Modulation-Reaction Networks

Leo Lobski[1] and Yoàv Montacute[2]

[1]University College London, UK
[2]National Institute of Informatics, Japan

24th International Conference on Computational Methods in Systems Biology
Lisbon, 24 July 2026

Outline

Motivation: information flow vs. matter flow

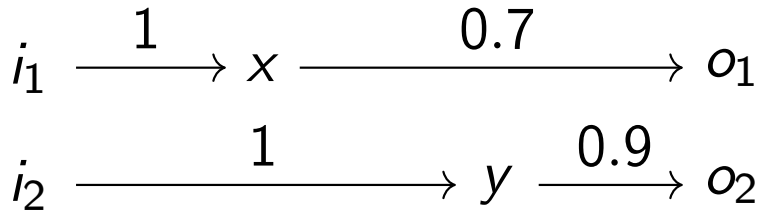
- Reaction networks model the flow of *matter* or *entities*:

$$i_1 \xrightarrow{1} x \xrightarrow{0.7} o_1$$

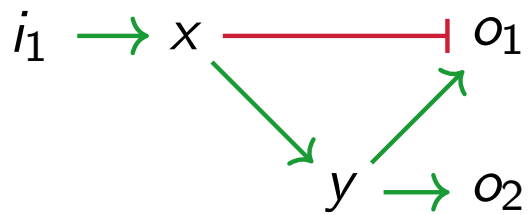
$$i_2 \xrightarrow{1} y \xrightarrow{0.9} o_2$$

Motivation: information flow vs. matter flow

- Reaction networks model the flow of *matter* or *entities*:



- Boolean networks model the flow of *information* or *logic*:



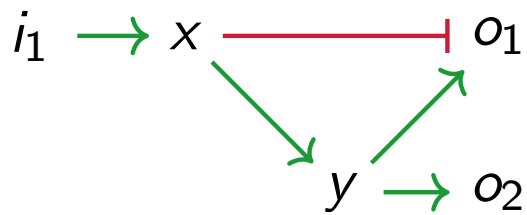
Motivation: information flow vs. matter flow

- Reaction networks model the flow of *matter* or *entities*:

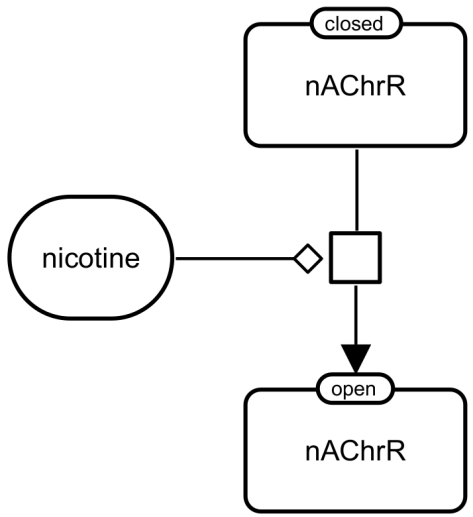
$$i_1 \xrightarrow{1} x \xrightarrow{0.7} o_1$$

$$i_2 \xrightarrow{1} y \xrightarrow{0.9} o_2$$

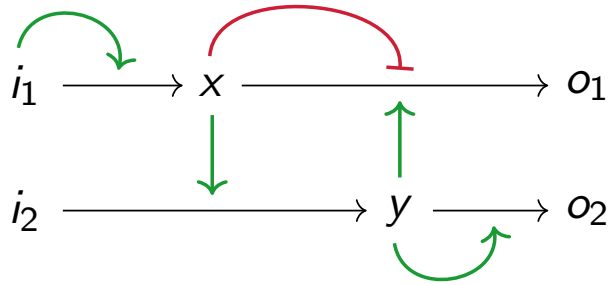
- Boolean networks model the flow of *information* or *logic*:



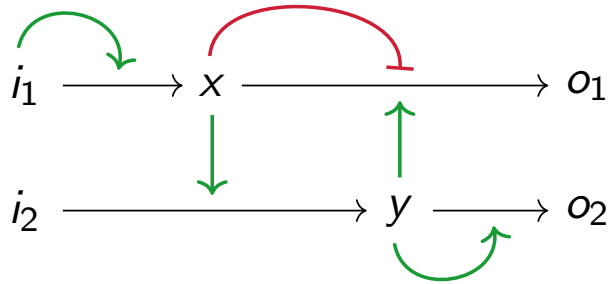
- Biochemical processes involve both:



MR-networks



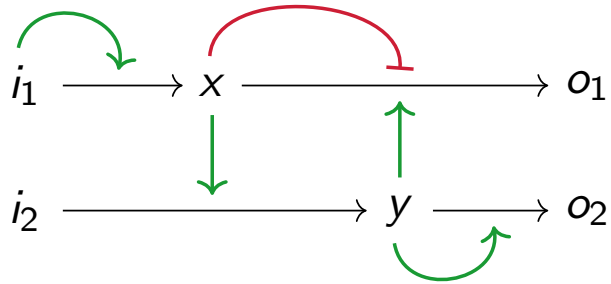
MR-networks



Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

MR-networks

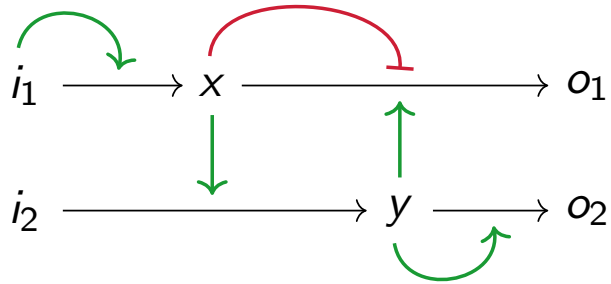


Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

- a finite set V of *vertices*,

MR-networks

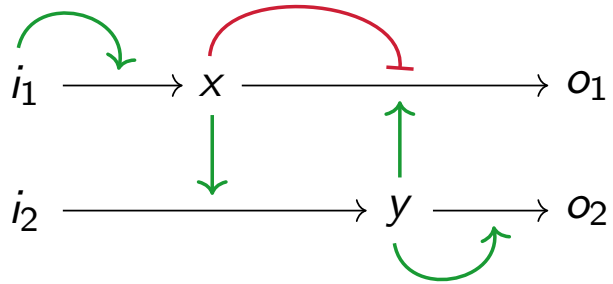


Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

- a finite set V of *vertices*,
- a binary relation $R \subseteq V \times V$ of *reactions*,

MR-networks

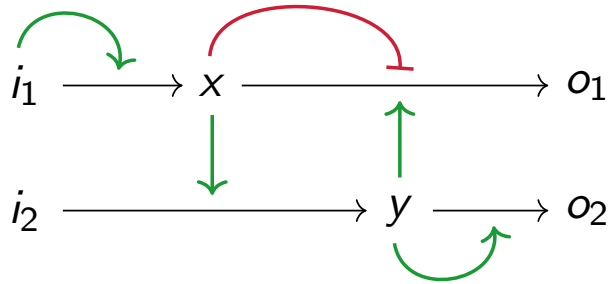


Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

- a finite set V of *vertices*,
- a binary relation $R \subseteq V \times V$ of *reactions*,
- a binary relation $M^{\rightarrow} \subseteq V \times R$ of *activations*,

MR-networks

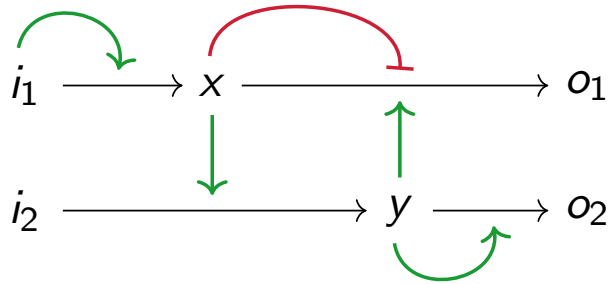


Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

- a finite set V of *vertices*,
- a binary relation $R \subseteq V \times V$ of *reactions*,
- a binary relation $M^{\rightarrow} \subseteq V \times R$ of *activations*,
- a binary relation $M^{\dashv} \subseteq V \times R$ of *inhibitions*.

MR-networks



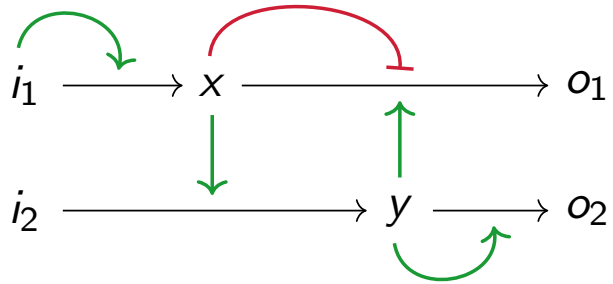
Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

- a finite set V of *vertices*,
- a binary relation $R \subseteq V \times V$ of *reactions*,
- a binary relation $M^{\rightarrow} \subseteq V \times R$ of *activations*,
- a binary relation $M^{\dashv} \subseteq V \times R$ of *inhibitions*.

The pairs in $M^{\rightarrow}$ and $M^{\dashv}$ are referred to as *modulations*.

MR-networks



Definition

An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

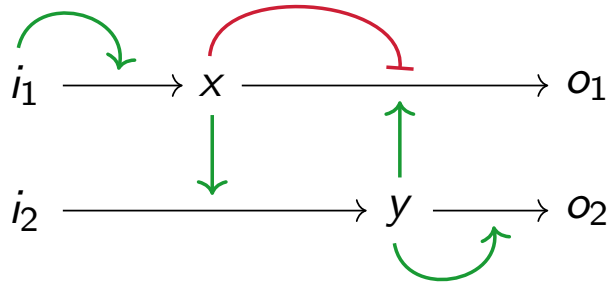
- a finite set V of *vertices*,
- a binary relation $R \subseteq V \times V$ of *reactions*,
- a binary relation $M^{\rightarrow} \subseteq V \times R$ of *activations*,
- a binary relation $M^{\dashv} \subseteq V \times R$ of *inhibitions*.

The pairs in $M^{\rightarrow}$ and $M^{\dashv}$ are referred to as *modulations*.

Example

Consider the MR-network with $V = \{x, y, z\}$, $R = \{(x, y)\}$, $M^{\rightarrow} = \{(z, (x, y))\}$ and $M^{\dashv} = \varnothing$:

MR-networks



Definition

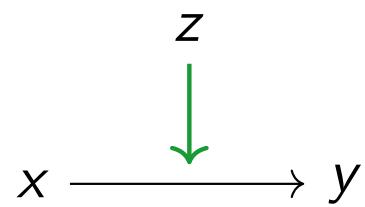
An *MR-network* $(V, R, M^{\rightarrow}, M^{\dashv})$ consists of:

- a finite set V of *vertices*,
- a binary relation $R \subseteq V \times V$ of *reactions*,
- a binary relation $M^{\rightarrow} \subseteq V \times R$ of *activations*,
- a binary relation $M^{\dashv} \subseteq V \times R$ of *inhibitions*.

The pairs in $M^{\rightarrow}$ and $M^{\dashv}$ are referred to as *modulations*.

Example

Consider the MR-network with $V = \{x, y, z\}$, $R = \{(x, y)\}$, $M^{\rightarrow} = \{(z, (x, y))\}$ and $M^{\dashv} = \varnothing$:



Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$.

Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.

Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.

The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.

The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

$$g(u, v) := f(u) \wedge \bigvee_{w \in V : M^{\rightarrow}(w,(u,v))} f(w) \wedge \bigwedge_{w \in V : M^{\dashv}(w,(u,v))} \neg f(w)$$

Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.

The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

$$g(u, v) := f(u) \wedge \bigvee_{w \in V : M^{\rightarrow}(w,(u,v))} f(w) \wedge \bigwedge_{w \in V : M^{\dashv}(w,(u,v))} \neg f(w)$$

$$f'(u) := f(u) \vee \bigvee_{w \in V : (w,u) \in R} g(w, u)$$

Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.
The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

$$g(u, v) := f(u) \wedge \bigvee_{w \in V : M^{\rightarrow}(w,(u,v))} f(w) \wedge \bigwedge_{w \in V : M^{\dashv}(w,(u,v))} \neg f(w)$$

$$f'(u) := f(u) \vee \bigvee_{w \in V : (w,u) \in R} g(w, u)$$

Example

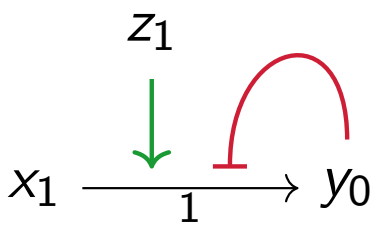
Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.
The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

$$g(u, v) := f(u) \wedge \bigvee_{w \in V : M^{\rightarrow}(w,(u,v))} f(w) \wedge \bigwedge_{w \in V : M^{\dashv}(w,(u,v))} \neg f(w)$$

$$f'(u) := f(u) \vee \bigvee_{w \in V : (w,u) \in R} g(w, u)$$

Example



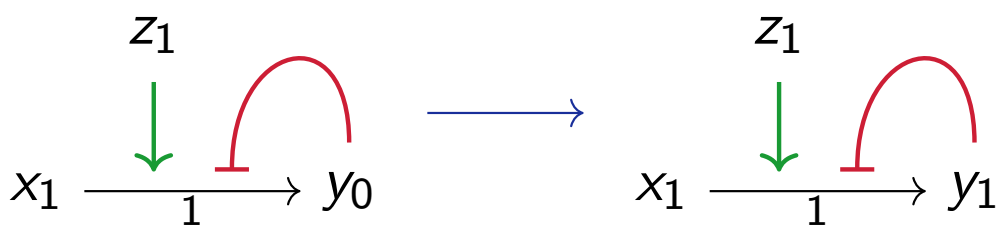
Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.
The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

$$g(u, v) := f(u) \wedge \bigvee_{w \in V : M^{\rightarrow}(w,(u,v))} f(w) \wedge \bigwedge_{w \in V : M^{\dashv}(w,(u,v))} \neg f(w)$$

$$f'(u) := f(u) \vee \bigvee_{w \in V : (w,u) \in R} g(w, u)$$

Example



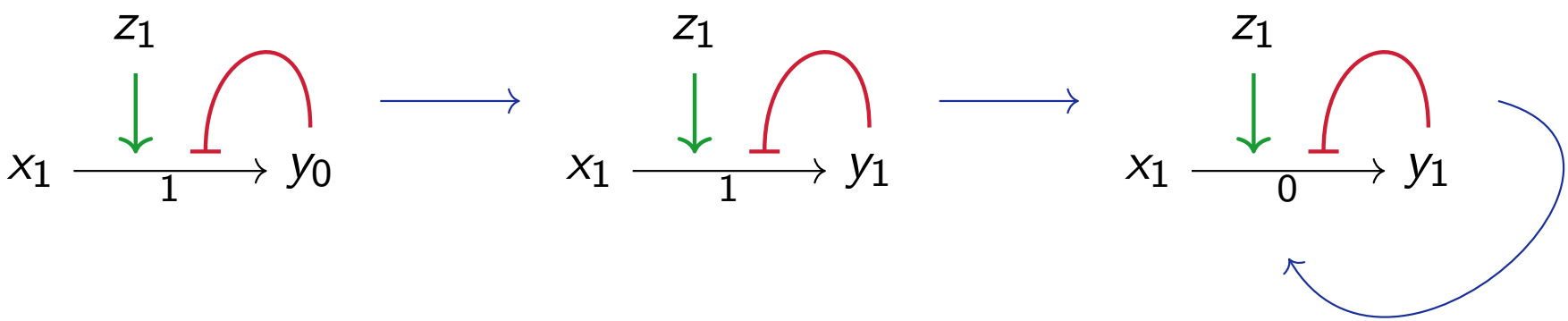
Synchronous update

A *valuation* is a function $f : V \to \mathbb{Z}_2$. A *state* is a pair (f, g) of a valuation f and a function $g : R \to \mathbb{Z}_2$.
The *synchronous* update rule $f \mapsto (f, g) \mapsto f'$ is given by:

$$g(u, v) := f(u) \wedge \bigvee_{w \in V : M^{\rightarrow}(w,(u,v))} f(w) \wedge \bigwedge_{w \in V : M^{\dashv}(w,(u,v))} \neg f(w)$$

$$f'(u) := f(u) \vee \bigvee_{w \in V : (w,u) \in R} g(w, u)$$

Example



Asynchronous update

- A reaction can go from inactive to active if its source is active, at least one of its activators is active, and none of its inhibitors are active

Asynchronous update

- A reaction can go from inactive to active if its source is active, at least one of its activators is active, and none of its inhibitors are active
- A reaction can go from active to inactive if the activation condition above is no longer satisfied, or if its target is active

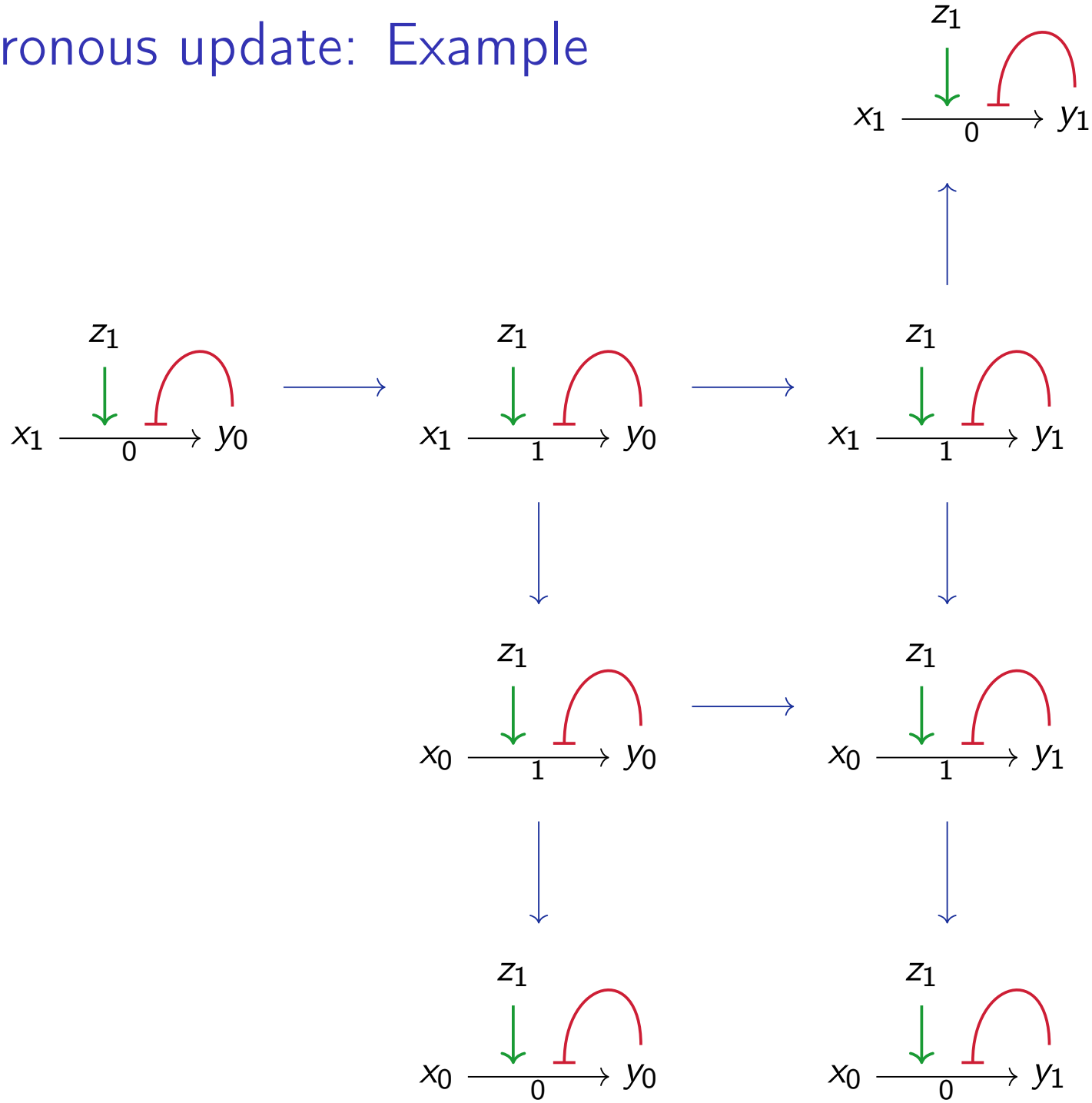
Asynchronous update

- A reaction can go from inactive to active if its source is active, at least one of its activators is active, and none of its inhibitors are active
- A reaction can go from active to inactive if the activation condition above is no longer satisfied, or if its target is active
- A vertex can go from inactive to active if there is at least one active incoming reaction

Asynchronous update

- A reaction can go from inactive to active if its source is active, at least one of its activators is active, and none of its inhibitors are active
- A reaction can go from active to inactive if the activation condition above is no longer satisfied, or if its target is active
- A vertex can go from inactive to active if there is at least one active incoming reaction
- A vertex can go from active to inactive if there is an active outgoing reaction

Asynchronous update: Example



The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{+}\varphi \mid \Diamond^{-}\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{+}\varphi \mid \Diamond^{-}\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

- In $\mu p.\varphi$, require that p occurs positively in φ

The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{+}\varphi \mid \Diamond^{-}\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

- In $\mu p.\varphi$, require that p occurs positively in φ
- Define the operations $\vee, \rightarrow$ and $\leftrightarrow$ in the usual way

The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{+}\varphi \mid \Diamond^{-}\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

- In $\mu p.\varphi$, require that p occurs positively in φ
- Define the operations $\vee, \rightarrow$ and $\leftrightarrow$ in the usual way
- Define $\top := 0 \vee 1$ and $\bot := 0 \wedge 1$

The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{+}\varphi \mid \Diamond^{-}\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

- In $\mu p.\varphi$, require that p occurs positively in φ
- Define the operations $\vee, \rightarrow$ and $\leftrightarrow$ in the usual way
- Define $\top := 0 \vee 1$ and $\bot := 0 \wedge 1$
- Define the duals $\Box^{+} := \neg\Diamond^{+}\neg$ and $\Box^{-} := \neg\Diamond^{-}\neg$

The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{+}\varphi \mid \Diamond^{-}\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

- In $\mu p.\varphi$, require that p occurs positively in φ
- Define the operations $\vee, \rightarrow$ and $\leftrightarrow$ in the usual way
- Define $\top := 0 \vee 1$ and $\bot := 0 \wedge 1$
- Define the duals $\Box^{+} := \neg\Diamond^{+}\neg$ and $\Box^{-} := \neg\Diamond^{-}\neg$
- define the dual $\nu p.\varphi := \neg\mu p.\neg\varphi[\neg p/p]$,

The logic MRL: Syntax

$$\varphi ::= 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^+\varphi \mid \Diamond^-\varphi \mid X\varphi \mid \mu p.\varphi \mid i \mid @_i\varphi$$

- In $\mu p.\varphi$, require that p occurs positively in φ
- Define the operations $\vee, \rightarrow$ and $\leftrightarrow$ in the usual way
- Define $\top := 0 \vee 1$ and $\bot := 0 \wedge 1$
- Define the duals $\Box^+ := \neg\Diamond^+\neg$ and $\Box^- := \neg\Diamond^-\neg$
- define the dual $\nu p.\varphi := \neg\mu p.\neg\varphi[\neg p/p]$,
- further define $F\varphi := \mu p.(\varphi \vee Xp)$ and $G\varphi := \neg F\neg\varphi$

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ
- $\Diamond^-\varphi$ holds at a vertex if some predecessor along an active reaction satisfies φ

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ
- $\Diamond^-\varphi$ holds at a vertex if some predecessor along an active reaction satisfies φ

- $X\varphi$ holds if φ holds at the next step

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ
- $\Diamond^-\varphi$ holds at a vertex if some predecessor along an active reaction satisfies φ

- $X\varphi$ holds if φ holds at the next step
- $F\varphi$ and $G\varphi$ hold if φ holds at some (resp. every) future step

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ
- $\Diamond^-\varphi$ holds at a vertex if some predecessor along an active reaction satisfies φ

- $X\varphi$ holds if φ holds at the next step
- $F\varphi$ and $G\varphi$ hold if φ holds at some (resp. every) future step

- $\mu p.\varphi$ denotes the least property p satisfying φ

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ
- $\Diamond^-\varphi$ holds at a vertex if some predecessor along an active reaction satisfies φ

- $X\varphi$ holds if φ holds at the next step
- $F\varphi$ and $G\varphi$ hold if φ holds at some (resp. every) future step

- $\mu p.\varphi$ denotes the least property p satisfying φ

- A nominal i holds at x iff x is the vertex named i

The logic MRL: Semantics

- 0 and 1 test the current activation value of a vertex
- The connectives $\neg$ and $\wedge$ are Boolean

- $\Diamond^+\varphi$ holds at a vertex if some successor along an active reaction satisfies φ
- $\Diamond^-\varphi$ holds at a vertex if some predecessor along an active reaction satisfies φ

- $X\varphi$ holds if φ holds at the next step
- $F\varphi$ and $G\varphi$ hold if φ holds at some (resp. every) future step

- $\mu p.\varphi$ denotes the least property p satisfying φ

- A nominal i holds at x iff x is the vertex named i
- $@_i\varphi$ asserts that φ holds at the vertex named i

The logic MRL: Semantics

Network modalities

$\mathcal{M}_i, x \vDash_\sigma j \iff f_i(x) = j, for j \in \mathbb{Z}_2$

$\mathcal{M}_i, x \vDash_\sigma p \iff (x, i) \in \sigma(p)$

$\mathcal{M}_i, x \vDash_\sigma \neg\varphi \iff \mathcal{M}_i, x \nvDash_\sigma \varphi$

$\mathcal{M}_i, x \vDash_\sigma \varphi \wedge \psi \iff \mathcal{M}_i, x \vDash_\sigma \varphi$ and $\mathcal{M}_i, x \vDash_\sigma \psi$

$\mathcal{M}_i, x \vDash_\sigma \Diamond^+\varphi \iff \exists y \in V$ s.t. $xRy, g_i(x, y) = 1$ and $\mathcal{M}_i, y \vDash_\sigma \varphi$

$\mathcal{M}_i, x \vDash_\sigma \Diamond^-\varphi \iff \exists y \in V$ s.t. $yRx, g_i(y, x) = 1$ and $\mathcal{M}_i, y \vDash_\sigma \varphi$

Computation modalities

$\mathcal{M}_i, x \vDash_\sigma X\varphi \iff \mathcal{M}_{i+1}, x \vDash_\sigma \varphi$

$\mathcal{M}_i, x \vDash_\sigma F\varphi \iff \mathcal{M}_\ell, x \vDash_\sigma \varphi$ for some $\ell \geq i$

Fixed-point modality

$\mathcal{M}_i, x \vDash_\sigma \mu p.\varphi \iff (x, i) \in \bigcap \left\{ S \subseteq V \times \mathbb{N} : \Phi_\sigma^{\varphi,p}(S) \subseteq S \right\},$

where $\Phi_\sigma^{\varphi,p}(S) = \{(x', i') \in V \times \mathbb{N} : \mathcal{M}_{i'}, x' \vDash_{\sigma[p \mapsto S]} \varphi\}$

Hybrid modalities

$\mathcal{M}_i, x \vDash_\sigma y \iff x = y$

$\mathcal{M}_i, x \vDash_\sigma @_y\varphi \iff \mathcal{M}_i, y \vDash_\sigma \varphi$

Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$

Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathsf{Sust}(y) := @_y\, FG\Diamond^{-}1$

Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathsf{Sust}(y) := @_y\, FG\Diamond^{-}1$
- $\mathsf{SustProd}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} \mathsf{Sust}(y)\right)$

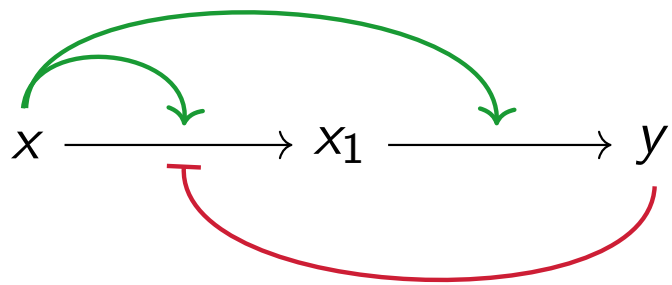
Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathsf{Sust}(y) := @_y\, FG\Diamond^- 1$
- $\mathsf{SustProd}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} \mathsf{Sust}(y)\right)$
- $\mathsf{Reach}(x, y) := @_x\, \mu p.\, (y \vee F\Diamond^+ p)$

Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathsf{Sust}(y) := @_y\, FG\Diamond^- 1$
- $\mathsf{SustProd}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} \mathsf{Sust}(y)\right)$
- $\mathsf{Reach}(x, y) := @_x\, \mu p.\, (y \vee F\Diamond^+ p)$

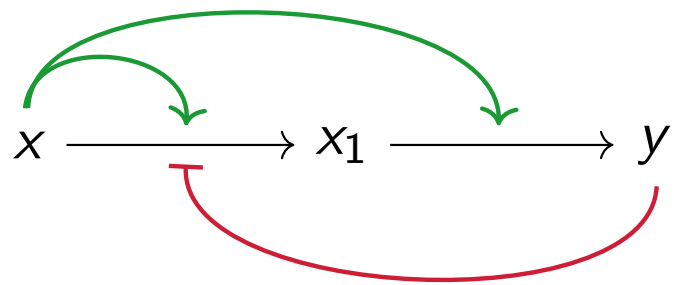
Example



Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathsf{Sust}(y) := @_y\, FG\Diamond^- 1$
- $\mathsf{SustProd}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} \mathsf{Sust}(y)\right)$
- $\mathsf{Reach}(x, y) := @_x\, \mu p.\, (y \vee F\Diamond^+ p)$

Example

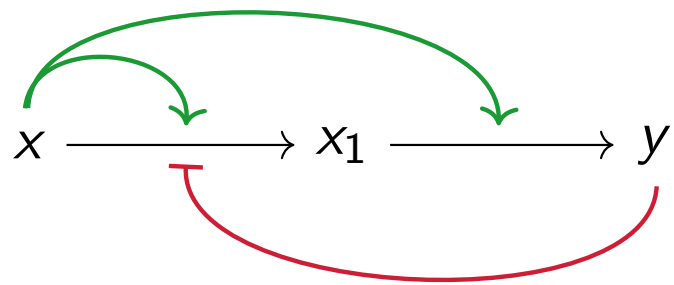


- $\mathcal{M} \vDash \mathsf{Prod}(x, y)$ and $\mathrm{M} \nvDash \mathsf{Prod}(x, x_1)$

Expressivity: Producibility and reachability

- $\mathsf{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathsf{Sust}(y) := @_y\, FG\Diamond^- 1$
- $\mathsf{SustProd}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} \mathsf{Sust}(y)\right)$
- $\mathsf{Reach}(x, y) := @_x\, \mu p.\, (y \vee F\Diamond^+ p)$

Example

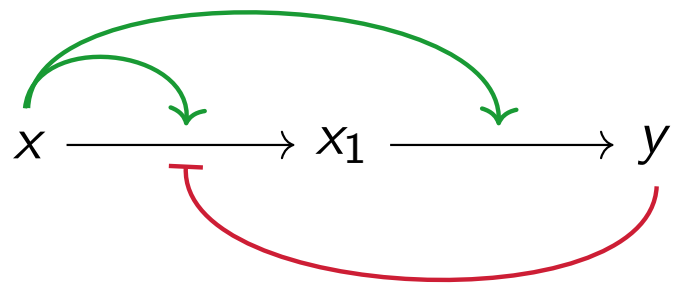


- $\mathcal{M} \vDash \mathsf{Prod}(x, y)$ and $\mathrm{M} \nvDash \mathsf{Prod}(x, x_1)$
- $\mathrm{M} \nvDash \mathsf{SustProd}(x, y)$ and $\mathcal{M} \vDash \mathsf{SustProd}(\{x, x_1\}, y)$

Expressivity: Producibility and reachability

- $\mathrm{Prod}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} @_y\, F1\right)$
- $\mathrm{Sust}(y) := @_y\, FG\Diamond^- 1$
- $\mathrm{SustProd}(X, Y) := (\bigwedge_{x \in X} @_x\, 1) \to \left(\bigwedge_{y \in Y} \mathrm{Sust}(y)\right)$
- $\mathrm{Reach}(x, y) := @_x\, \mu p.\, (y \vee F\Diamond^+ p)$

Example



- $\mathcal{M} \vDash \mathrm{Prod}(x, y)$ and $\mathrm{M} \nvDash \mathrm{Prod}(x, x_1)$
- $\mathrm{M} \nvDash \mathrm{SustProd}(x, y)$ and $\mathcal{M} \vDash \mathrm{SustProd}(\{x, x_1\}, y)$
- $\mathrm{Reach}(x, y)$ is satisfied if and only if x is initially present and y is initially absent

Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a \, G \, \Box^+ \, (\bigvee_{a' \in A} a')$

Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$

Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$
- $\mathsf{PreAttr}(A, B) := \mathsf{Sink}(A) \wedge \mathsf{Flow}(B, A)$

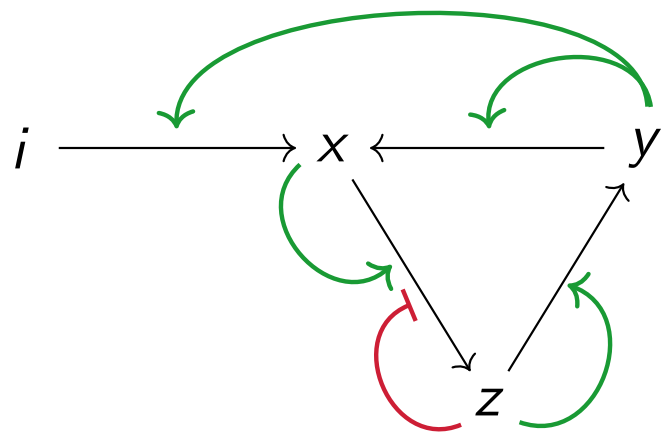
Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$
- $\mathsf{PreAttr}(A, B) := \mathsf{Sink}(A) \wedge \mathsf{Flow}(B, A)$
- $\mathsf{Attr}(A, B) := \mathsf{PreAttr}(A, B) \bigwedge_{A' \not\subseteq A} \neg \mathsf{PreAttr}(A', B)$

Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$
- $\mathsf{PreAttr}(A, B) := \mathsf{Sink}(A) \wedge \mathsf{Flow}(B, A)$
- $\mathsf{Attr}(A, B) := \mathsf{PreAttr}(A, B) \bigwedge_{A' \not\subseteq A} \neg \mathsf{PreAttr}(A', B)$

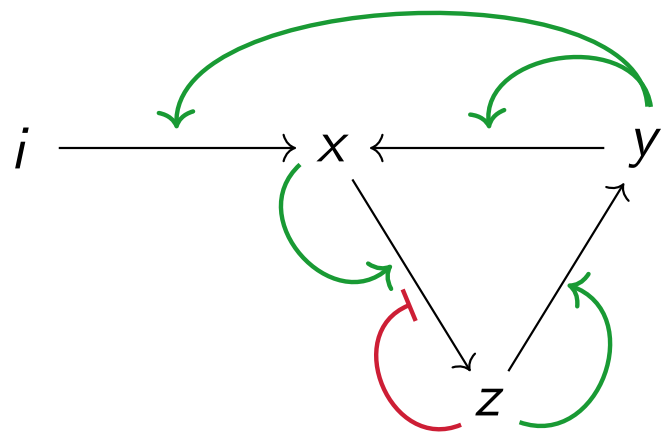
Example



Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$
- $\mathsf{PreAttr}(A, B) := \mathsf{Sink}(A) \wedge \mathsf{Flow}(B, A)$
- $\mathsf{Attr}(A, B) := \mathsf{PreAttr}(A, B) \bigwedge_{A' \not\subseteq A} \neg \mathsf{PreAttr}(A', B)$

Example

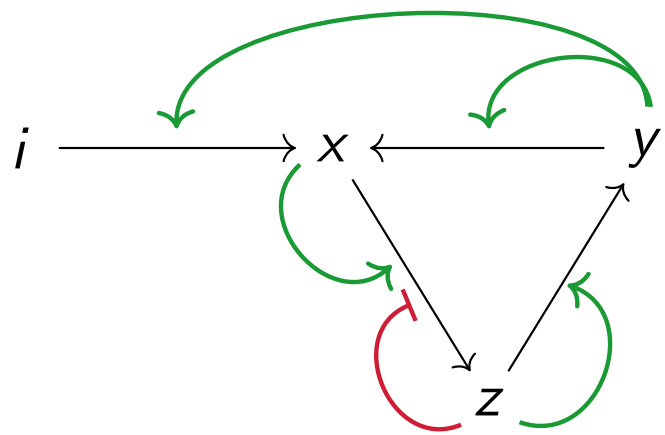


- Let $i, y \mapsto 1$ and 0 otherwise

Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$
- $\mathsf{PreAttr}(A, B) := \mathsf{Sink}(A) \wedge \mathsf{Flow}(B, A)$
- $\mathsf{Attr}(A, B) := \mathsf{PreAttr}(A, B) \bigwedge_{A' \not\subseteq A} \neg \mathsf{PreAttr}(A', B)$

Example

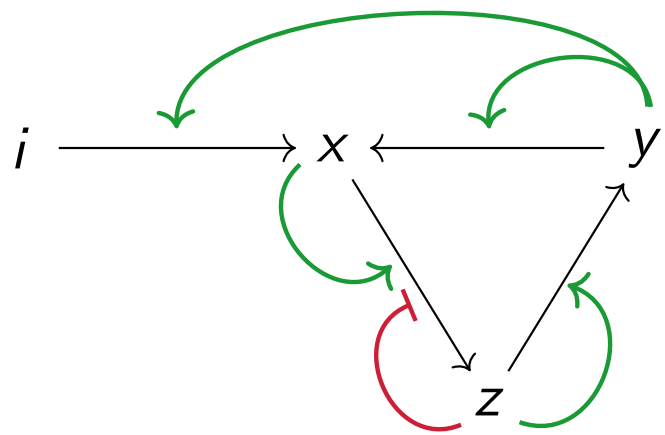


- Let $i, y \mapsto 1$ and 0 otherwise
- $\mathcal{M}_0 \vDash \mathsf{Flow}(V, z)$ and $\mathrm{M}_0 \nvDash \mathsf{Sink}(z)$

Expressivity: Attractors

- $\mathsf{Sink}(A) := \bigwedge_{a \in A} @_a\, G\, \Box^+ \left(\bigvee_{a' \in A} a'\right)$
- $\mathsf{Flow}(B, A) := \bigwedge_{b \in B} @_b\, \mu p.\, \left(\bigvee_{a \in A} a \vee \left(F \Diamond^+ p \wedge \Box^+ p\right)\right)$
- $\mathsf{PreAttr}(A, B) := \mathsf{Sink}(A) \wedge \mathsf{Flow}(B, A)$
- $\mathsf{Attr}(A, B) := \mathsf{PreAttr}(A, B) \bigwedge_{A' \not\subseteq A} \neg \mathsf{PreAttr}(A', B)$

Example



- Let $i, y \mapsto 1$ and 0 otherwise
- $\mathcal{M}_0 \vDash \mathsf{Flow}(V, z)$ and $\mathrm{M}_0 \nvDash \mathsf{Sink}(z)$
- $\mathcal{M}_0 \vDash \mathsf{Attr}(\{x, y, z\}, V)$ and $\mathcal{M}_2 \vDash \mathsf{Attr}(x, V)$

Model-checking complexity & Inexpressivity

Theorem

Checking $\mathcal{M}_i, x \vDash \varphi$ can be done in time $O(|V|^d \cdot |\varphi|^d)$ where d is the alternation depth of φ.

Model-checking complexity & Inexpressivity

Theorem

Checking $\mathcal{M}_i, x \vDash \varphi$ *can be done in time* $O(|V|^d \cdot |\varphi|^d)$ *where* d *is the alternation depth of* φ.

Proposition

There is no MRL *sentence* φ *such that for all MR-models* $(\mathcal{M}, f_0)$ *one has* $\mathcal{M}_i, x \vDash \varphi \iff \left|\{y \in V : yRx, g_i(y, x) = 1\}\right| = 2$.

Ongoing and future work

- Expressivity under asynchronous update rule

Ongoing and future work

- Expressivity under asynchronous update rule
- Stochastic update rule and corresponding logic

Ongoing and future work

- Expressivity under asynchronous update rule
- Stochastic update rule and corresponding logic
- Extending to other modulations of SBGN

Ongoing and future work

- Expressivity under asynchronous update rule
- Stochastic update rule and corresponding logic
- Extending to other modulations of SBGN
- Relation to known models of computation: e.g. Petri nets, asynchronous automata

Ongoing and future work

- Expressivity under asynchronous update rule
- Stochastic update rule and corresponding logic
- Extending to other modulations of SBGN
- Relation to known models of computation: e.g. Petri nets, asynchronous automata
- Composition of MR-networks

Ongoing and future work

- Expressivity under asynchronous update rule
- Stochastic update rule and corresponding logic
- Extending to other modulations of SBGN
- Relation to known models of computation: e.g. Petri nets, asynchronous automata
- Composition of MR-networks

Thank you for your attention!