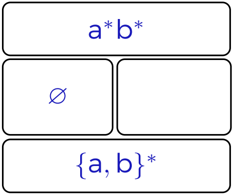
- insertion
- deletion
- substitution



a^*b^*

$\lambda \to a$

$\{a, b\}^*$

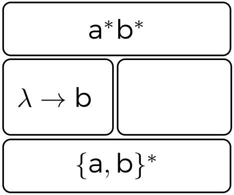


a^*b^*

$\varnothing$

$\{a, b\}^*$

↑
output here



a^*b^*

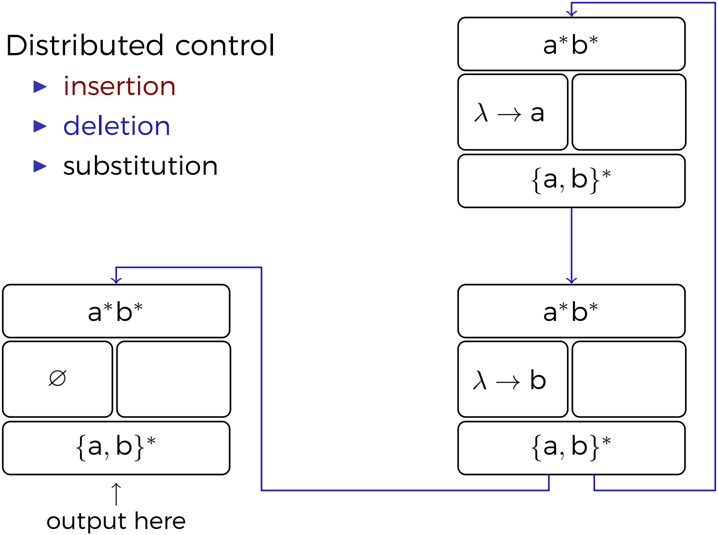
$\lambda \to b$

$\{a, b\}^*$

Networks of Evolutionary Processors (NEPs)

Distributed control

- insertion
- deletion
- substitution



$a^* b^*$

$\lambda \to a$

$\{a, b\}^*$

$a^* b^*$

$\varnothing$

$\{a, b\}^*$

↑

output here

$a^* b^*$

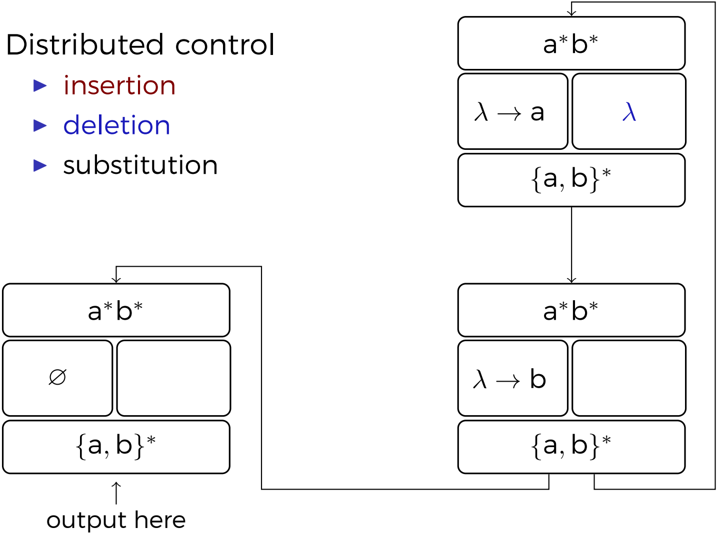
$\lambda \to b$

$\{a, b\}^*$

Networks of Evolutionary Processors (NEPs)

Distributed control

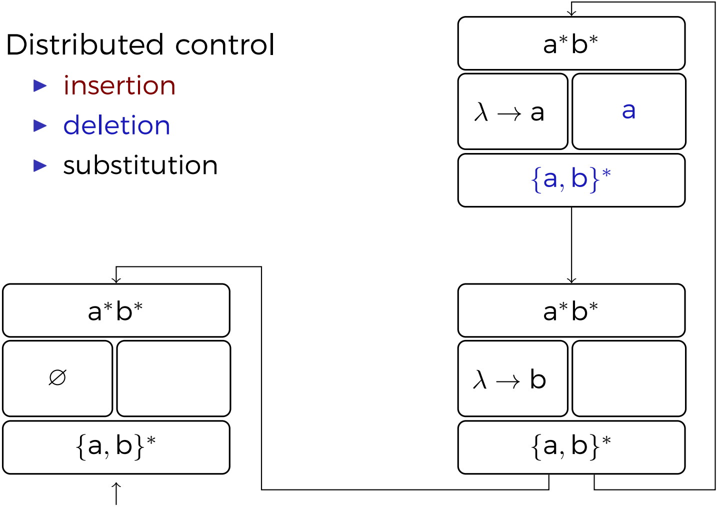
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

- insertion
- deletion
- substitution



a^*b^*

$\lambda \to a$

a

$\{a,b\}^*$

a^*b^*

$\lambda \to b$

$\{a,b\}^*$

a^*b^*

$\varnothing$

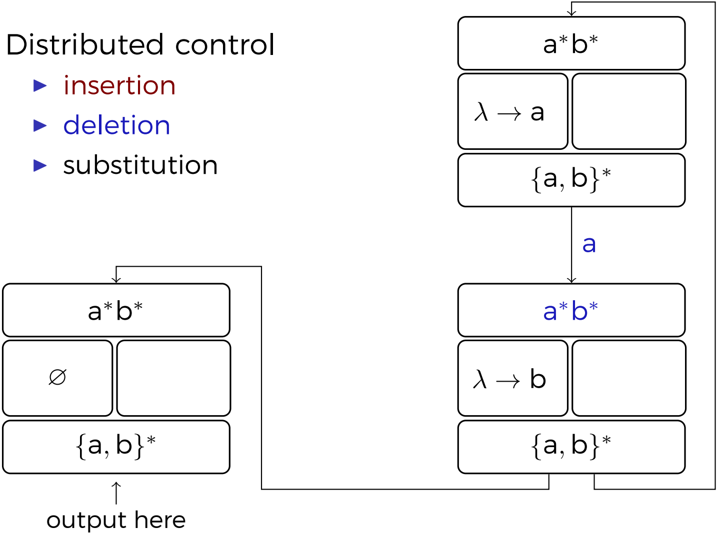
$\{a,b\}^*$

output here

Networks of Evolutionary Processors (NEPs)

Distributed control

- insertion
- deletion
- substitution



a^*b^*

$\lambda \to a$

$\{a,b\}^*$

a

a^*b^*

$\lambda \to b$

$\{a,b\}^*$

a^*b^*

$\varnothing$

$\{a,b\}^*$

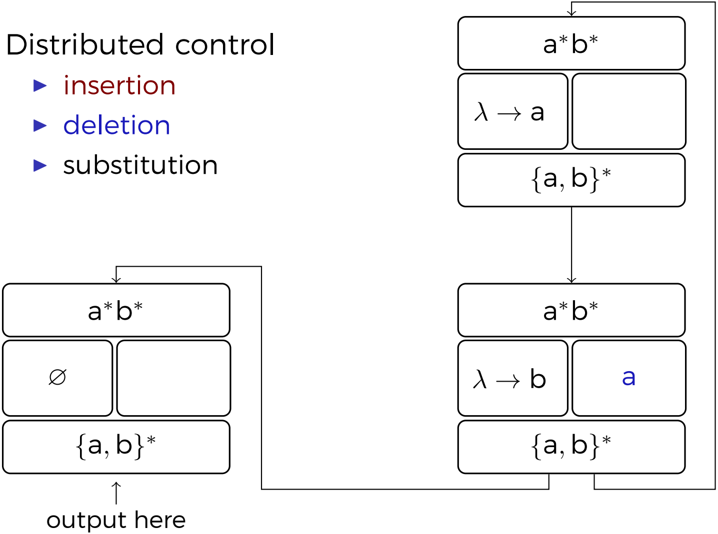
↑

output here

Networks of Evolutionary Processors (NEPs)

Distributed control

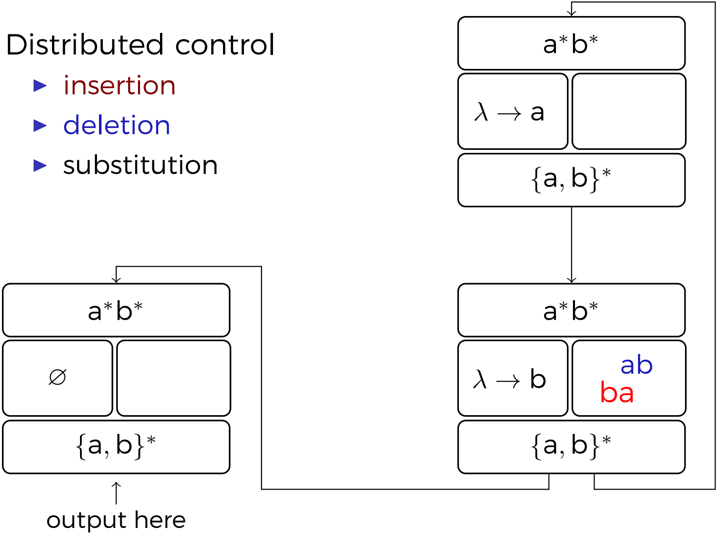
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

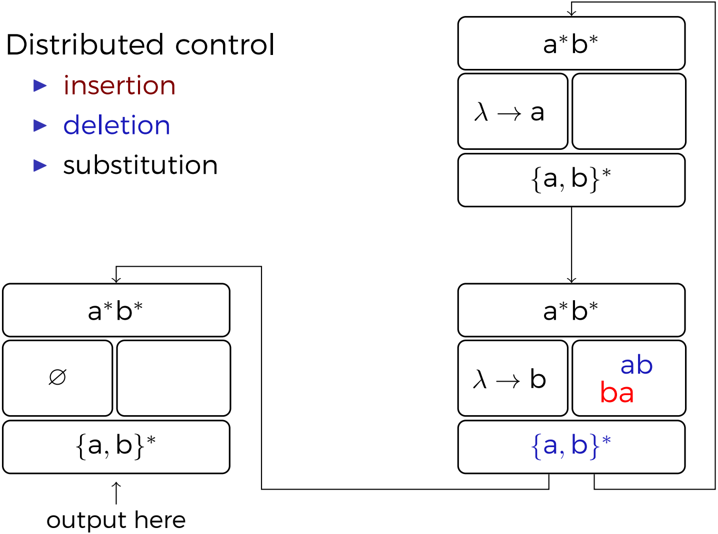
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

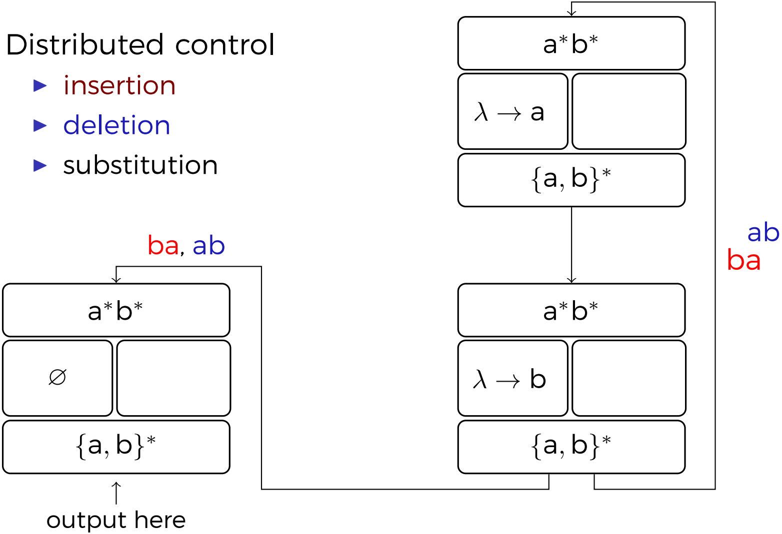
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

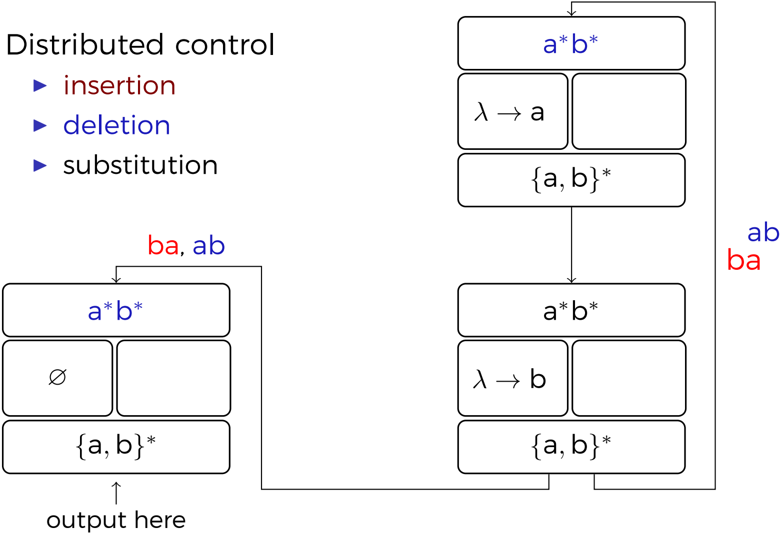
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

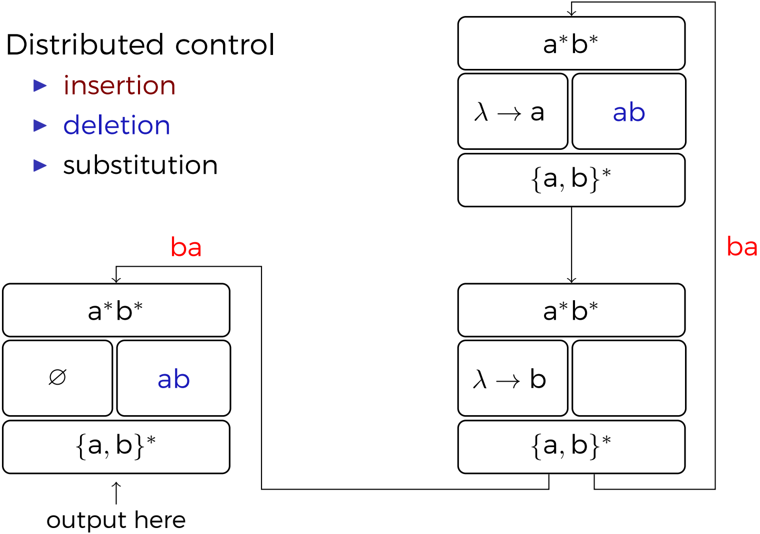
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

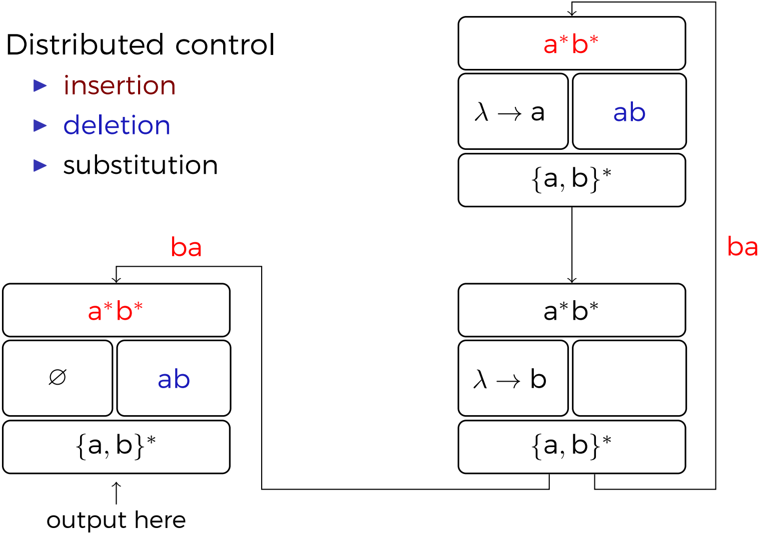
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

Distributed control

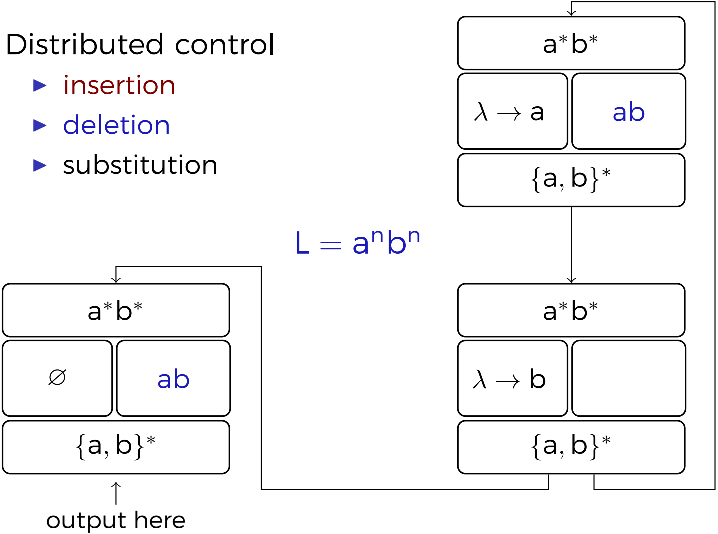
- insertion
- deletion
- substitution



Networks of Evolutionary Processors (NEPs)

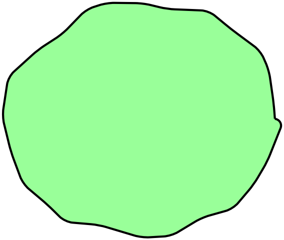
Distributed control

- insertion
- deletion
- substitution



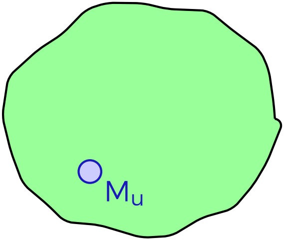
$L = a^n b^n$

Universality



computing devices

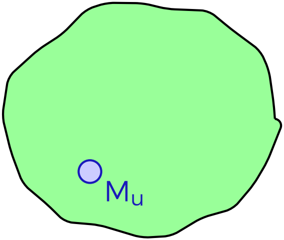
Universality



computing devices

Universality

M_u universal:

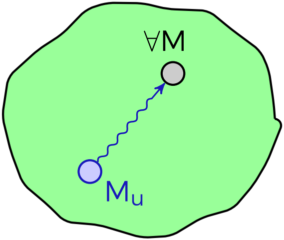


computing devices

Universality

M_u universal:

M_u can simulate any other M



computing devices

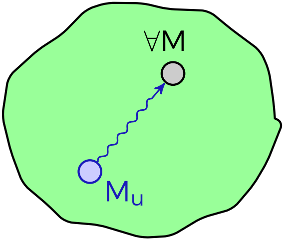
Universality

M_u universal:

M_u can simulate any other M

$$h(x)$$

- h – coding function



computing devices

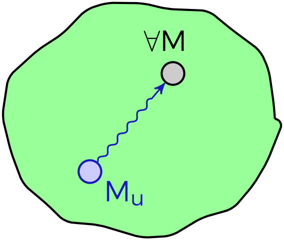
Universality

M_u universal:

M_u can simulate any other M

$$h(x), g(M)$$

- h – coding function
- g – Gödel enumeration



computing devices

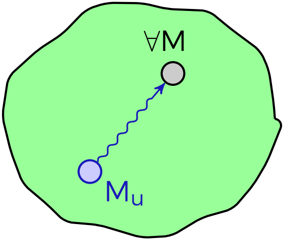
Universality

M_u universal:

M_u can simulate any other M

$$\langle h(x), g(M) \rangle$$

- h – coding function
- g – Gödel enumeration
- $\langle a, b \rangle$ – pairing function



computing devices

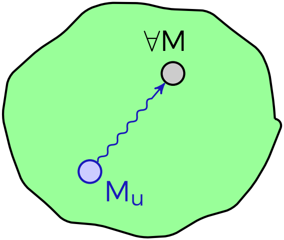
Universality

M_u universal:

M_u can simulate any other M

$M_u(\langle h(x), g(M) \rangle)$

- h – coding function
- g – Gödel enumeration
- $\langle a, b \rangle$ – pairing function



computing devices

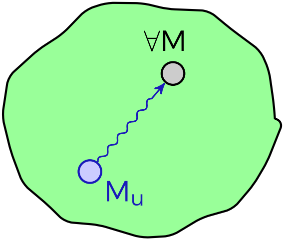
Universality

M_u universal:

M_u can simulate any other M

$$f\big(\, M_u(\, \langle h(x), g(M) \rangle \,)\,\big)$$

- h – coding function
- g – Gödel enumeration
- $\langle a, b \rangle$ – pairing function
- f – decoding function



computing devices

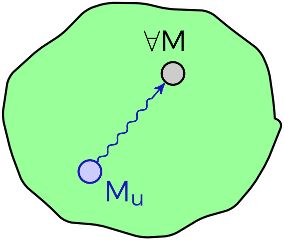
Universality

M_u universal:

M_u can simulate any other M

$$f\big(\, M_u(\, \langle h(x), g(M) \rangle \,)\, \big) = M(x)$$

- h – coding function
- g – Gödel enumeration
- $\langle a, b \rangle$ – pairing function
- f – decoding function



computing devices

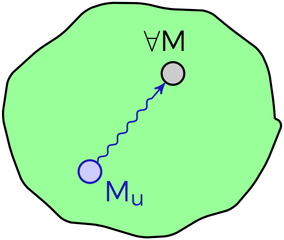
Universality

M_u universal:

M_u can simulate any other M

$$f(\,M_u(\,\langle h(x), g(M)\rangle\,)\,) = M(x)$$

- h – coding function
- g – Gödel enumeration
- $\langle a, b\rangle$ – pairing function
- f – decoding function



computing devices

$f = h = id \quad \Longrightarrow \quad M_u$ – strongly universal

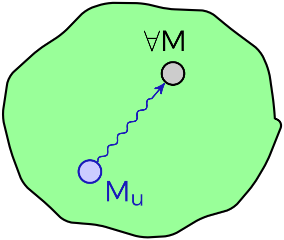
Universality

M_u universal:

M_u can simulate any other M

$$f(\,M_u(\,\langle h(x), g(M)\rangle\,)\,) = M(x)$$

- h – coding function
- g – Gödel enumeration
- $\langle a, b\rangle$ – pairing function
- f – decoding function



computing devices

$f = h = id \implies$ M_u – strongly universal

otherwise $\implies$ M_u – weakly universal

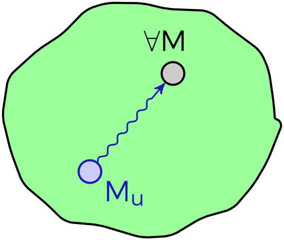
Universality

M_u universal:

M_u can simulate any other M

$$f(\,M_u(\,\langle h(x), g(M)\rangle\,)\,) = M(x)$$

- h – coding function
- g – Gödel enumeration
- $\langle a, b\rangle$ – pairing function
- f – decoding function



computing devices

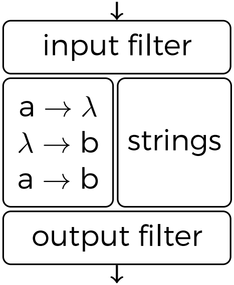
$f = h = id \implies M_u$ – strongly universal

$\text{otherwise} \implies M_u$ – weakly universal

Distinction especially important for numbers

Universal NEPs

Minimise the number of rules



Evolutionary processor

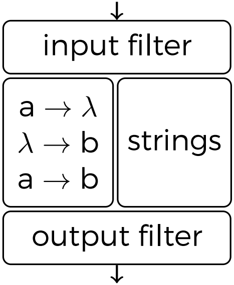
Universal NEPs

Minimise the number of rules

5 rules: strongly universal

4 rules: weakly universal

- 3 nodes



input filter

$a \to \lambda$

$\lambda \to b$

$a \to b$

strings

output filter

Evolutionary processor

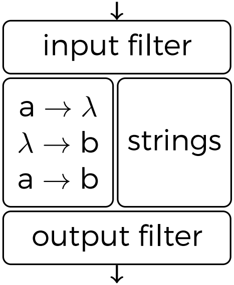
Universal NEPs

Minimise the number of rules

5 rules: strongly universal

4 rules: weakly universal

- 3 nodes
- simulate register machines



Evolutionary processor

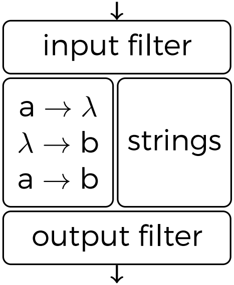
Universal NEPs

Minimise the number of rules

5 rules: strongly universal

4 rules: weakly universal

- 3 nodes
- simulate register machines
- exponential slowdown



Evolutionary processor

Universal NEPs

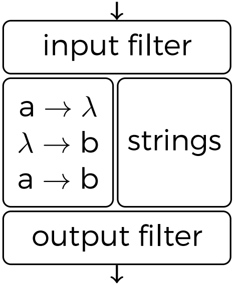
Minimise the number of rules

5 rules: strongly universal

4 rules: weakly universal

- 3 nodes
- simulate register machines
- exponential slowdown

7 rules



Evolutionary processor

Universal NEPs

Minimise the number of rules

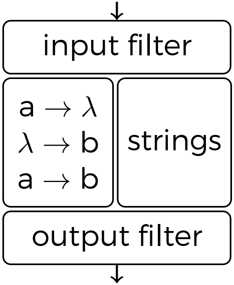
5 rules: strongly universal

4 rules: weakly universal

- 3 nodes
- simulate register machines
- exponential slowdown

7 rules

- 4 nodes
- simulate Turing machines
- polynomial slowdown



Evolutionary processor

Presentation Map

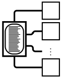
Insertion and Deletion

One-sided insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control

Multiset Rewriting

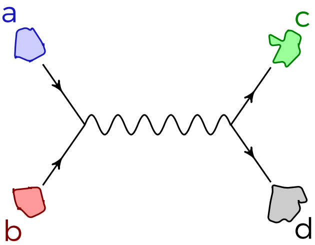
Small universal register machines



Small universal Petri nets

Biochemical Reactions as Multiset Rewriting

Biochemical reaction

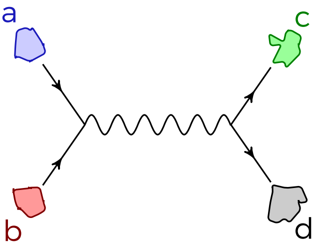


Biochemical Reactions as Multiset Rewriting

Multiset rewriting

$a \; b \;\rightarrow\; c \; d$

Biochemical reaction

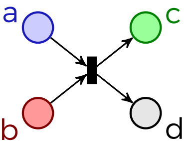


Biochemical Reactions as Multiset Rewriting

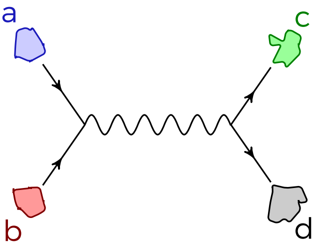
Multiset rewriting

$$a\ b \to c\ d$$

Petri nets



Biochemical reaction

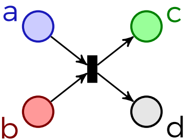


Biochemical Reactions as Multiset Rewriting

Multiset rewriting

$$a\ b \rightarrow c\ d$$

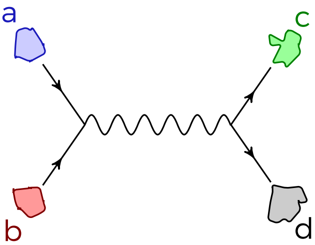
Petri nets



Register machines

a	b	c	d
-1	-1	$+1$	$+1$

Biochemical reaction

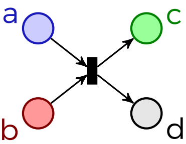


Biochemical Reactions as Multiset Rewriting

Multiset rewriting

$$a\ b \rightarrow c\ d$$

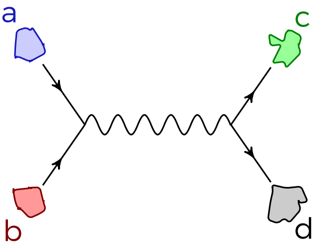
Petri nets



Register machines

a	b	c	d
-1	-1	$+1$	$+1$

Biochemical reaction



Universality

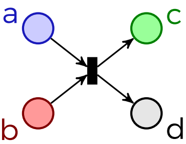
- small systems

Biochemical Reactions as Multiset Rewriting

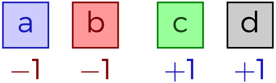
Multiset rewriting

$$a\ b \rightarrow c\ d$$

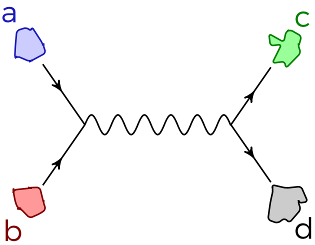
Petri nets



Register machines



Biochemical reaction



Universality

- small systems

Presentation as Petri nets

Presentation Map

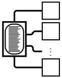
Insertion and Deletion

One-sided insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control

Multiset Rewriting

Small universal register machines

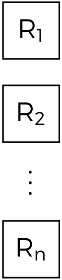


- Universal register machines with 3 and 2 registers
- Generalised register machines

Small universal Petri nets

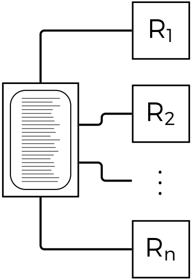
Register Machines

- Registers
  - nonnegative integers



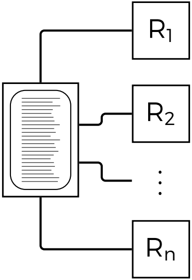
Register Machines

- Registers
  - nonnegative integers
- Instructions
  - with labels/states



Register Machines

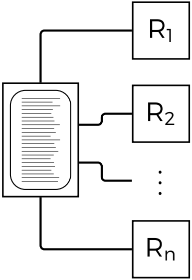
- Registers
  - nonnegative integers
- Instructions
  - with labels/states



Increment, (p, RiP, q):

Register Machines

- Registers
  - nonnegative integers
- Instructions
  - with labels/states

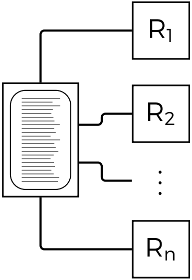


Increment, (p, RiP, q): $\longrightarrow\overset{p}{\boxed{\text{RiP}}}\rightarrow q$

Decrement, (p, RiM, q): $\longrightarrow\overset{p}{\boxed{\text{RiM}}}\rightarrow q$

Register Machines

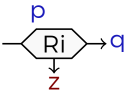
- Registers
  - nonnegative integers
- Instructions
  - with labels/states



Increment, (p, RiP, q):

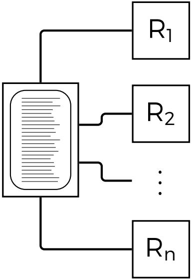
Decrement, (p, RiM, q):

Zero check, (p, Ri, q, z):



Register Machines

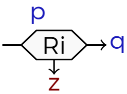
- Registers
  - nonnegative integers
- Instructions
  - with labels/states



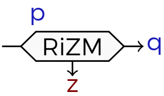
Increment, (p, RiP, q):

Decrement, (p, RiM, q):

Zero check, (p, Ri, q, z):



Ri & RiM, (p, RiZM, q, z):



Register Machines

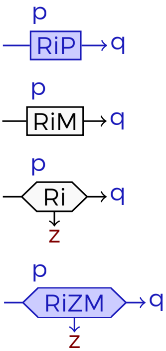
- Registers
  - nonnegative integers
- Instructions
  - with labels/states

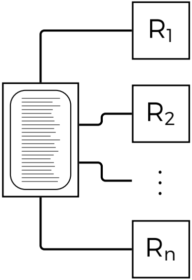
Increment, (p, RiP, q):

Decrement, (p, RiM, q):

Zero check, (p, Ri, q, z):

Ri & RiM, (p, RiZM, q, z):

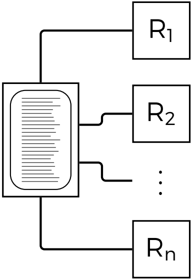




RiP and RiZM most often used

Register Machines

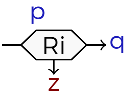
- Registers
  - nonnegative integers
- Instructions
  - with labels/states



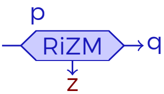
Increment, (p, RiP, q):

Decrement, (p, RiM, q):

Zero check, (p, Ri, q, z):



Ri & RiM, (p, RiZM, q, z):



RiP and RiZM most often used

Computationally complete

Universal Register Machines

$\exists$ universal register machines (RiP and RiZM)

- strongly universal U_{22} with 22 instructions
  - 8 registers
- weakly universal U_{20} with 20 instructions
  - 7 registers

[I. Korec 1996]

Universal Register Machines

∃ universal register machines (RiP and RiZM)

- strongly universal U_{22} with 22 instructions
  - 8 registers
- weakly universal U_{20} with 20 instructions
  - 7 registers

[I. Korec 1996]

Reduce the number of registers to 2 [M. Minsky 1967]

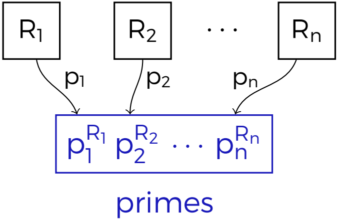
Universal Register Machines

∃ universal register machines (RiP and RiZM)

- strongly universal U_{22} with 22 instructions
  - 8 registers
- weakly universal U_{20} with 20 instructions
  - 7 registers

[I. Korec 1996]

Reduce the number of registers to 2 [M. Minsky 1967]



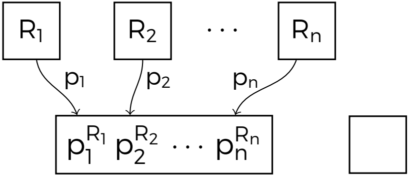
Universal Register Machines

∃ universal register machines (RiP and RiZM)

- strongly universal U_{22} with 22 instructions
  - 8 registers
- weakly universal U_{20} with 20 instructions
  - 7 registers

[I. Korec 1996]

Reduce the number of registers to 2 [M. Minsky 1967]



primes

for × and /
(RiP and RiZM)

Universal Register Machines

$\exists$ universal register machines (RiP and RiZM)

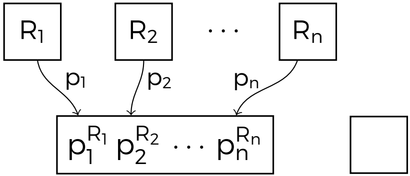
- strongly universal U_{22} with 22 instructions
  - 8 registers
- weakly universal U_{20} with 20 instructions
  - 7 registers

[I. Korec 1996]

Reduce the number of registers to 2 [M. Minsky 1967]

We constructed:

- strongly universal U_3
  - 3 registers
  - 367 instructions



primes

for × and /
(RiP and RiZM)

Universal Register Machines

∃ universal register machines (→RiP→ and →RiZM→)

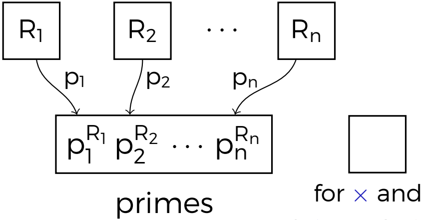
- strongly universal U_{22} with 22 instructions
  - 8 registers
- weakly universal U_{20} with 20 instructions
  - 7 registers

[I. Korec 1996]

Reduce the number of registers to 2 [M. Minsky 1967]

We constructed:

- strongly universal U_3
  - 3 registers
  - 367 instructions
- weakly universal U_2
  - 2 registers
  - 277 instructions



for × and /

(RiP and RiZM)

Presentation Map

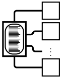
Insertion and Deletion

One-sided insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control

Multiset Rewriting

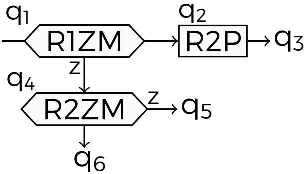
Small universal register machines



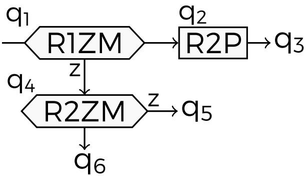
- Universal register machines with 3 and 2 registers
- Generalised register machines

Small universal Petri nets

Generalised Register Machines

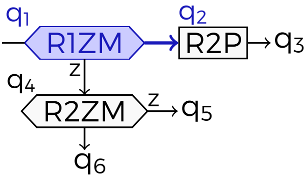


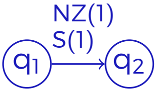
Generalised Register Machines



- move actions to edges

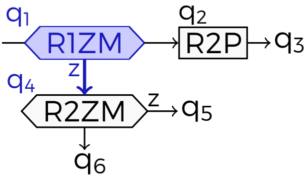
Generalised Register Machines

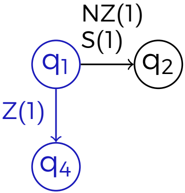




- move actions to edges

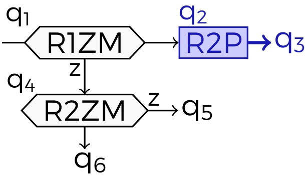
Generalised Register Machines

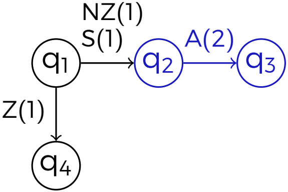




- move actions to edges

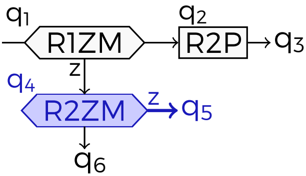
Generalised Register Machines

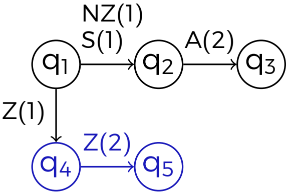




- move actions to edges

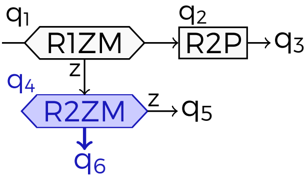
Generalised Register Machines



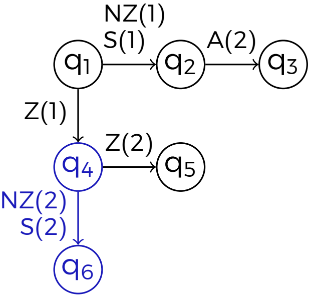


- move actions to edges

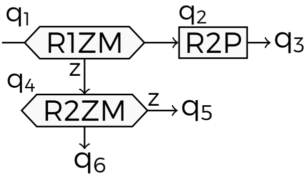
Generalised Register Machines



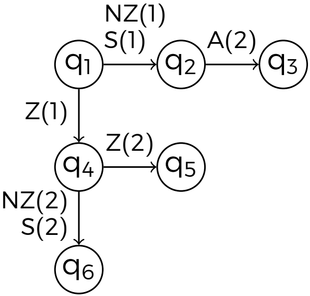
- move actions to edges



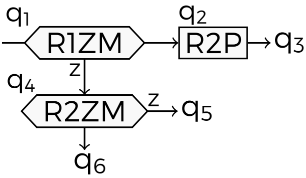
Generalised Register Machines



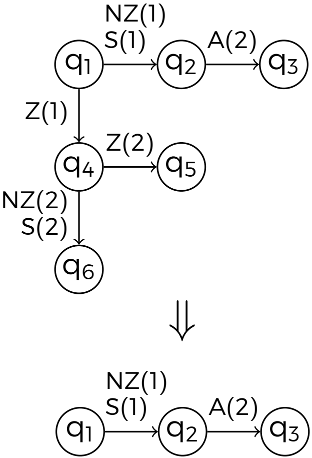
- move actions to edges
- allow multiple actions



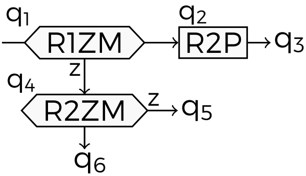
Generalised Register Machines



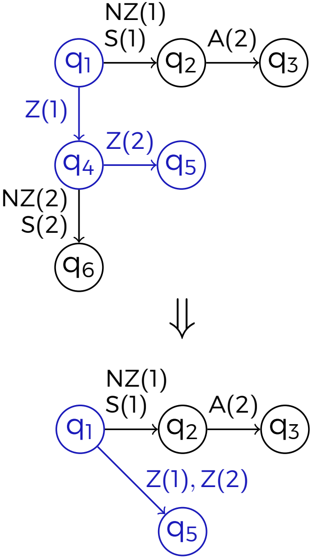
- move actions to edges
- allow multiple actions



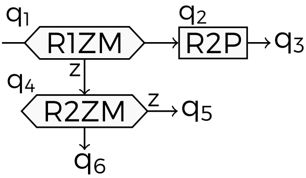
Generalised Register Machines



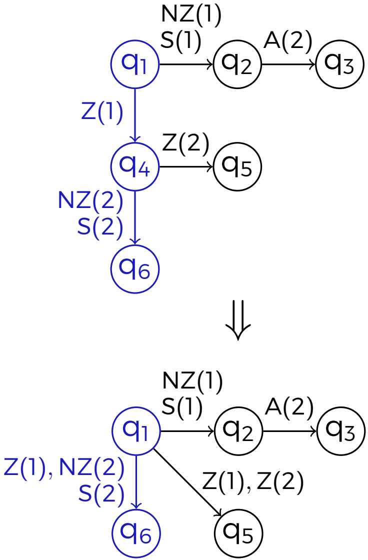
- move actions to edges
- allow multiple actions



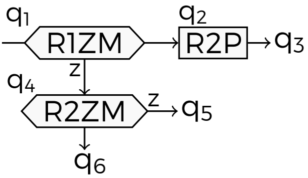
Generalised Register Machines



- move actions to edges
- allow multiple actions

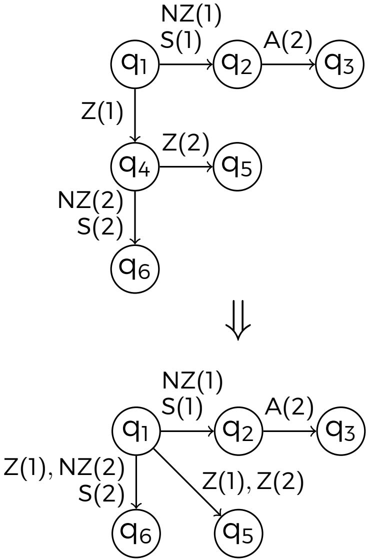


Generalised Register Machines

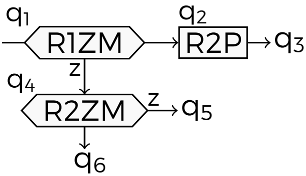


- move actions to edges
- allow multiple actions

State compression



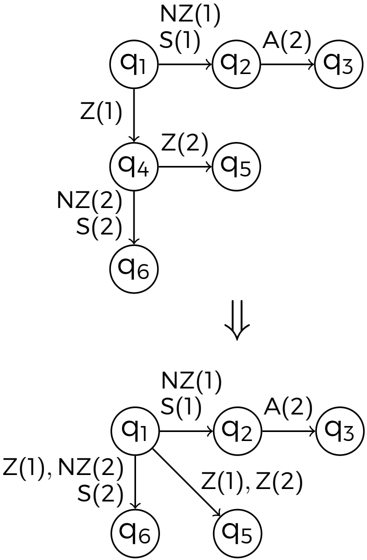
Generalised Register Machines



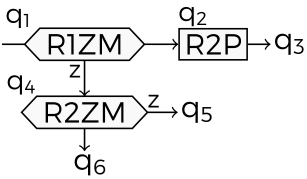
- move actions to edges
- allow multiple actions

State compression

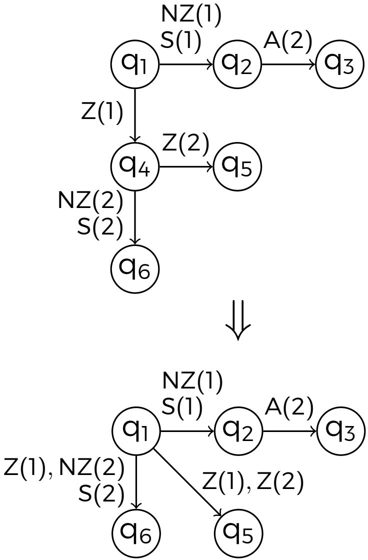
- $U_{22} \Rightarrow$ strongly universal U_7
  - 7 states



Generalised Register Machines



- move actions to edges
- allow multiple actions



State compression

- $U_{22} \Rightarrow$ strongly universal U_7
  - 7 states
- $U_{20} \Rightarrow$ weakly universal U'_7
  - 7 states

Presentation Map

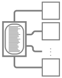
Insertion and Deletion

One-sided insertion-deletion systems $(u, x, v)_{ins/del}$

Insertion-deletion systems with control

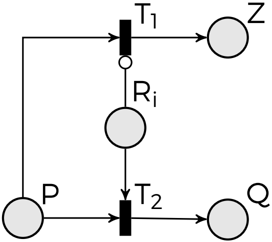
Multiset Rewriting

Small universal register machines



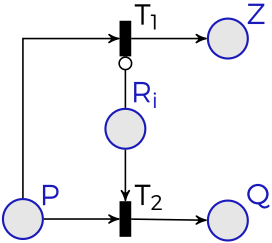
Small universal Petri nets

Petri Nets with Inhibitor Arcs



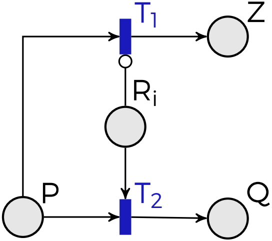
Petri Nets with Inhibitor Arcs

- places



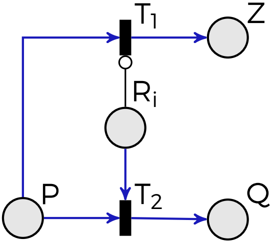
Petri Nets with Inhibitor Arcs

- places
- transitions



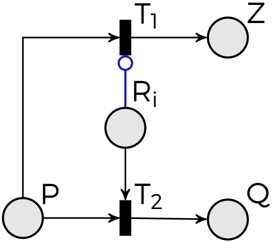
Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs



Petri Nets with Inhibitor Arcs

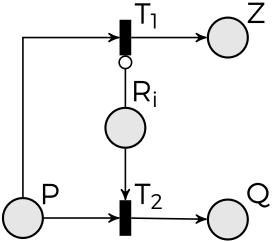
- places
- transitions
- normal arcs
- inhibitor arcs



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

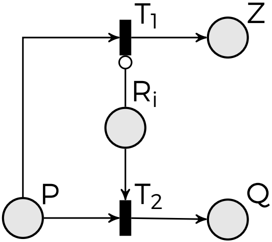
Size = (p, t, i, d)



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

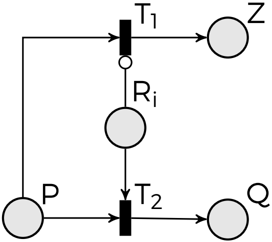
Size = (p, t, i, d)



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

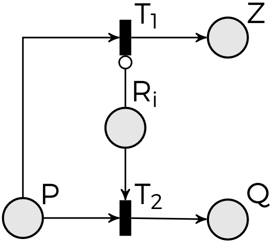
Size = (p, t, i, d)



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

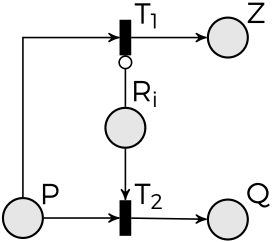
Size = (p, t, i, d)



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

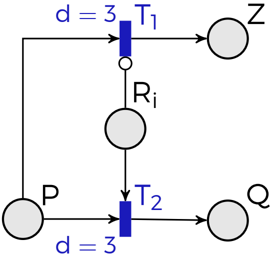
Size = (p, t, i, d)



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

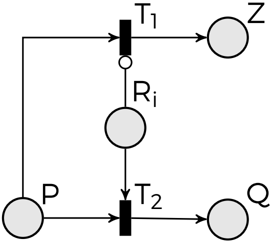
Size = (p, t, i, d)



Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

Size = (p, t, i, d)



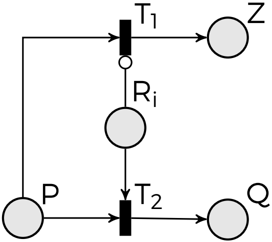
Build small universal Petri nets

- result in halting configuration

Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

Size = (p, t, i, d)



Simulates (p, RiZM, q, z)

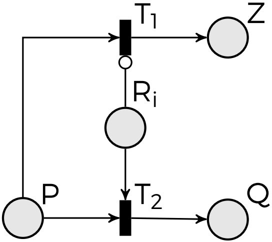
Build small universal Petri nets

- result in halting configuration

Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

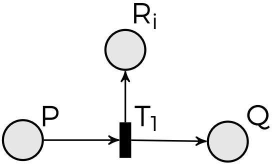
Size = (p, t, i, d)



Simulates (p, RiZM, q, z)

Build small universal Petri nets

- result in halting configuration

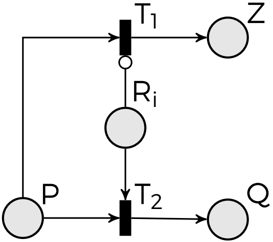


Simulates (p, RiP, q)

Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

Size = (p, t, i, d)

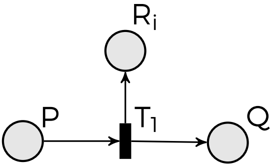


Simulates (p, RiZM, q, z)

Build small universal Petri nets

- result in halting configuration

Direct simulation of U_{22} and U_{20}

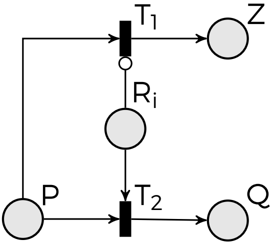


Simulates (p, RiP, q)

Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

Size = (p, t, i, d)



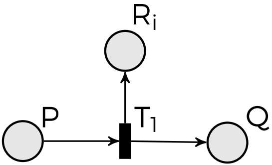
Simulates (p, RiZM, q, z)

Build small universal Petri nets

- result in halting configuration

Direct simulation of U_{22} and U_{20}

- strongly universal
  (p : 30, t : 34, i : 12, d : 3)
- weakly universal
  (p : 27, t : 31, i : 11, d : 3)

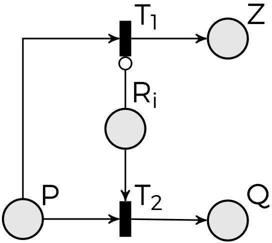


Simulates (p, RiP, q)

Petri Nets with Inhibitor Arcs

- places
- transitions
- normal arcs
- inhibitor arcs

Size = (p, t, i, d)



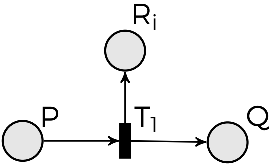
Simulates (p, RiZM, q, z)

Build small universal Petri nets

- result in halting configuration

Direct simulation of U_{22} and U_{20}

- strongly universal
  $(p : 30, t : 34, i : 12, d : 3)$
- weakly universal
  $(p : 27, t : 31, i : 11, d : 3)$



Simulates (p, RiP, q)

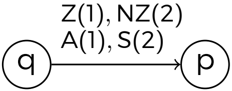
Minimal transition degree

Minimising the Number of Transitions

Simulate compressed generalised register machines

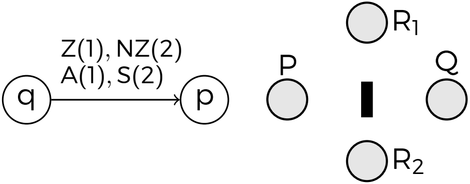
Minimising the Number of Transitions

Simulate compressed generalised register machines



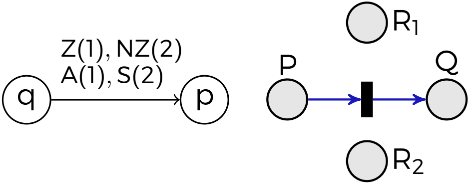
Minimising the Number of Transitions

Simulate compressed generalised register machines



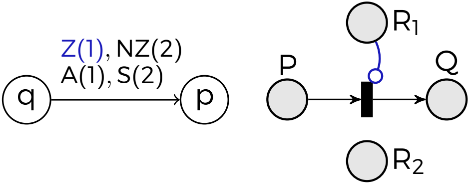
Minimising the Number of Transitions

Simulate compressed generalised register machines



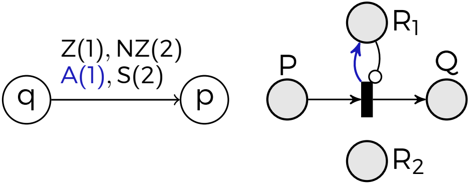
Minimising the Number of Transitions

Simulate compressed generalised register machines



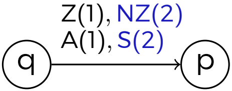
Minimising the Number of Transitions

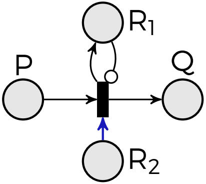
Simulate compressed generalised register machines



Minimising the Number of Transitions

Simulate compressed generalised register machines





Minimising the Number of Transitions

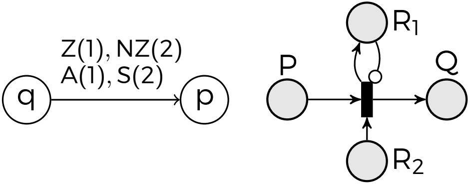
Simulate compressed generalised register machines

- strongly universal

  $(p : 14, t : 23, i : 30, d : 6)$

- weakly universal

  $(p : 13, t : 21, i : 23, d : 6)$



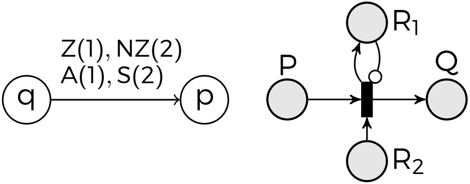
Minimising the Number of Transitions

Simulate compressed generalised register machines

- strongly universal

  (p : 14, t : 23, i : 30, d : 6)
- weakly universal

  (p : 13, t : 21, i : 23, d : 6)

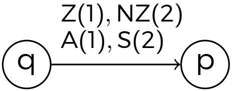


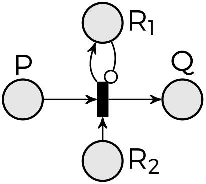
Binary-code the states

Minimising the Number of Transitions

Simulate compressed generalised register machines

- strongly universal
  $(p : 14, t : 23, i : 30, d : 6)$
- weakly universal
  $(p : 13, t : 21, i : 23, d : 6)$





Binary-code the states

(q_4, RiP, q_6)

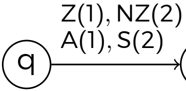
Minimising the Number of Transitions

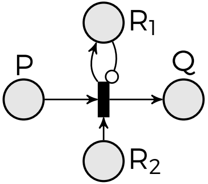
Simulate compressed generalised register machines

- strongly universal

  $(p : 14, t : 23, i : 30, d : 6)$
- weakly universal

  $(p : 13, t : 21, i : 23, d : 6)$





Binary-code the states

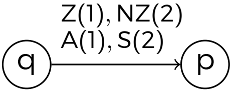
(q_4, RiP, q_6)

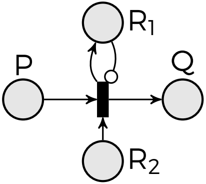
$(100)_2$ $(110)_2$

Minimising the Number of Transitions

Simulate compressed generalised register machines

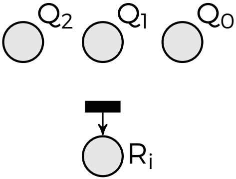
- strongly universal
  $(p : 14, t : 23, i : 30, d : 6)$
- weakly universal
  $(p : 13, t : 21, i : 23, d : 6)$





Binary-code the states

(q_4, RiP, q_6)
$(100)_2 \quad (110)_2$



Minimising the Number of Transitions

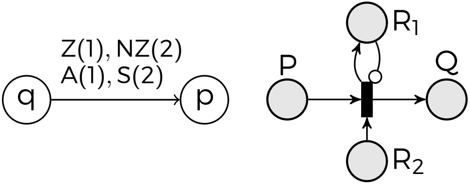
Simulate compressed generalised register machines

- strongly universal

  $(p : 14, t : 23, i : 30, d : 6)$

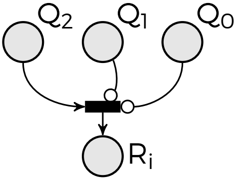
- weakly universal

  $(p : 13, t : 21, i : 23, d : 6)$



Binary-code the states

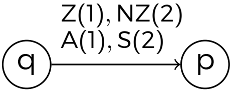
(q_4, RiP, q_6)
$(100)_2 \quad (110)_2$

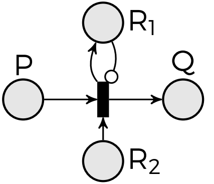


Minimising the Number of Transitions

Simulate compressed generalised register machines

- strongly universal
  ($p:14, t:23, i:30, d:6$)
- weakly universal
  ($p:13, t:21, i:23, d:6$)

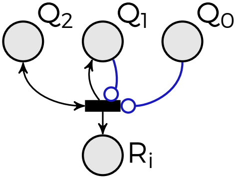




Binary-code the states

(q_4, RiP, q_6)

$(100)_2 \quad (110)_2$



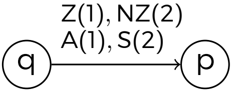
Minimising the Number of Transitions

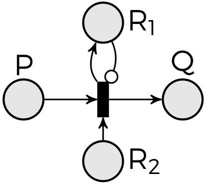
Simulate compressed generalised register machines

- strongly universal

  $(p : 14, t : 23, i : 30, d : 6)$
- weakly universal

  $(p : 13, t : 21, i : 23, d : 6)$





Binary-code the states

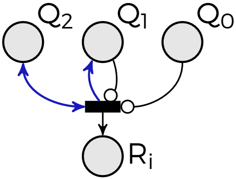
- strongly universal

  $(p : 11, t : 23, i : 37, d : 10)$
- weakly universal

  $(p : 10, t : 21, i : 30, d : 10)$

(q_4, RiP, q_6)

$(100)_2 \quad (110)_2$

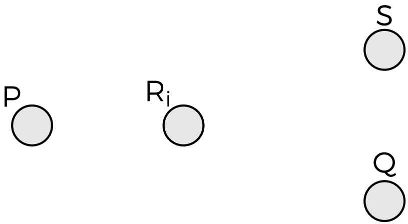


Minimising the Number of Inhibitor Arcs

$$(p, RiZM, q, s)$$

Minimising the Number of Inhibitor Arcs

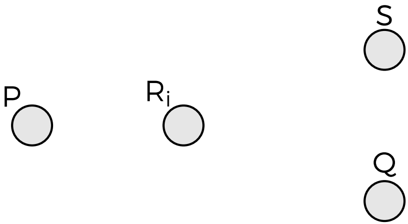
$(p, RiZM, q, s)$



Minimising the Number of Inhibitor Arcs

Factor out the inhibitor arc

$(p, RiZM, q, s)$

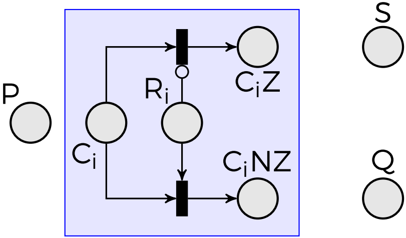


Minimising the Number of Inhibitor Arcs

Factor out the inhibitor arc

- checker subnets

$(p, RiZM, q, s)$

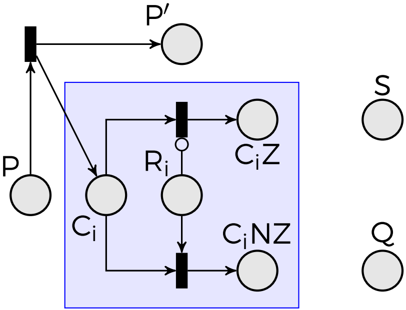


Minimising the Number of Inhibitor Arcs

Factor out the inhibitor arc

- checker subnets

$(p, RiZM, q, s)$

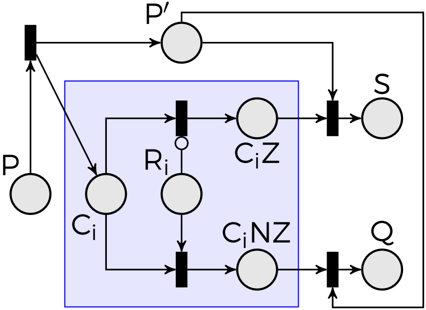


Minimising the Number of Inhibitor Arcs

Factor out the inhibitor arc

- checker subnets

$(p, RiZM, q, s)$



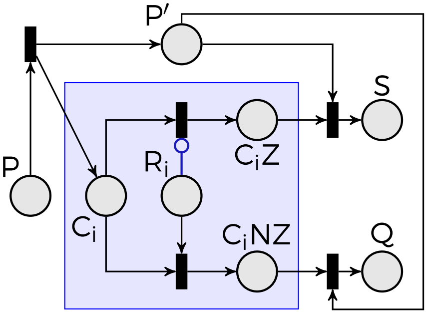
Minimising the Number of Inhibitor Arcs

Factor out the inhibitor arc

- checker subnets

One inhibitor per register

$(p, RiZM, q, s)$



Minimising the Number of Inhibitor Arcs

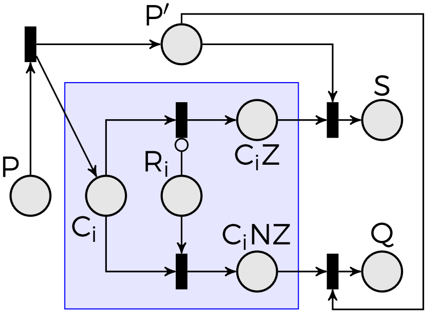
Factor out the inhibitor arc

- checker subnets

One inhibitor per register

Simulate U_3 and U_2

$(p, RiZM, q, s)$



Minimising the Number of Inhibitor Arcs

Factor out the inhibitor arc

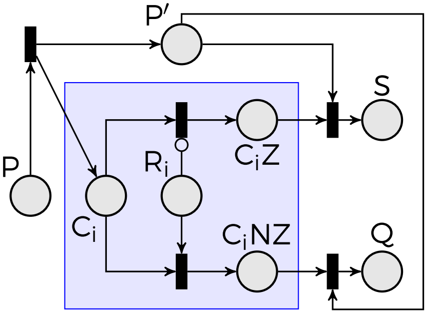
- checker subnets

One inhibitor per register

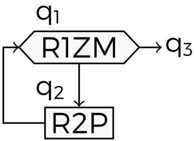
Simulate U_3 and U_2

- strong universality
  $(p : 525, t : 659, i : 3, d : 3)$
- weak universality
  $(p : 397, t : 504, i : 2, d : 3)$

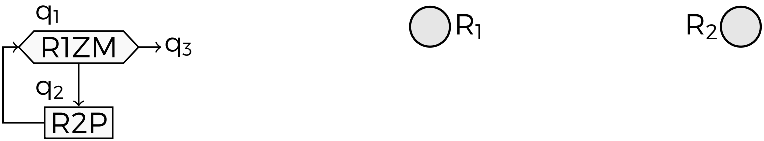
$(p, RiZM, q, s)$



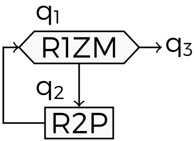
Minimising the Number of Places

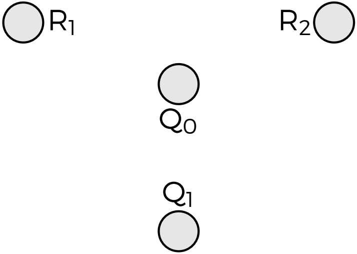


Minimising the Number of Places

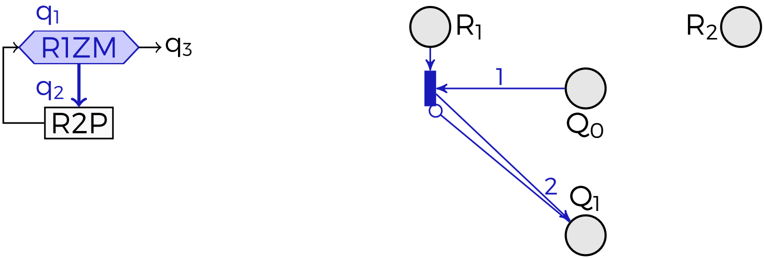


Minimising the Number of Places

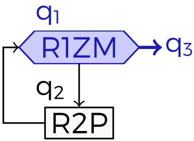


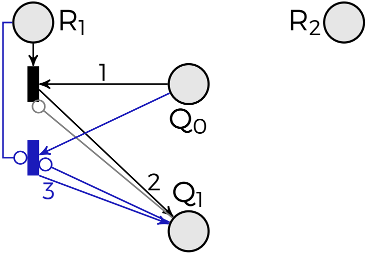


Minimising the Number of Places

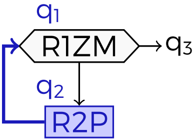


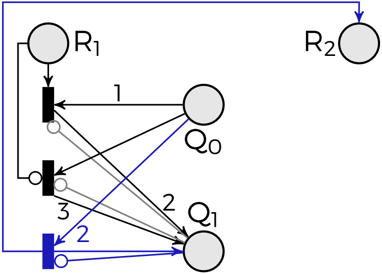
Minimising the Number of Places



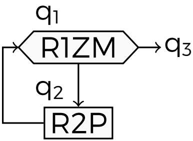


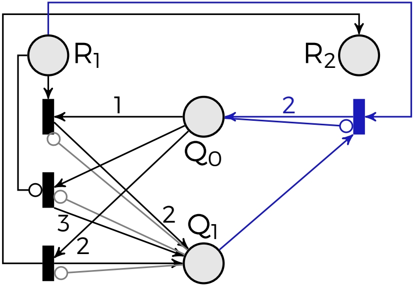
Minimising the Number of Places



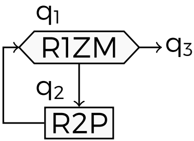


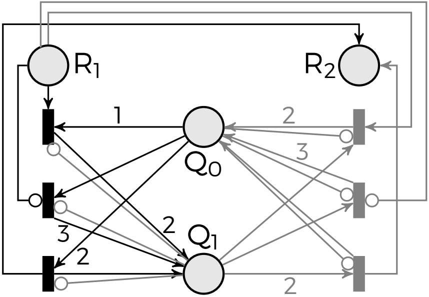
Minimising the Number of Places



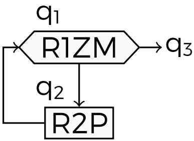


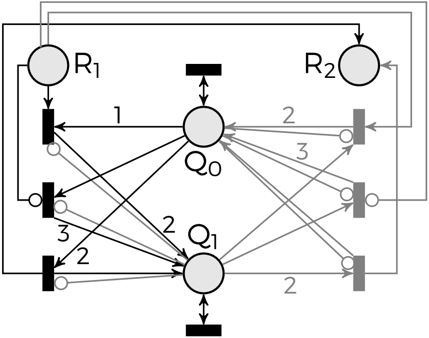
Minimising the Number of Places



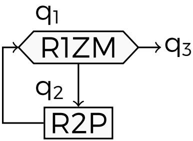


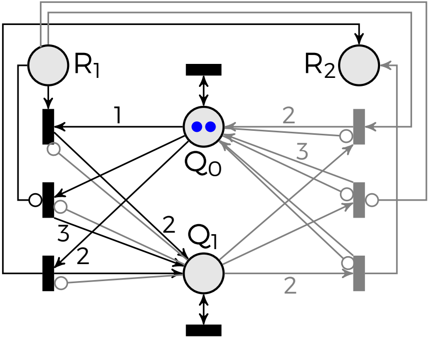
Minimising the Number of Places





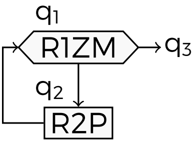
Minimising the Number of Places

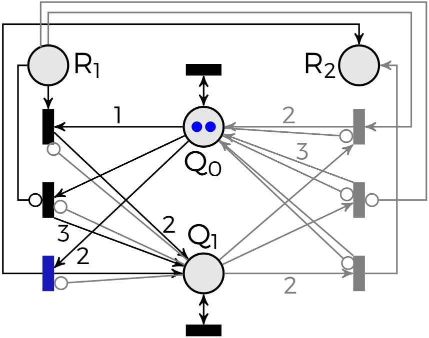




nondeterminism

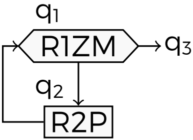
Minimising the Number of Places

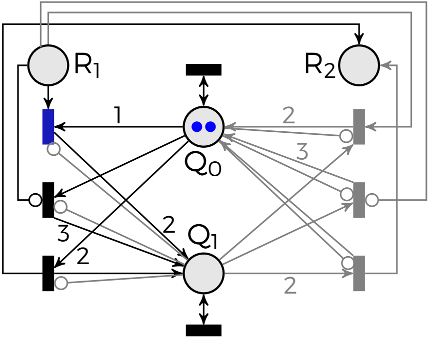




nondeterminism

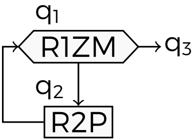
Minimising the Number of Places

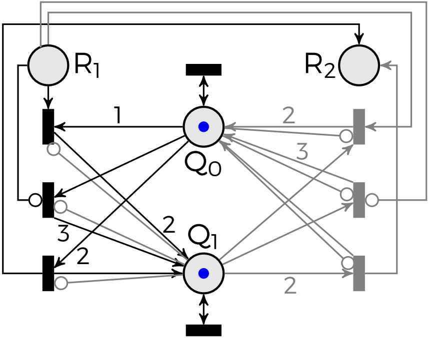




nondeterminism

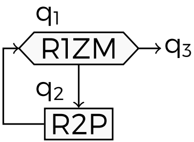
Minimising the Number of Places



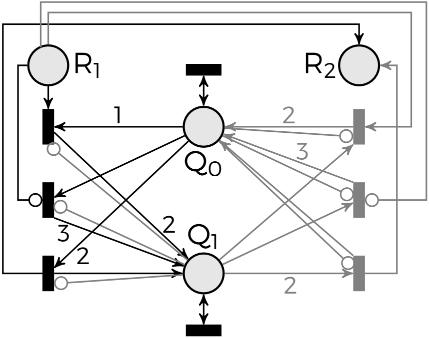


nondeterminism

Minimising the Number of Places

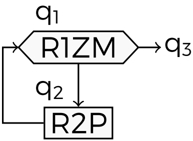


$$\#\text{places} = \#\text{registers} + 2$$



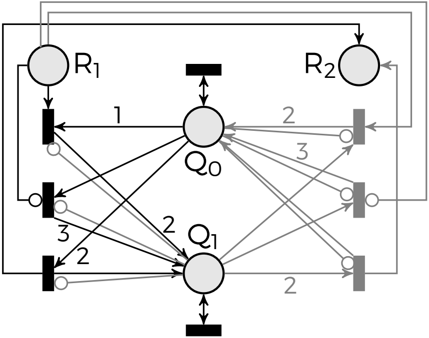
nondeterminism

Minimising the Number of Places



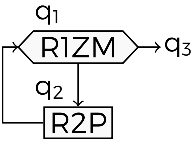
$\#\text{places} = \#\text{registers} + 2$

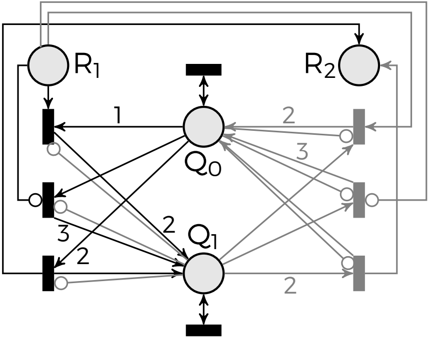
$\text{Max degree} = f(\text{state coding})$



nondeterminism

Minimising the Number of Places





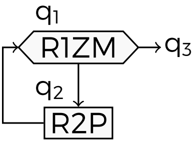
$\#\text{places} = \#\text{registers} + 2$

$\text{Max degree} = f(\text{state coding})$

$\text{cost}(q_i \rightarrow q_j) = \text{code}(i) + \text{code}(j)$

nondeterminism

Minimising the Number of Places

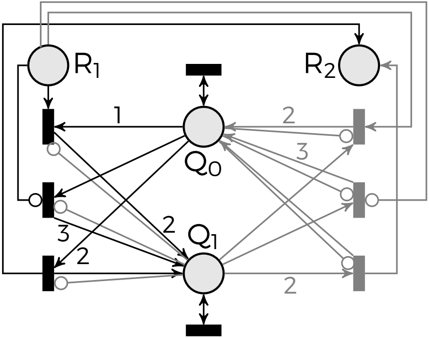


$$\#\text{places} = \#\text{registers} + 2$$

$$\text{Max degree} = f(\text{state coding})$$

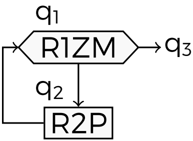
$$\text{cost}(q_i \to q_j) = \text{code}(i) + \text{code}(j)$$

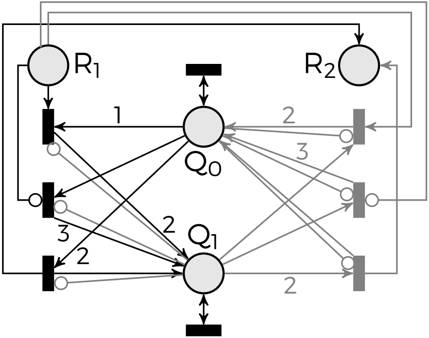
minimise worst cost



nondeterminism

Minimising the Number of Places





$$\#\text{places} = \#\text{registers} + 2$$

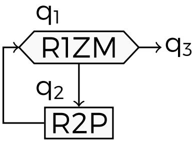
$$\text{Max degree} = f(\text{state coding})$$

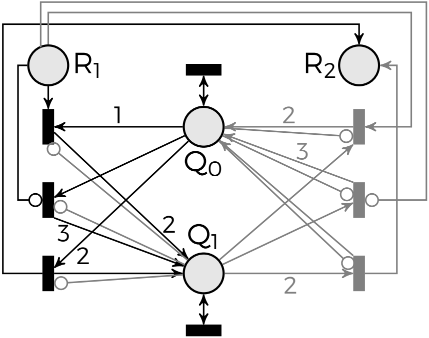
$$\text{cost}(q_i \to q_j) = \text{code}(i) + \text{code}(j)$$

$$\begin{array}{rl} \text{minimise} & \text{worst cost} \\ \text{subject to} & \text{cost}(q_i \to q_j) < \text{worst cost} \end{array}$$

nondeterminism

Minimising the Number of Places





$\#\text{places} = \#\text{registers} + 2$

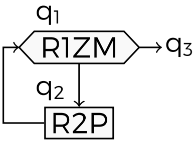
$\text{Max degree} = f(\text{state coding})$

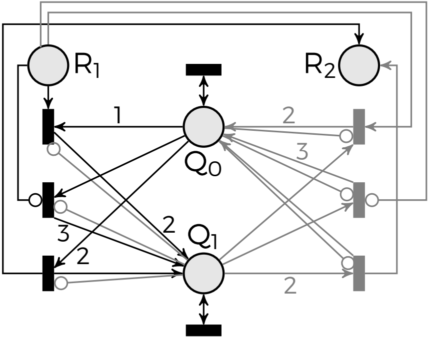
$\text{cost}(q_i \to q_j) = \text{code}(i) + \text{code}(j)$

minimise worst cost
subject to $\text{cost}(q_i \to q_j) <$ worst cost
one code per state, one state per code

nondeterminism

Minimising the Number of Places





$\#\text{places} = \#\text{registers} + 2$

$\text{Max degree} = f(\text{state coding})$

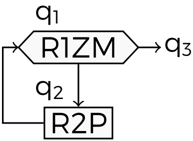
$\text{cost}(q_i \to q_j) = \text{code}(i) + \text{code}(j)$

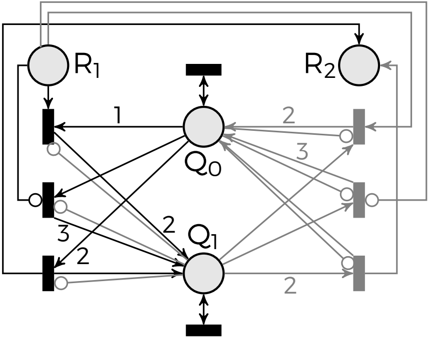
minimise worst cost
subject to $\text{cost}(q_i \to q_j) <$ worst cost
one code per state, one state per code

nondeterminism

Simulate U_3 and U_2

Minimising the Number of Places





$\#\text{places} = \#\text{registers} + 2$

Max degree $=$ f(state coding)

$\text{cost}(q_i \to q_j) = \text{code}(i) + \text{code}(j)$

minimise worst cost

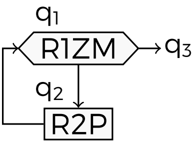
subject to $\text{cost}(q_i \to q_j) <$ worst cost
one code per state, one state per code

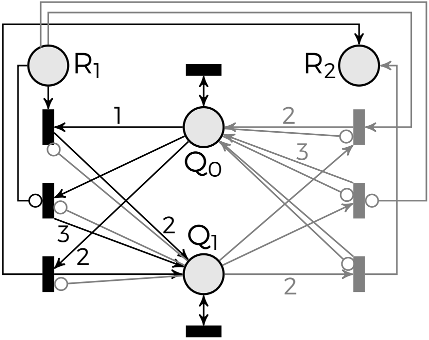
(40 000 variables)

nondeterminism

Simulate U_3 and U_2

Minimising the Number of Places





$\#\text{places} = \#\text{registers} + 2$

Max degree $=$ f(state coding)

$\text{cost}(q_i \to q_j) = \text{code}(i) + \text{code}(j)$

minimise worst cost

subject to $\text{cost}(q_i \to q_j) <$ worst cost

one code per state, one state per code

(40 000 variables)

nondeterminism

Simulate U_3 and U_2

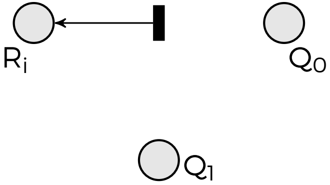
- strongly universal ($p : 5, t : 590, i : 734, d : 208$)
- weakly universal ($p : 4, t : 452, i : 562, d : 162$)

Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

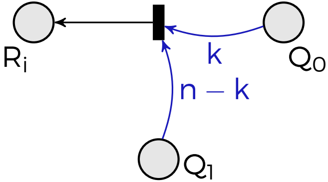
Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)



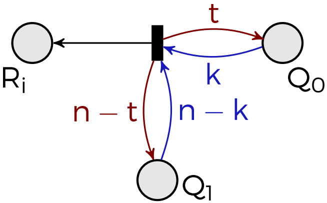
Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)



Deterministic Petri Nets with Few Places

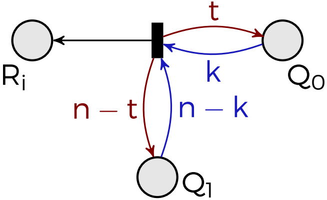
(q_k, RiP, q_t)



Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

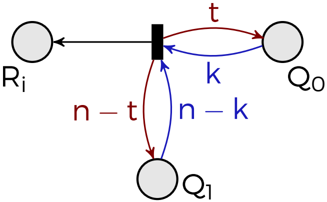


Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

Deterministic evolution



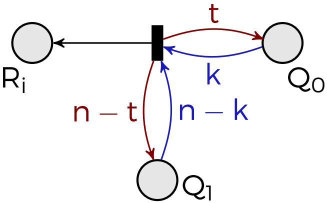
Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

Deterministic evolution

Simulate U_3 and U_2

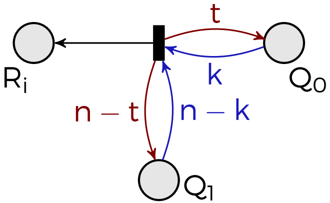


Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

Deterministic evolution



Simulate U_3 and U_2

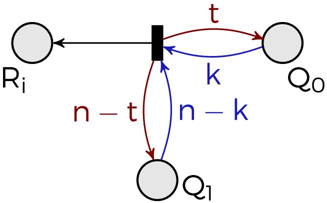
- strongly universal ($p: 5, t: 293, i: 146, d: 314$)
- weakly universal ($p: 4, t: 224, i: 112, d: 242$)

Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

Deterministic evolution



Simulate U_3 and U_2

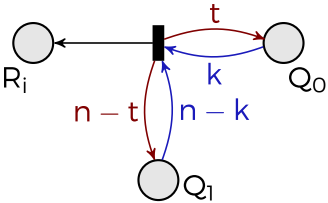
- strongly universal $(p:5, t:293, i:146, d:314)$
  - nondeterministic: $(p:5, t:590, i:734, d:208)$
- weakly universal $(p:4, t:224, i:112, d:242)$
  - nondeterministic: $(p:4, t:452, i:562, d:162)$

Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

Deterministic evolution



Simulate U_3 and U_2

- strongly universal $(p : 5, t : 293, i : 146, d : 314)$
  - nondeterministic: $(p : 5, t : 590, i : 734, d : 208)$
- weakly universal $(p : 4, t : 224, i : 112, d : 242)$
  - nondeterministic: $(p : 4, t : 452, i : 562, d : 162)$

Deterministic vs. Nondeterministic

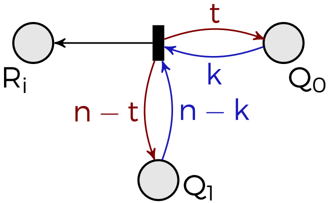
- fewer transitions and inhibitor arcs

Deterministic Petri Nets with Few Places

(q_k, RiP, q_t)

$(k, n-k) \not\leq (t, n-t)$

Deterministic evolution



Simulate U_3 and U_2

- strongly universal $(p:5, t:293, i:146, d:314)$
  - nondeterministic: $(p:5, t:590, i:734, d:208)$
- weakly universal $(p:4, t:224, i:112, d:242)$
  - nondeterministic: $(p:4, t:452, i:562, d:162)$

Deterministic vs. Nondeterministic

- fewer transitions and inhibitor arcs
- bigger transition degree

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes?

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes? Yes!

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes? Yes!

Constructed universal register machines U_3 and U_2

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes? Yes!

Constructed universal register machines U_3 and U_2

- reduce the number of instructions of U_3 and U_2?

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes? Yes!

Constructed universal register machines U_3 and U_2

- reduce the number of instructions of U_3 and U_2?

Constructed generalised register machines U_7 and U_7'

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes? Yes!

Constructed universal register machines U_3 and U_2

- reduce the number of instructions of U_3 and U_2?

Constructed generalised register machines U_7 and U'_7

Constructed a series of small universal Petri nets

Conclusions and Open Questions

$(1, m, 0; 1, q, 0)$ generate complex languages

- computational completeness?

Introduced derivation graphs

- wave normal form?
- further applications?

Control mechanisms increase the computational power

- $(1, 2, 0; 1, 1, 0)$ and $(1, 1, 0; 1, 2, 0)$: universality with graph control with 2 nodes? Yes!

Constructed universal register machines U_3 and U_2

- reduce the number of instructions of U_3 and U_2?

Constructed generalised register machines U_7 and U_7'

Constructed a series of small universal Petri nets

- fewer transitions?

Thank You for Your Attention!

- One-sided insertion-deletion systems
- $(1,1,0;1,2,0) \sim (1,2,0;1,1,0) \sim (1,m,0;1,q,0)$, $m \cdot q \neq 0$, $m + q > 2$
- Derivation graphs
- Computational completeness with control
- graph control, 3 states $(1,2,0;1,1,0)$, $(1,1,0;1,2,0)$
- semi-conditional $(1,0,0;1,0,0)$
- random context $(2,0,0;1,1,0)$
- Universal NEPs with 4, 5, and 7 rules
- Universal register machines with 3 and 2 registers
- Universal generalised register machines with 7 states
- Small Universal Petri Nets

	Strong universality								Weak universality							
Places	30	14	11	21	525	304	5	5	27	13	10	19	397	232	4	4
Transitions	34	23	23	25	659	438	590	293	31	21	21	23	504	339	452	224
Inhibitor arcs	12	30	37	12	3	3	734	146	11	23	30	11	2	2	562	112
Max degree	3	6	10	5	3	22	208	314	3	6	10	5	3	20	162	242