Identifying System Dynamics with Machine Learning

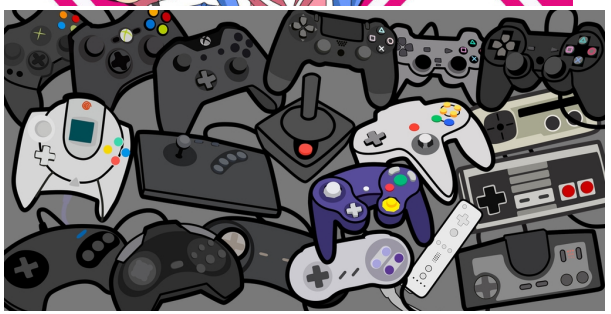
Tony Ribeiro

MeForBio team, LS2N, École Centrale de Nantes, France
Inoue Laboratory, NII, Tokyo

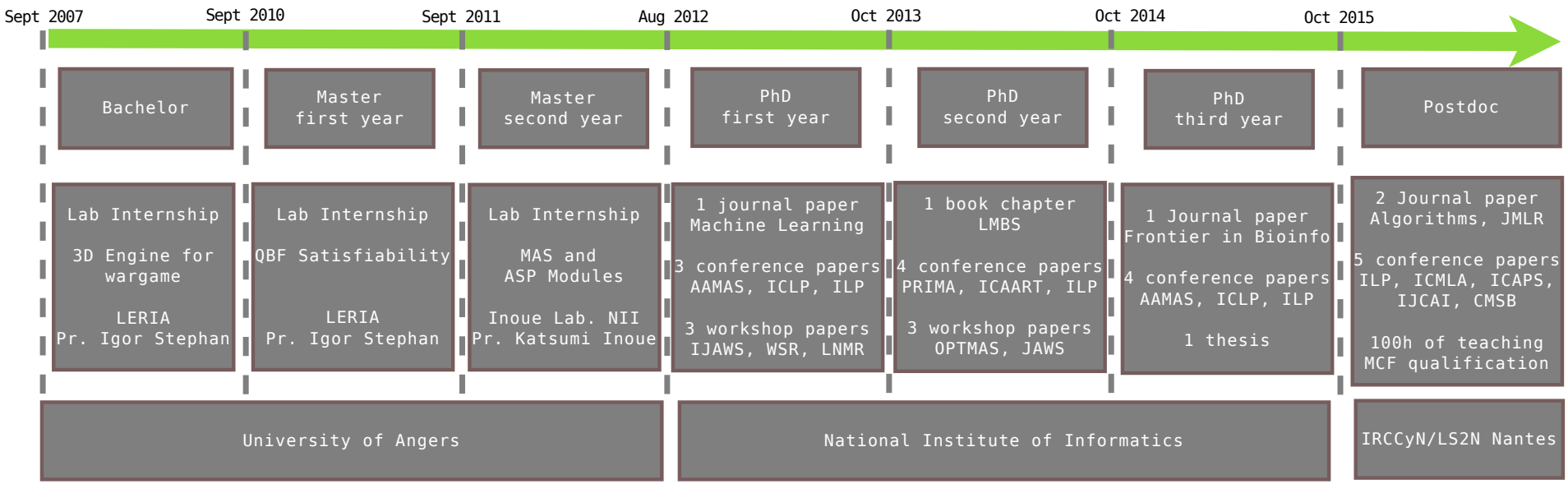
Nantes, April 9th 2018

Outline

About Me



Study



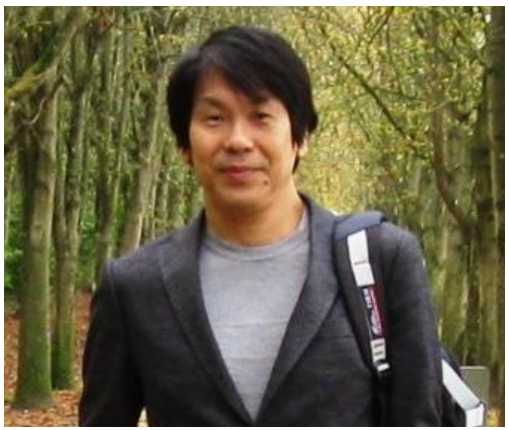
Thesis:
Studies on Learning Dynamics of Systems from State Transitions

Inoue Lab

Morgan Magnin

Tenda Okimoto

Nicolas Schwind



Prof. Katsumi Inoue

Main Research Topics

- Machine Learning
- Multi-agent System
- Multi-objective Optimization
- System Resilience

Maxime Clément

Kotaro Okazaki

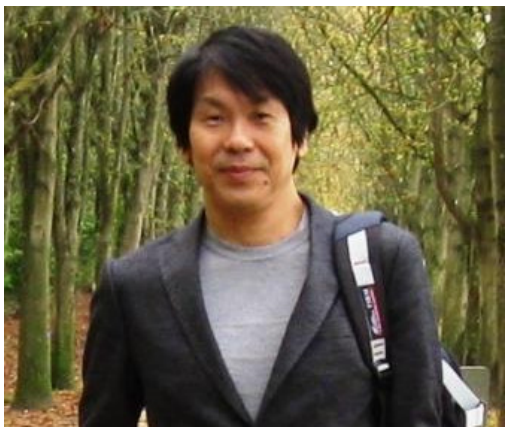
Tony Ribeiro

Inoue Lab

Morgan Magnin

Tenda Okimoto

Nicolas Schwind



Prof. Katsumi Inoue

Main Research Topics

- Machine Learning
- Multi-agent System
- Multi-objective Optimization
- System Resilience

Maxime Clément

Kotaro Okazaki

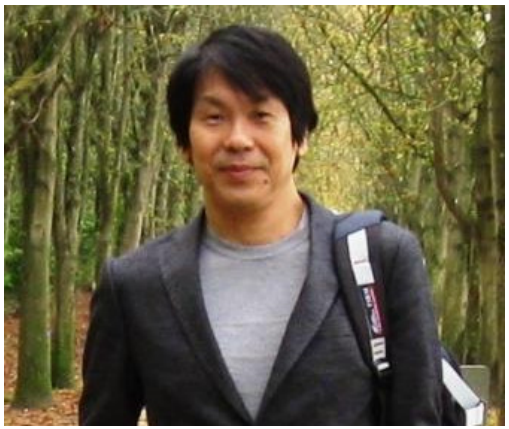
Tony Ribeiro

Inoue Lab

Morgan Magnin

Tenda Okimoto

Nicolas Schwind



Prof. Katsumi Inoue

Main Research Topics

- Machine Learning
- **Multi-agent System**
- Multi-objective Optimization
- System Resilience

Maxime Clément

Kotaro Okazaki

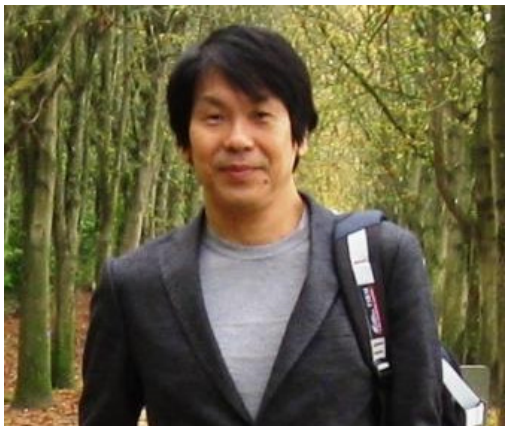
Tony Ribeiro

Inoue Lab

Morgan Magnin

Tenda Okimoto

Nicolas Schwind



Prof. Katsumi Inoue

Main Research Topics

- Machine Learning
- Multi-agent System
- Multi-objective Optimization
- System Resilience

Maxime Clément

Kotaro Okazaki

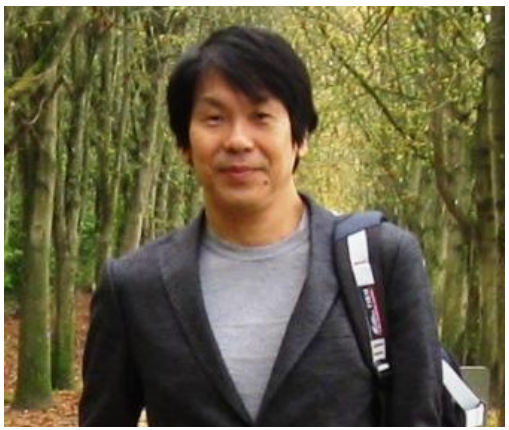
Tony Ribeiro

Inoue Lab

Morgan Magnin

Tenda Okimoto

Nicolas Schwind



Prof. Katsumi Inoue

Main Research Topics

- Machine Learning
- Multi-agent System
- Multi-objective Optimization
- System Resilience

Maxime Clément

Kotaro Okazaki

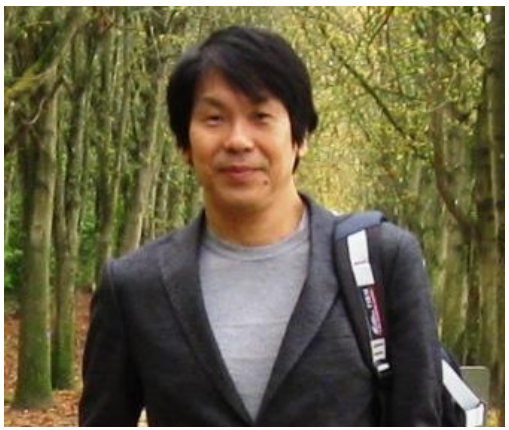
Tony Ribeiro

Inoue Lab

Morgan Magnin

Tenda Okimoto

Nicolas Schwind



Prof. Katsumi Inoue

Main Research Topics

- Machine Learning
- Multi-agent System
- Multi-objective Optimization
- System Resilience

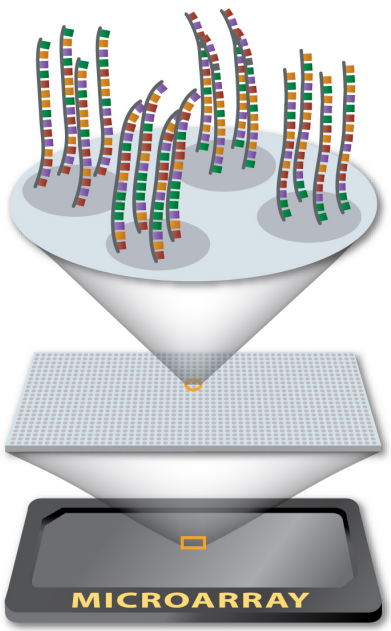
Maxime Clément

Kotaro Okazaki

Tony Ribeiro

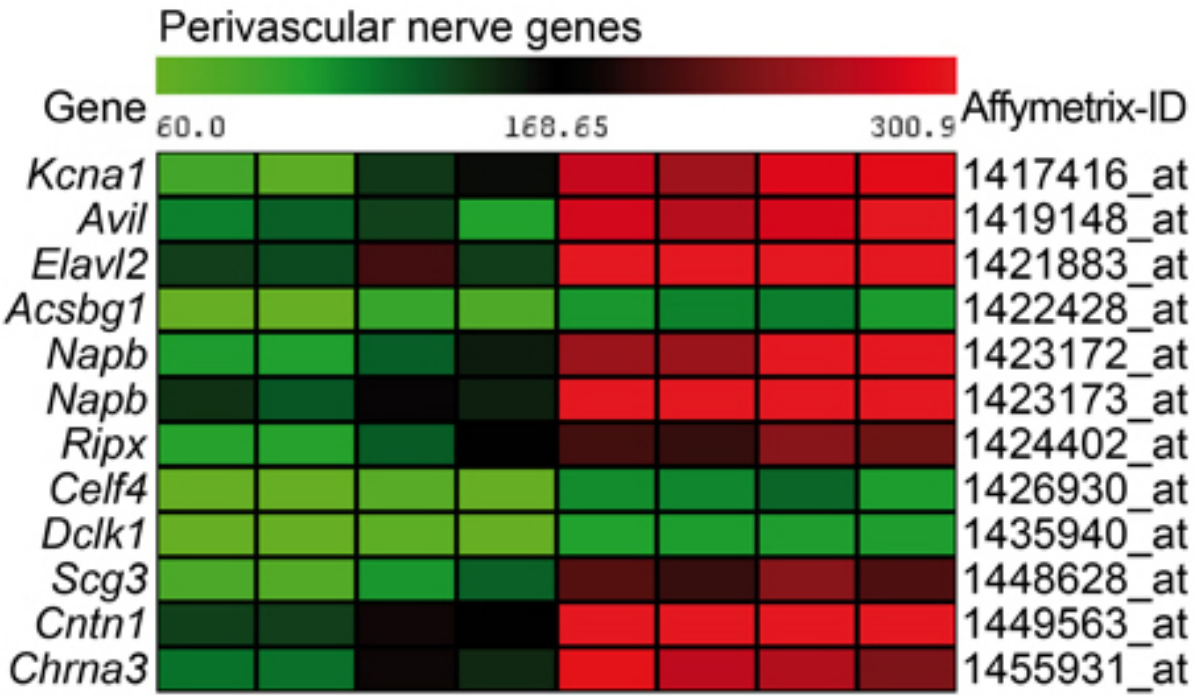
Motivations

With the development of new measurement technologies a large amount of time series data is now produced every day.



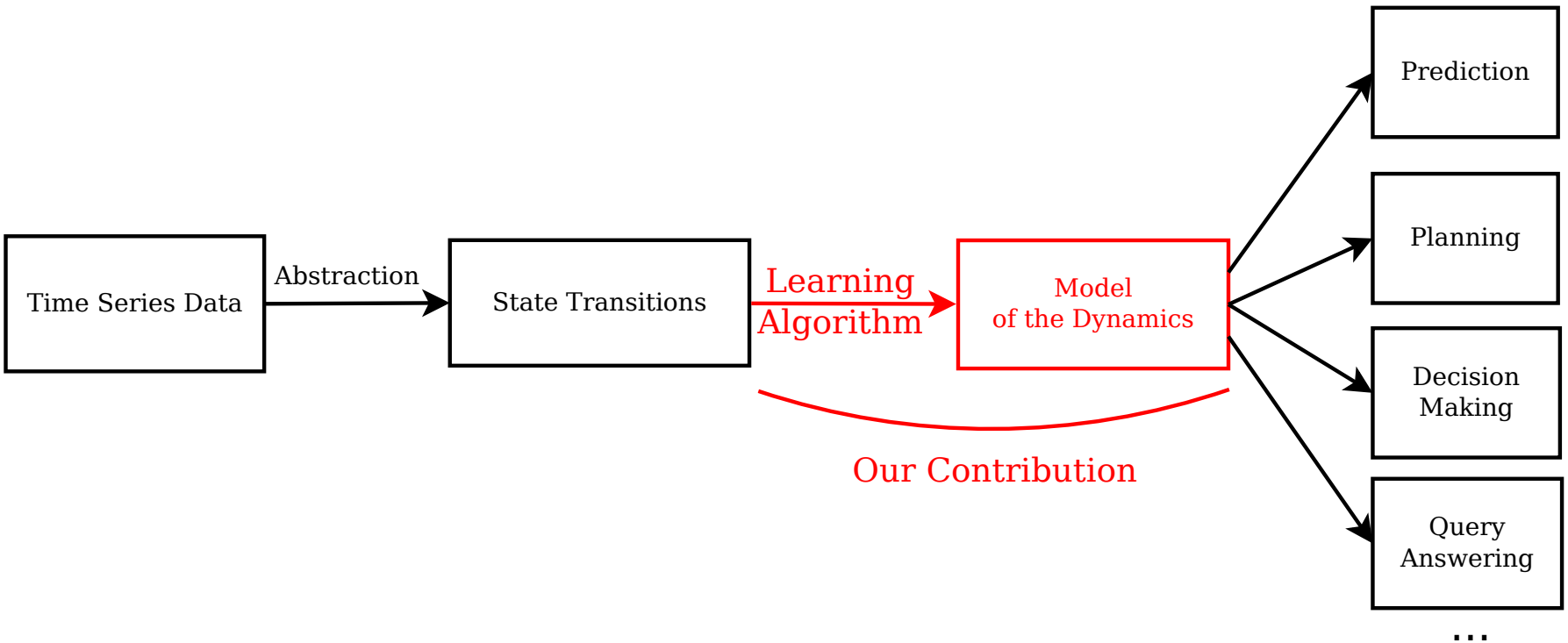
Motivations

But data alone may be not sufficient: we need readability, usability.



Microarray data from Perivascular nerve genes (*Abd Alla et al. 2013*)

Motivations



Problem:

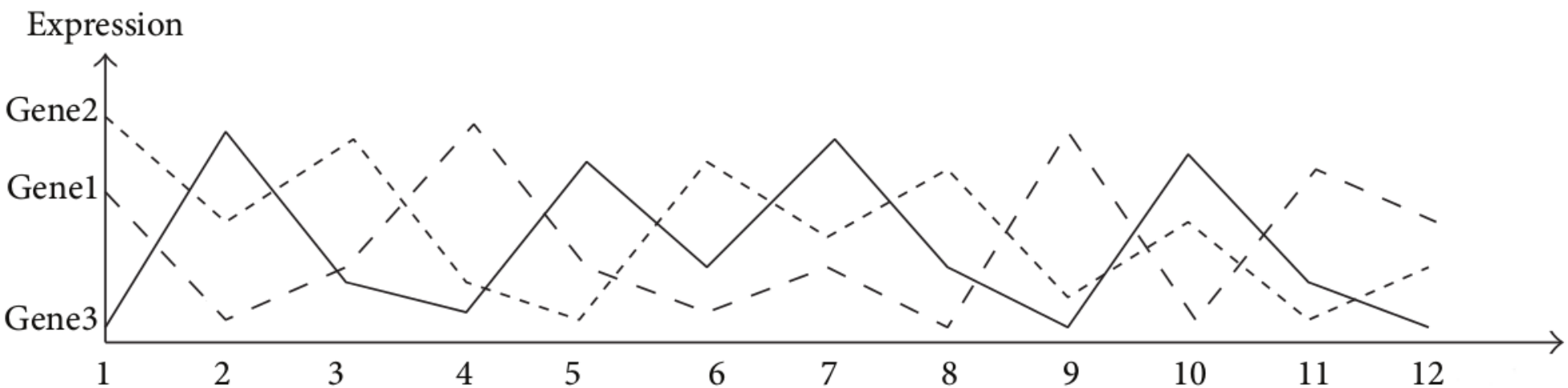
- Time series data alone are not sufficient, we need models.

Goal:

- Automated modeling of systems dynamics from these data.

Abstraction: From Time Series Data to State Transitions

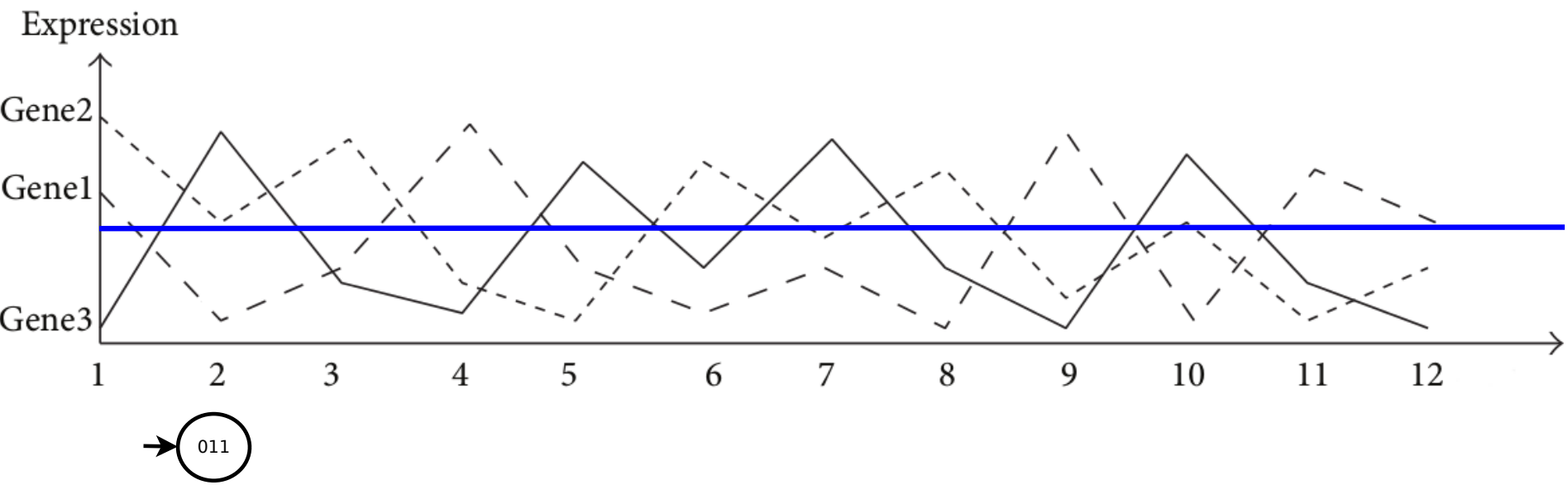
Time series data can be abstracted in many ways. It depends of what we know and what we want to learn about the system.



Time series data of gene expression (*Nakajima and Akutsu 2014*).

Abstraction: From Time Series Data to State Transitions

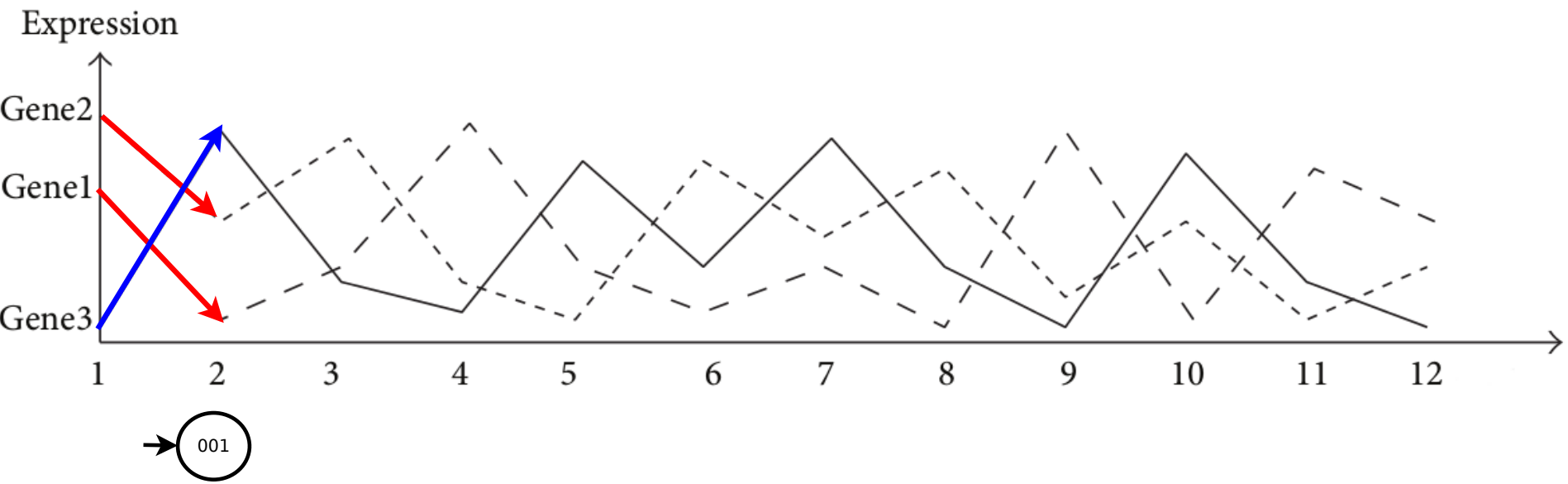
Time series data can be abstracted in many ways. It depends of what we know and what we want to learn about the system.



We can set thresholds to discretize the level of expression of genes.

Abstraction: From Time Series Data to State Transitions

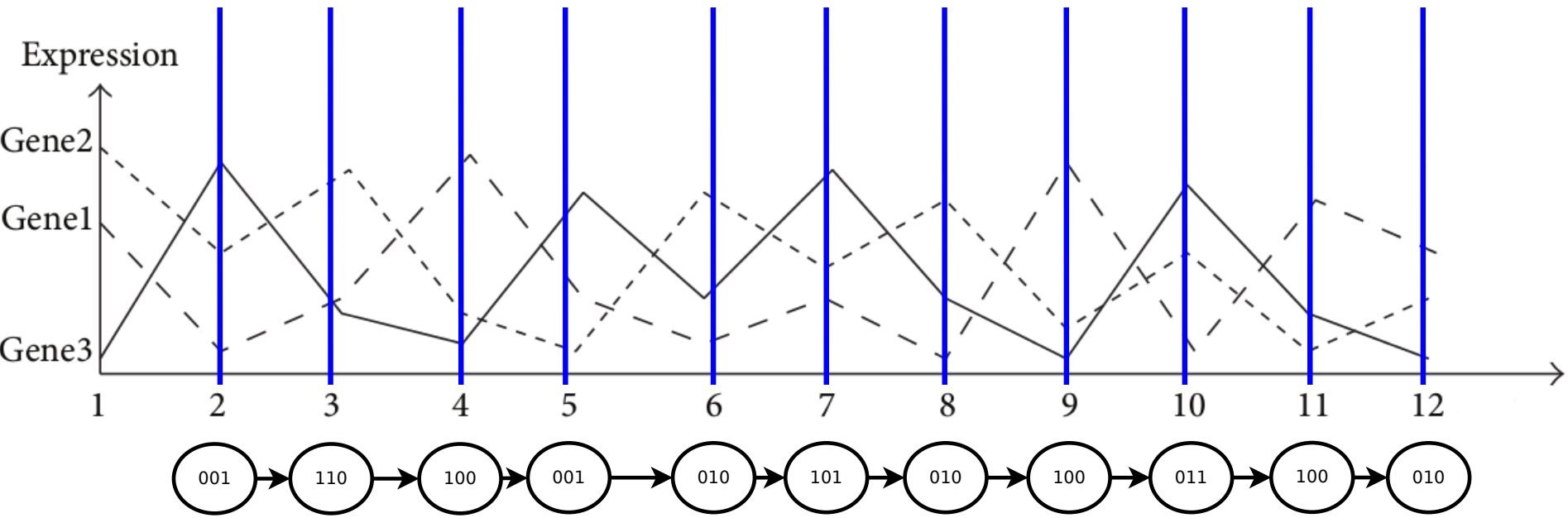
Time series data can be abstracted in many ways. It depends of what we know and what we want to learn about the system.



We can consider the concentration behavior (increase/decrease).

Abstraction: From Time Series Data to State Transitions

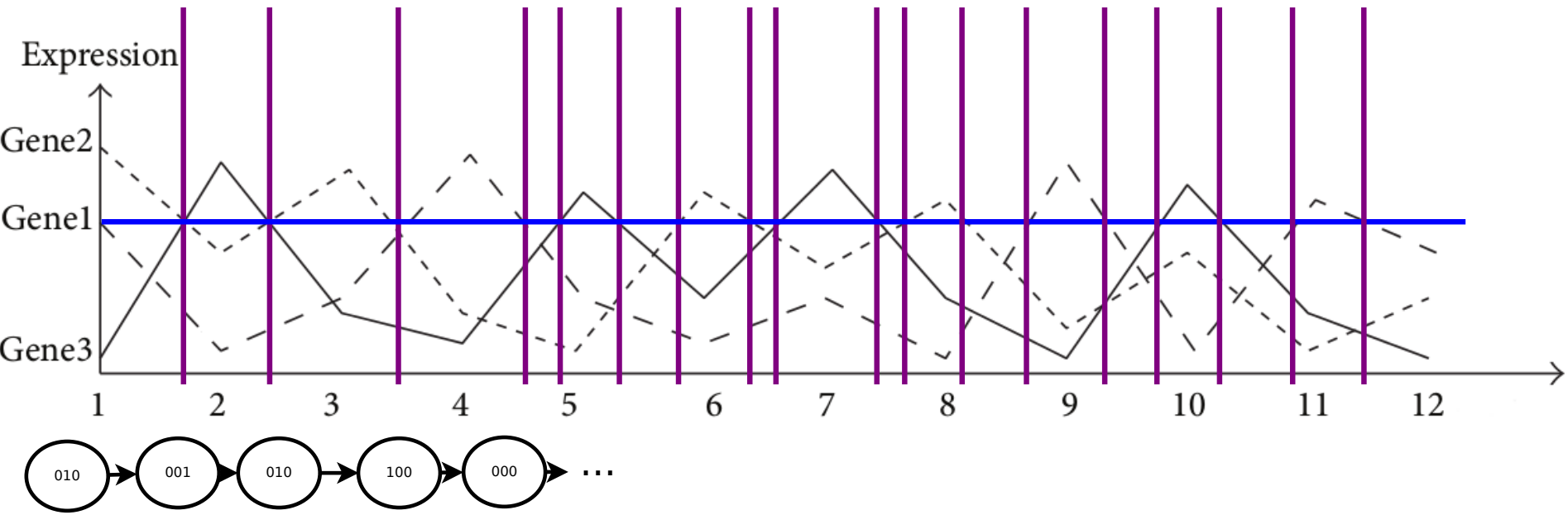
Time series data can be abstracted in many ways. It depends of what we know and what we want to learn about the system.



Transitions can be time-based: a state for each time unit.

Abstraction: From Time Series Data to State Transitions

Time series data can be abstracted in many ways. It depends of what we know and what we want to learn about the system.

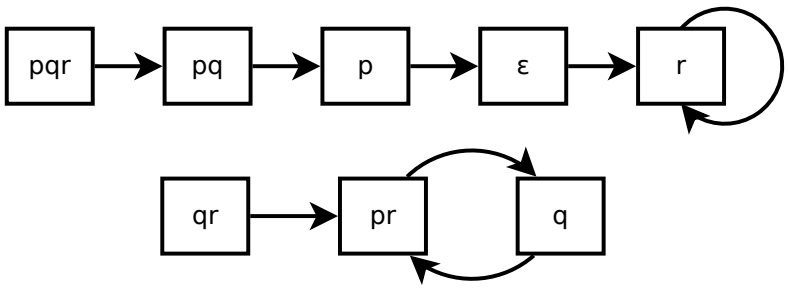


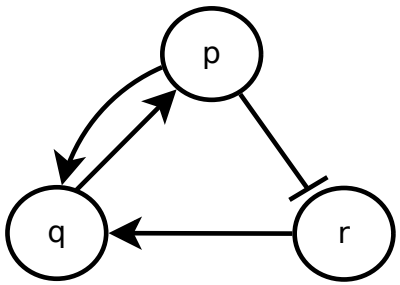
Transitions can be event-based: a state for each change.

Learning From Interpretation Transitions (LFIT)

A framework for learning system dynamics from state transitions.

- Basic Idea:
  - Learn a logic program by observing the behavior of a system.
  - This logic program represents the dynamics of the system.





Input: Behavior of the system

Output: Dynamics of the system

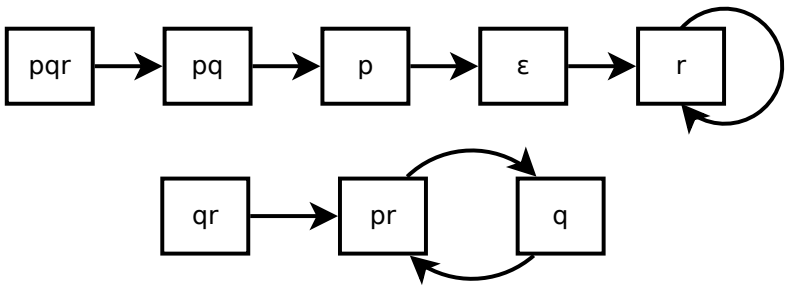
$$p(t{+}1) \leftarrow q(t).$$
$$q(t{+}1) \leftarrow p(t) \wedge r(t).$$
$$r(t{+}1) \leftarrow \neg p(t).$$

Representation: Logic Progam

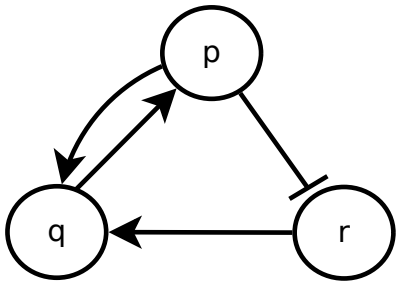
Learning From Interpretation Transitions (LFIT)

A framework for learning system dynamics from state transitions.

- Basic Idea:
  - Learn a logic program by observing the behavior of a system.
  - This logic program represents the dynamics of the system.



What happens



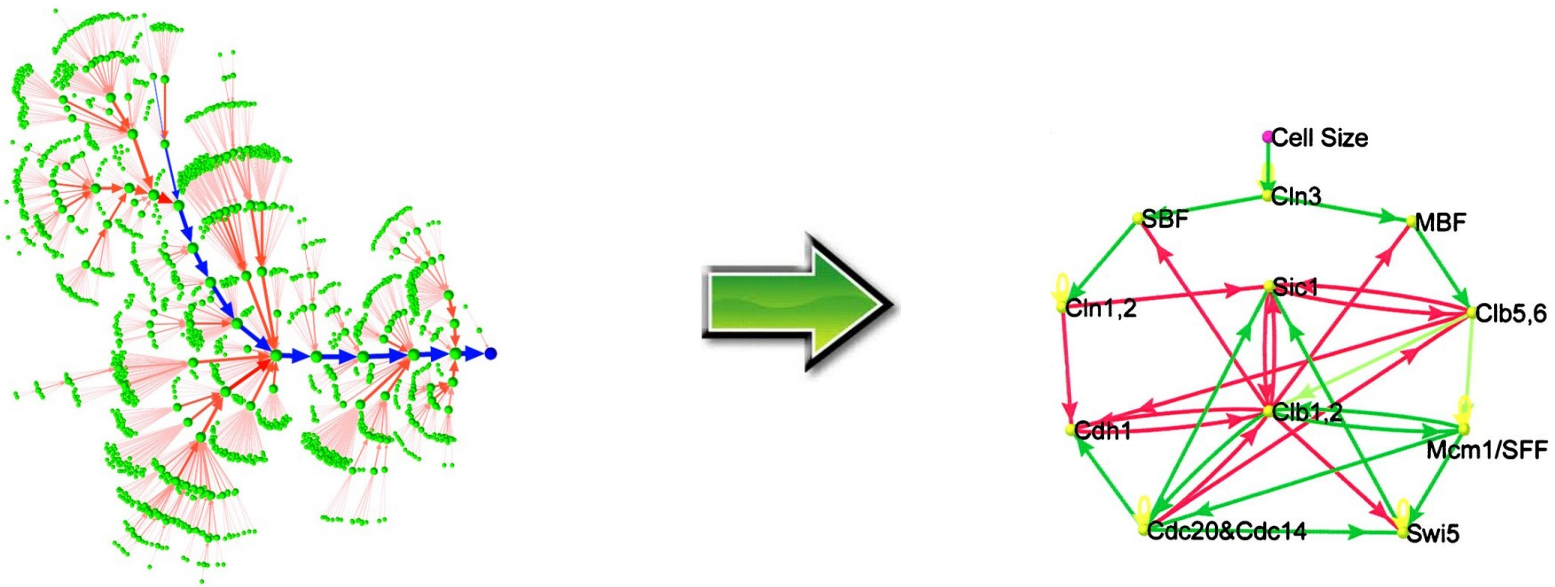
Why it happens

$$p(t+1) \leftarrow q(t).$$
$$q(t+1) \leftarrow p(t) \wedge r(t).$$
$$r(t+1) \leftarrow \neg p(t).$$

Representation: Logic Progam

Applications in Bioinformatics

Construction of gene regulatory networks.



A state transition diagram (left) and the Boolean network of the *budding yeast* (right) (*Li et al. 2004*).

Outline

Outline

Outline

Background: Machine Learning

Arthur Samuel, 1959: "Field of study that gives computers the ability to learn without being explicitly programmed."

The learning algorithms can be categorized according to the learning style they use:

Supervised learning

Example: given a set of samples we want to learn what is a cat.

The learning algorithms can be categorized according to the learning style they use:

Unsupervised learning

Example: given a set of samples, classify them into two groups.

The learning algorithms can be categorized according to the learning style they use:

Unsupervised learning

Example: given a set of samples, classify them into two groups.

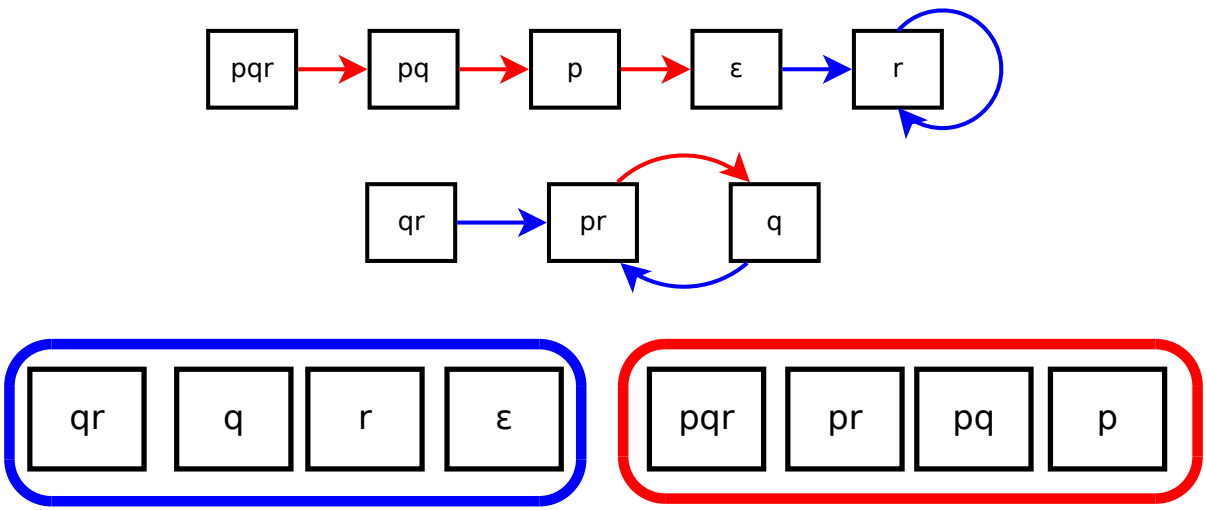
The learning algorithms can be categorized according to the learning style they use:

Unsupervised learning

Example: given a set of samples, classify them.

Example

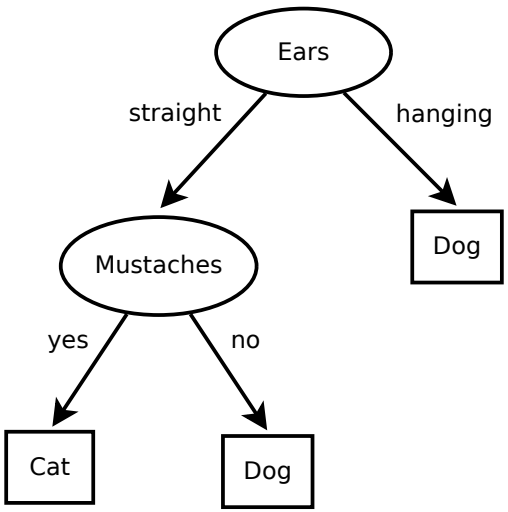
Our algorithms belong to supervised learning: samples are states labeled according to a variable value in next state:

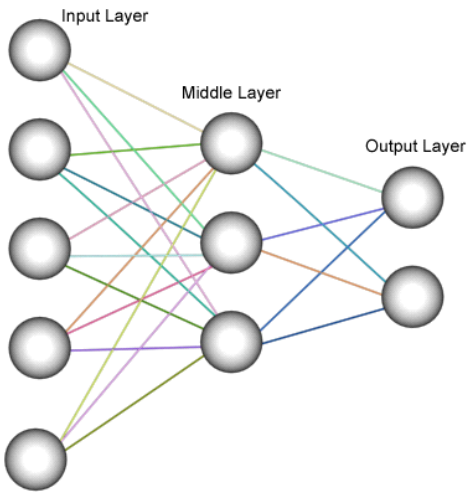


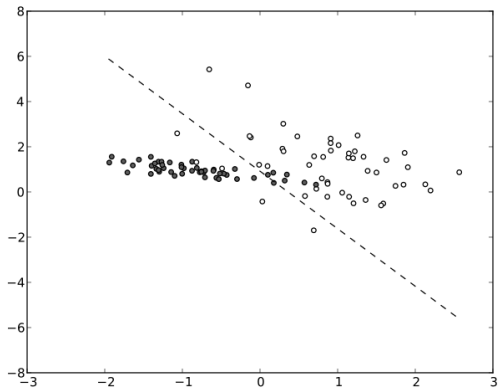
The states where r is in next state and the ones where r is not.

Example

From the samples, a machine learning algorithm learns a model:







There exists several model representations: decision tree, neural network, support vector machine...

Example

From the samples, a machine learning algorithm learns a model:

Logic rules

Cats if ears=straight and whiskers=yes.
Dogs if ears=hanging or whiskers=no.

Example: rules which define cats and dogs.

Outline

Background: Logic Programming

Logic Programming is a form of programming, where a program is defined by a set of elementary facts and logic rules associating these facts with their (more or less) direct consequences.

$$cat \leftarrow ears\_straight \wedge whiskers.$$
$$dog \leftarrow ears\_hanging.$$
$$dog \leftarrow \neg whiskers.$$

Example: a logic program that defines what is *cat* and *dog*.

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \quad (1)$$

where p and p_i's are atoms $(n > m \geq 0)$.

Definition (Herbrand base)

The Herbrand base of a logic program P, denoted by $\mathcal{B}$, is the set of all ground atoms in the language of P.

Example

- $R_1 = cat \leftarrow ears\_straight \wedge whiskers$
- $R_2 = dog \leftarrow ears\_hanging$
- $R_3 = dog \leftarrow \neg whiskers$
- $P = \{R_1, R_2, R_3\}$ is a logic program
- The Herbrand base of P is $\mathcal{B} = \{cat, dog, ears\_straight, ears\_hanging, whiskers\}$

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \tag{1}$$

- left part of $\leftarrow$ is the *head* denoted $h(R)$
- the conjunction to the right of $\leftarrow$ is the *body* denoted $b(R)$
- the positive part of the body is $b^+(R)$
- the negative part of the body is $b^-(R)$

Definition (Interpretation)

Let $\mathcal{B}$ be the Herbrand base of a logic program P. An *interpretation* is a subset of $\mathcal{B}$.

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \tag{1}$$

- left part of $\leftarrow$ is the *head* denoted $h(R)$
- the conjunction to the right of $\leftarrow$ is the *body* denoted $b(R)$
- the positive part of the body is $b^+(R)$
- the negative part of the body is $b^-(R)$

Definition (Interpretation)

Let $\mathcal{B}$ be the Herbrand base of a logic program P. An *interpretation* is a subset of $\mathcal{B}$.

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \tag{1}$$

- left part of $\leftarrow$ is the *head* denoted $h(R)$
- the conjunction to the right of $\leftarrow$ is the *body* denoted $b(R)$
- the positive part of the body is $b^+(R)$
- the negative part of the body is $b^-(R)$

Definition (Interpretation)

Let $\mathcal{B}$ be the Herbrand base of a logic program P. An *interpretation* is a subset of $\mathcal{B}$.

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \tag{1}$$

- left part of $\leftarrow$ is the *head* denoted $h(R)$
- the conjunction to the right of $\leftarrow$ is the *body* denoted $b(R)$
- the positive part of the body is $b^+(R)$
- the negative part of the body is $b^-(R)$

Definition (Interpretation)

Let $\mathcal{B}$ be the Herbrand base of a logic program P. An *interpretation* is a subset of $\mathcal{B}$.

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \tag{1}$$

- left part of $\leftarrow$ is the *head* denoted $h(R)$
- the conjunction to the right of $\leftarrow$ is the *body* denoted $b(R)$
- the positive part of the body is $b^+(R)$
- the negative part of the body is $b^-(R)$

Definition (Interpretation)

Let $\mathcal{B}$ be the Herbrand base of a logic program P. An *interpretation* is a subset of $\mathcal{B}$.

We consider a propositional *logic program* P as a set of *rules* R of the form

$$R := p \leftarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_n \tag{1}$$

- left part of $\leftarrow$ is the *head* denoted $h(R)$
- the conjunction to the right of $\leftarrow$ is the *body* denoted $b(R)$
- the positive part of the body is $b^+(R)$
- the negative part of the body is $b^-(R)$

Definition (T_P Operator)

For a logic program P and an interpretation I, the *immediate consequence operator* (or T_P *operator*) is the mapping $T_P : 2^{\mathcal{B}} \rightarrow 2^{\mathcal{B}}$:

$$T_P(I) = \{\, h(R) \mid R \in P,\, b^+(R) \subseteq I,\, b^-(R) \cap I = \emptyset \}. \tag{2}$$

The state transitions of a logic program P can be represented as a set of pairs of interpretations $(I, T_P(I))$.

In our methods, we represent and learn the dynamics of a system: when things happen, in which circonstances a variable takes which value. Using the T_P operator we can reproduce the state transitions.

$$p(t+1) \leftarrow q(t).$$
$$q(t+1) \leftarrow p(t) \wedge r(t).$$
$$r(t+1) \leftarrow \neg p(t).$$

In our methods, we represent and learn the dynamics of a system: when things happen, in which circonstances a variable takes which value. Using the T_P operator we can reproduce the state transitions.

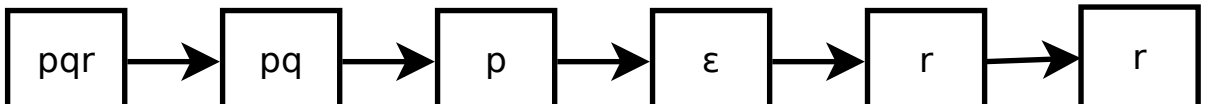
$$p \leftarrow q.$$
$$q \leftarrow p \wedge r.$$
$$r \leftarrow \neg p.$$

In our methods, we represent and learn the dynamics of a system: when things happen, in which circonstances a variable takes which value. Using the T_P operator we can reproduce the state transitions.

$$p \leftarrow q.$$
$$q \leftarrow p \wedge r.$$
$$r \leftarrow \neg p.$$

Example

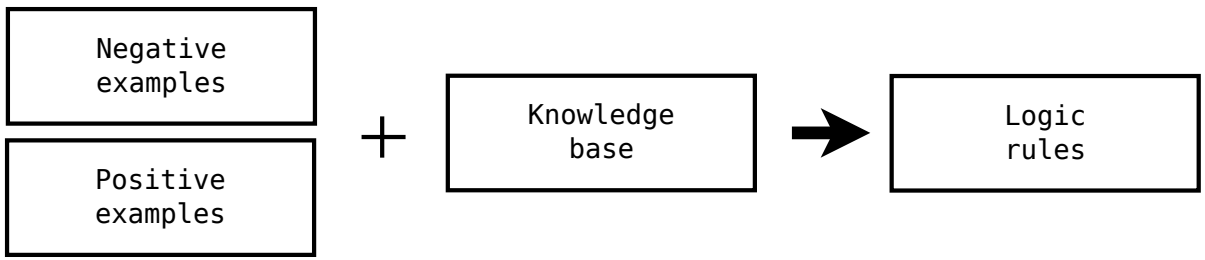
- $T_P(\{p, q, r\}) = \{p, q\}$
- $T_P(\{p, q\}) = \{p\}$
- $T_P(\{p\}) = \emptyset$
- $T_P(\emptyset) = \{r\}$
- $T_P(\{r\}) = \{r\}$



Outline

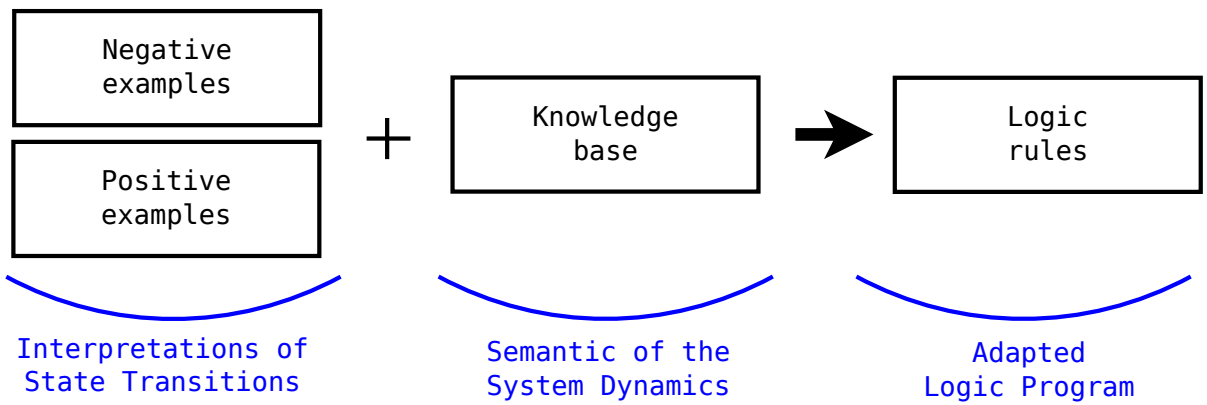
Background: Inductive Logic Programming

What we do is a form of Inductive Logic Programming (ILP). From a base of positive and negative examples, an ILP system tries to deduce a logic program which confirms the positive examples and refutes the negative ones.



Background: Inductive Logic Programming

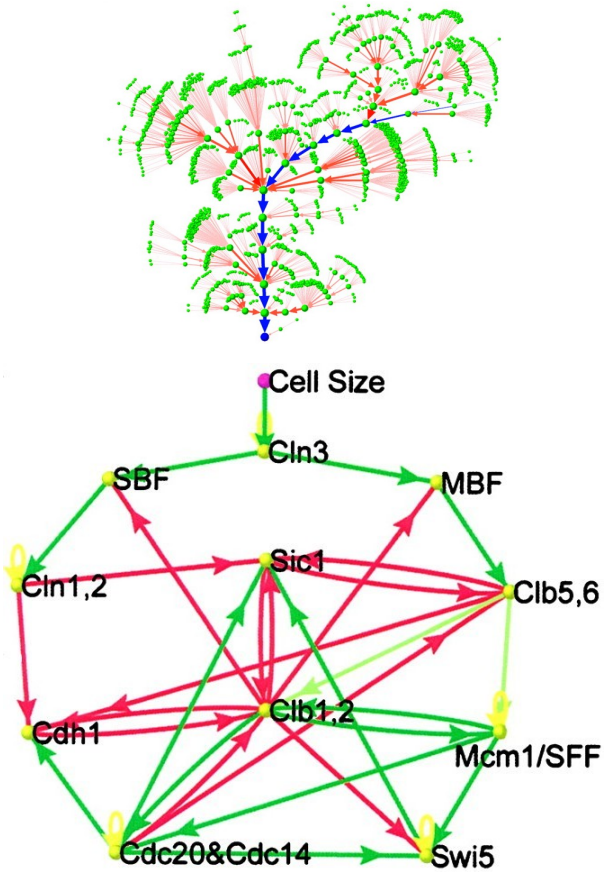
What we do is a form of Inductive Logic Programming (ILP). From a base of positive and negative examples, an ILP system tries to deduce a logic program which confirms the positive examples and refutes the negative ones.

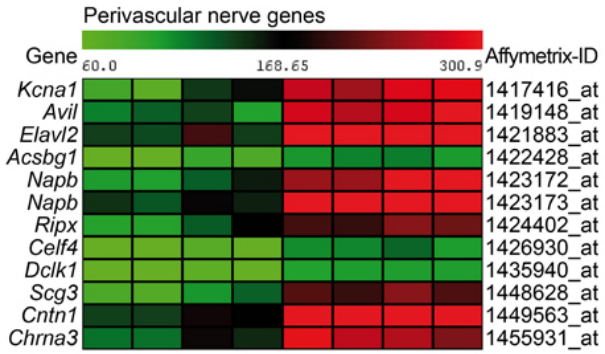


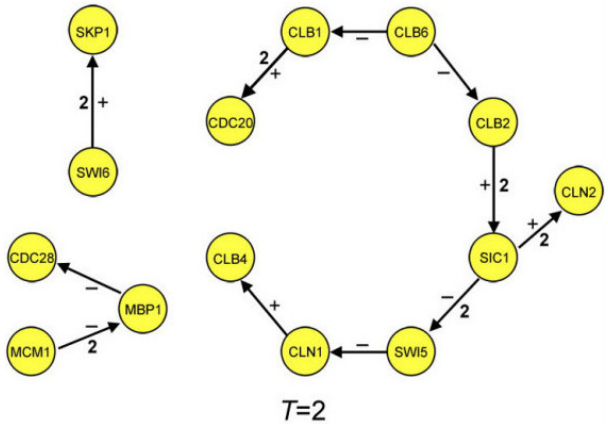
Outline

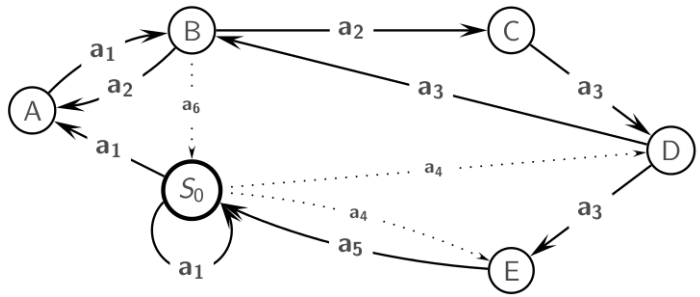
Contribution

We propose several algorithms that capture different types of dynamics.









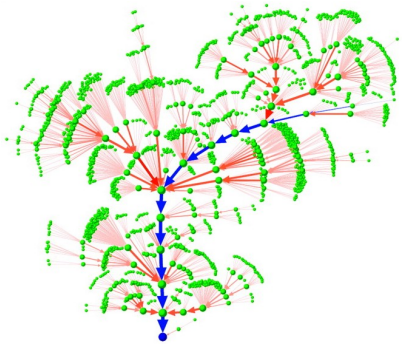
Memory-less Systems

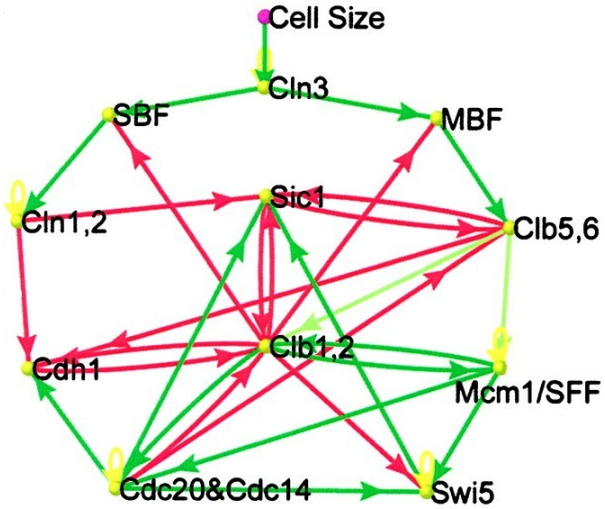
Systems with delays

Non-deterministic Systems

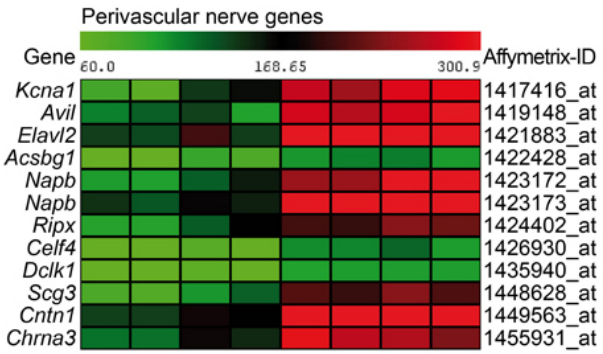
Contribution

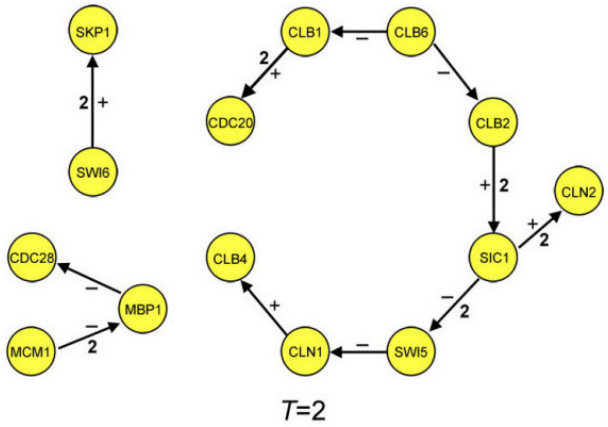
Memory-less models: Boolean network, gene regulatory network, neural network and Petri net.



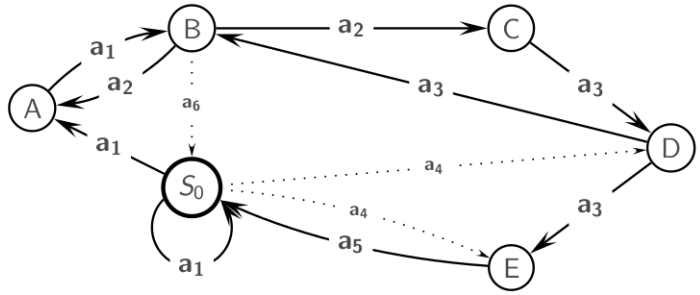


Memory-less Systems





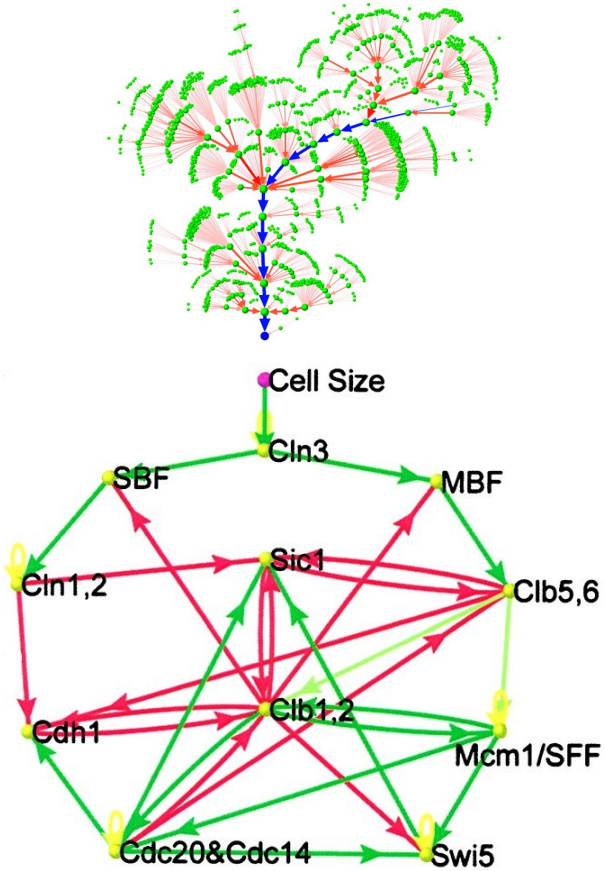
Systems with delays

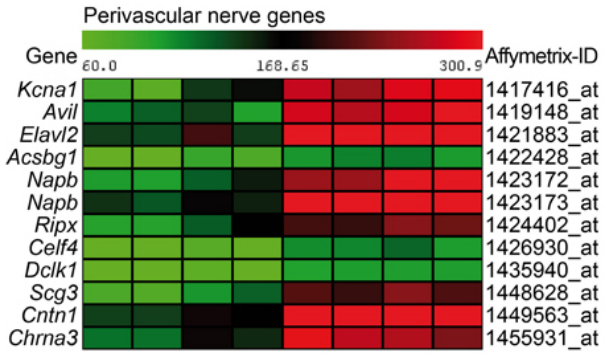


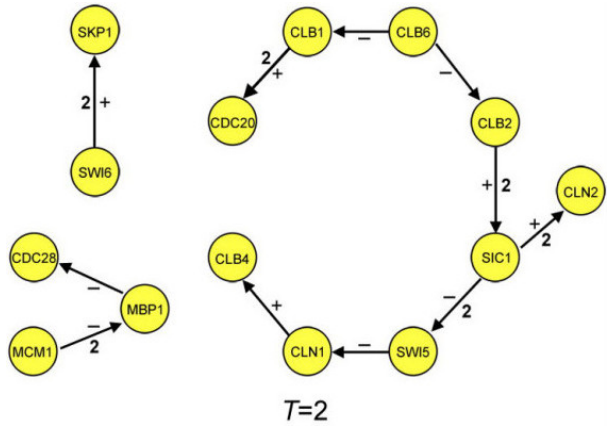
Non-deterministic Systems

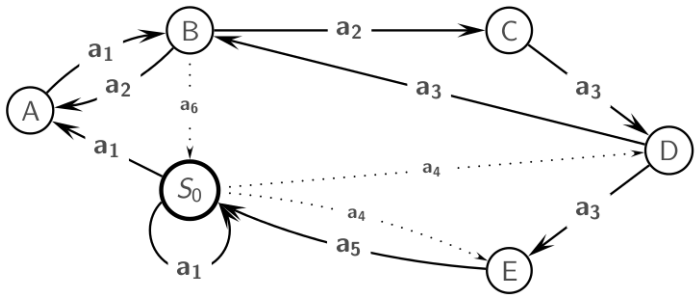
Contribution

Dynamics with duration, accumulation, i.e. delays. Delayed models: timed Boolean network, timed Petri net.









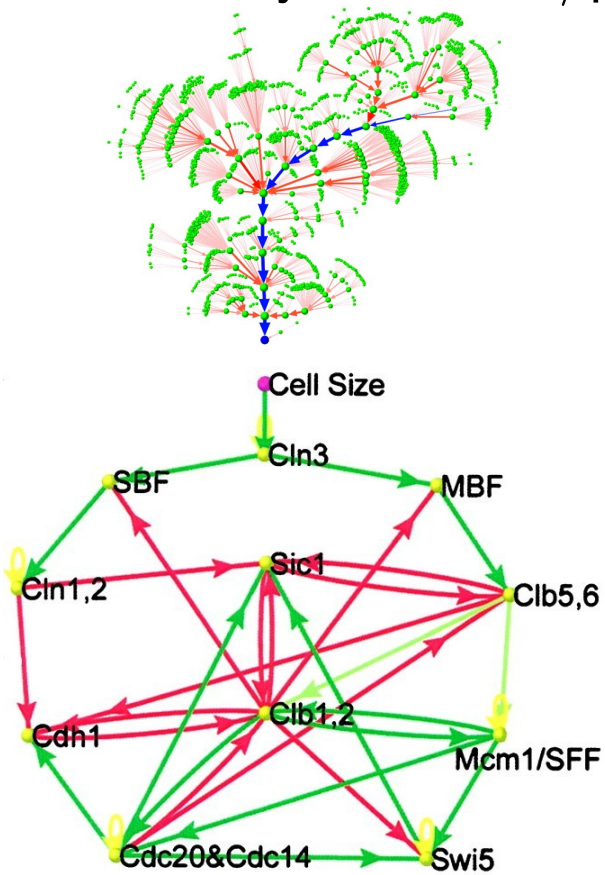
Memory-less Systems

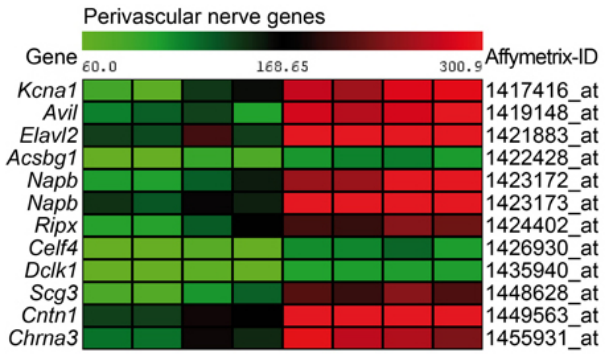
Systems with delays

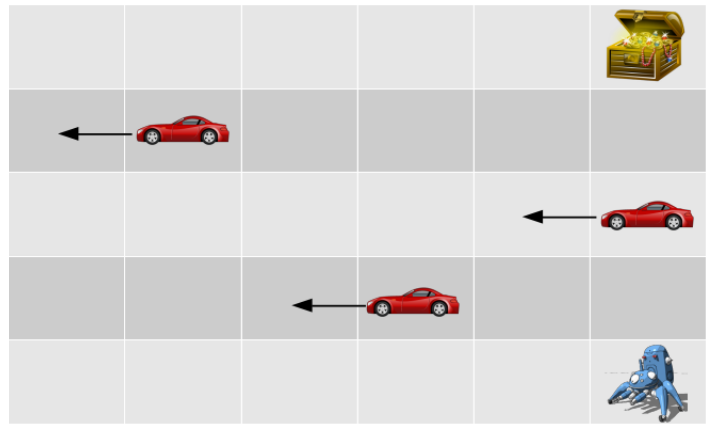
Non-deterministic Systems

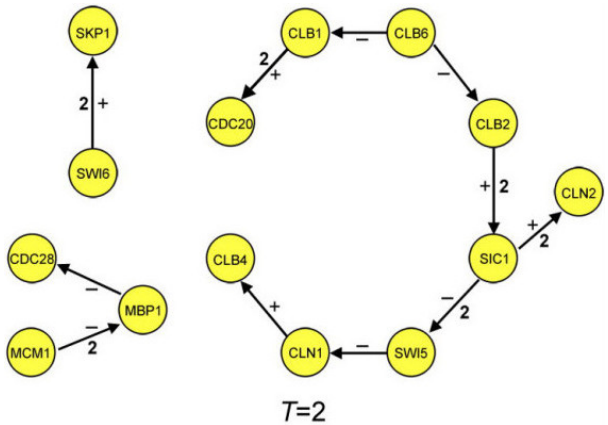
Contribution

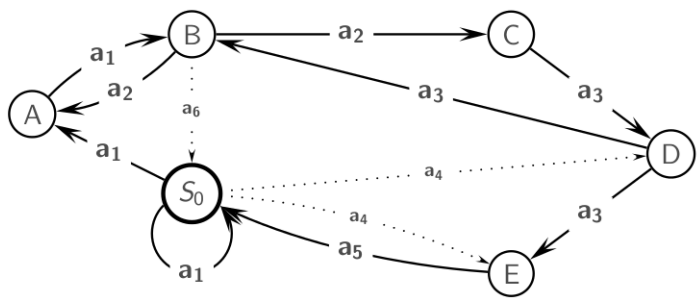
From one state, multiple futures. Non-deterministic models: asynchronous/probabilistic Boolean network, action model.











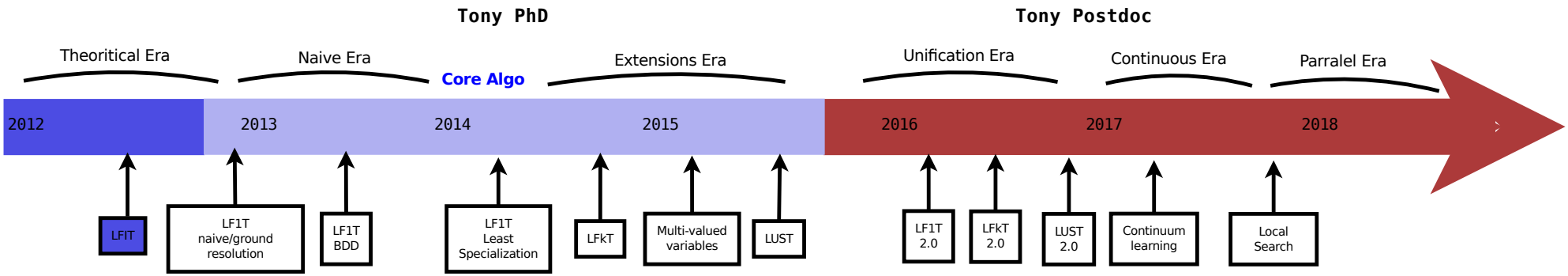
Memory-less Systems

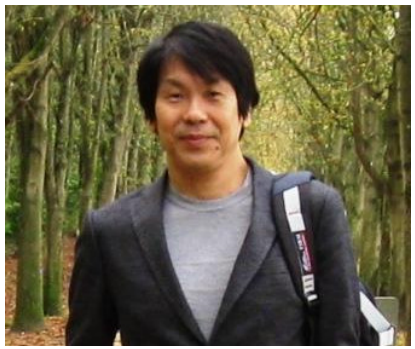
Systems with delays

Non-deterministic Systems

Outline

LFIT Chronology



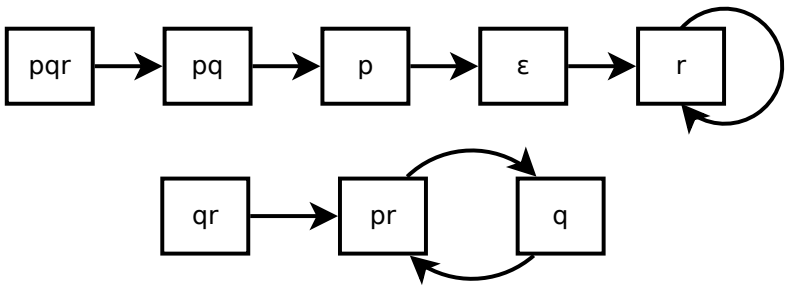


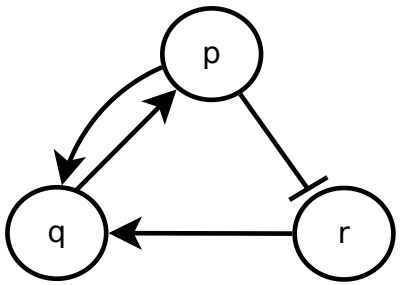
Short paper **ILP 2012**.

Learning From Interpretation Transitions (LFIT)

A framework for learning system dynamics from state transitions.

- Basic Idea:
  - Learn a logic program by observing the behavior of a system.
  - This logic program represents the dynamics of the system.





Input: Behavior of the system

Output: Dynamics of the system

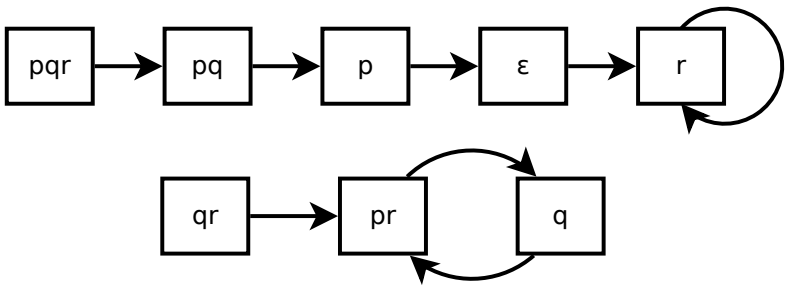
$$p(t+1) \leftarrow q(t).$$
$$q(t+1) \leftarrow p(t) \wedge r(t).$$
$$r(t+1) \leftarrow \neg p(t).$$

Representation: Logic Progam

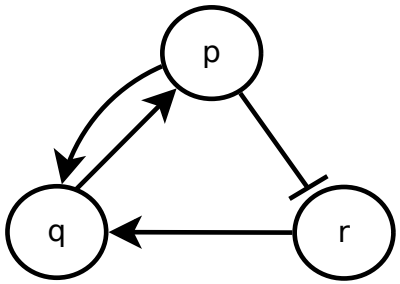
Learning From Interpretation Transitions (LFIT)

A framework for learning system dynamics from state transitions.

- Basic Idea:
  - Learn a logic program by observing the behavior of a system.
  - This logic program represents the dynamics of the system.



What happens

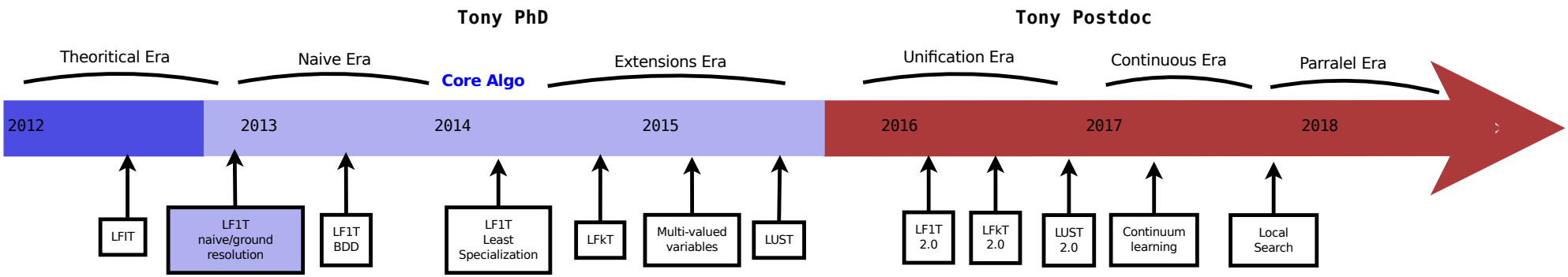


Why it happens

$$p(t+1) \leftarrow q(t).$$
$$q(t+1) \leftarrow p(t) \wedge r(t).$$
$$r(t+1) \leftarrow \neg p(t).$$

Representation: Logic Progam

LFIT Chronology

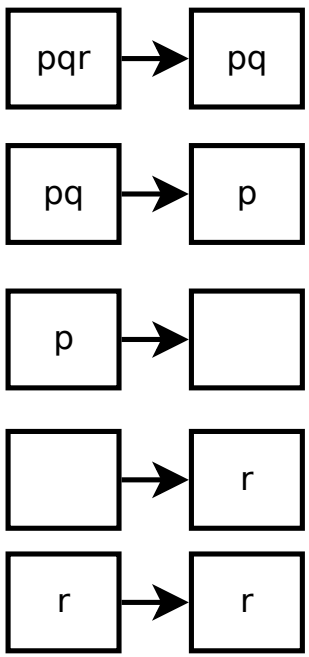




Long paper **ILP 2013/2014**, journal **MLJ 2014**.

LF1T: Learning From 1-step Transitions (memory-less systems)

INPUT:
A set of pairs of interpretations



OUTPUT:
A normal logic program

$$p(t{+}1) \leftarrow q(t).$$
$$q(t{+}1) \leftarrow p(t) \wedge r(t).$$
$$r(t{+}1) \leftarrow \neg p(t).$$

Learning from 1-step transition (LF1T) *(Inoue et al. MLJ 2013)*

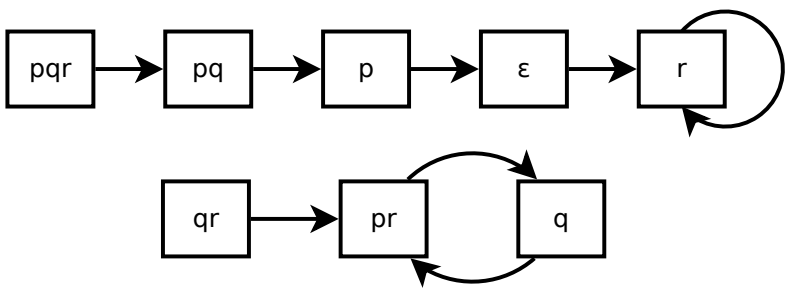
Input:

- The variables of the systems (pqr)
- A set E of state transitions (trace of executions)

Output:

- A set of rules P which represents the dynamics of the system

State Transitions



LF1T Input

$$E = \{\ (pqr, pq),\ (pq, p),\ (p, \emptyset),\ (\emptyset, r),\ (r, r),\ (qr, pr)\ (pr, q)\ (q, pr)\ \}$$

Learning from 1 step transition *(Inoue et al. MLJ 2013)*

Let $(I, J) \in E$, for each $A \in J$ we can infer a positive rule R_A^I:

$$R_A^I := (A \leftarrow \bigwedge_{B_i \in I} B_i \wedge \bigwedge_{C_j \in \beta \setminus I} \neg C_j)$$

Example

From the state transition (pqr, pq) we can infer 2 rules:

- $p \leftarrow p \wedge q \wedge r$.
- $q \leftarrow p \wedge q \wedge r$.

Learning from 1 step transition *(Inoue et al. MLJ 2013)*

Let $(I, J) \in E$, for each $A \in J$ we can infer a positive rule R_A^I:

$$R_A^I := (A \leftarrow \bigwedge_{B_i \in I} B_i \wedge \bigwedge_{C_j \in \beta \setminus I} \neg C_j)$$

Example

From the next transition (pq, p) we can infer 1 logic rule:

- $p \leftarrow p \wedge q \wedge \neg r$.

Generalization Techniques

Generalize knowledge to learn the real relationship.

- $p \leftarrow p \wedge q \wedge r$.
- $p \leftarrow p \wedge q \wedge \neg r$.

Generalization Techniques

Generalize knowledge to learn the real relationship.

- $p \leftarrow p \wedge q \wedge r$.
- $p \leftarrow p \wedge q \wedge \neg r$.
- $p \leftarrow p \wedge q$.

Generalization Techniques

Generalize knowledge to learn the real relationship.

- $p \leftarrow p \wedge q$.
- $p \leftarrow \neg p \wedge q$.

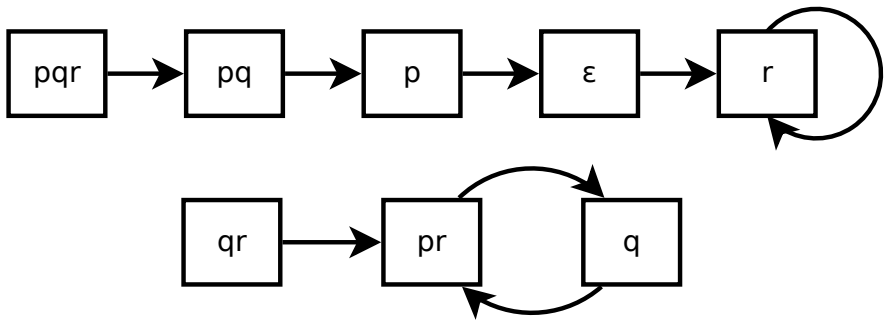
Generalization Techniques

Generalize knowledge to learn the real relationship.

- $p \leftarrow p \wedge q$.
- $p \leftarrow \neg p \wedge q$.
- $p \leftarrow q$.

Running Example

Input State Transitions

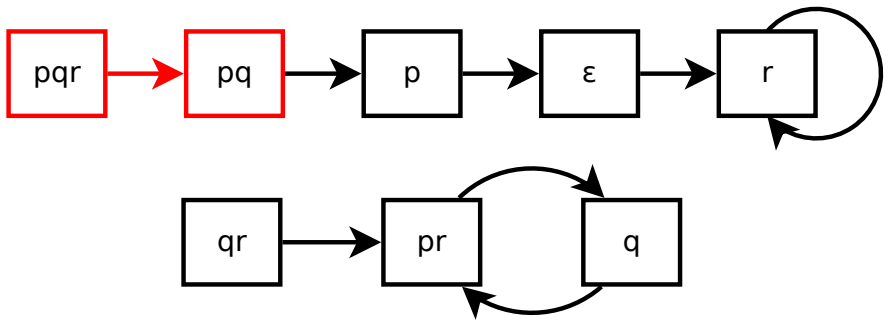


Learned Rules:
$\emptyset$

Knowledge Base:
$\emptyset$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow p \wedge q \wedge r.$

$q \leftarrow p \wedge q \wedge r.$

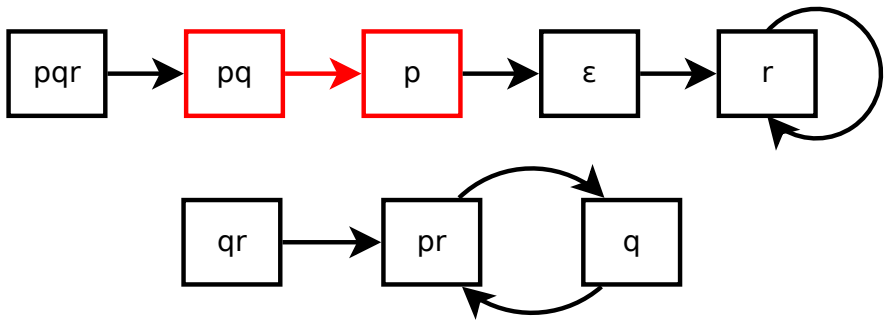
Knowledge Base:

$p \leftarrow p \wedge q \wedge r.$

$q \leftarrow p \wedge q \wedge r.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow p \wedge q \wedge \neg r.$

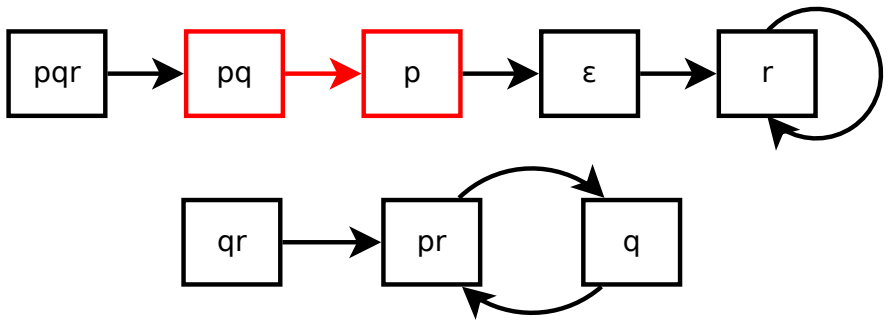
Knowledge Base:

$p \leftarrow p \wedge q \wedge r.$

$q \leftarrow p \wedge q \wedge r.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow p \wedge q \wedge \neg r.$

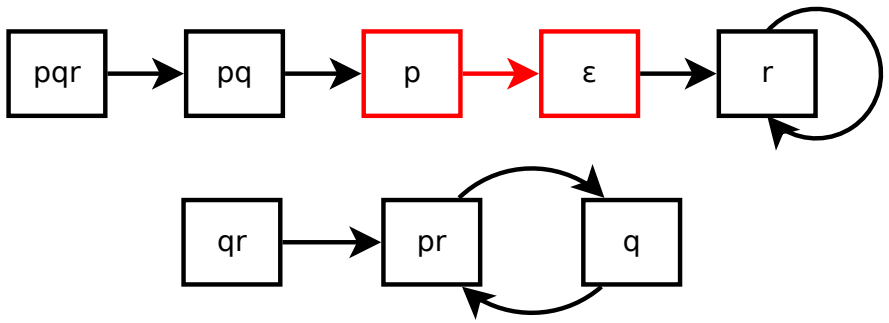
Knowledge Base:

$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge q \wedge r.$

Running Example

Input State Transitions

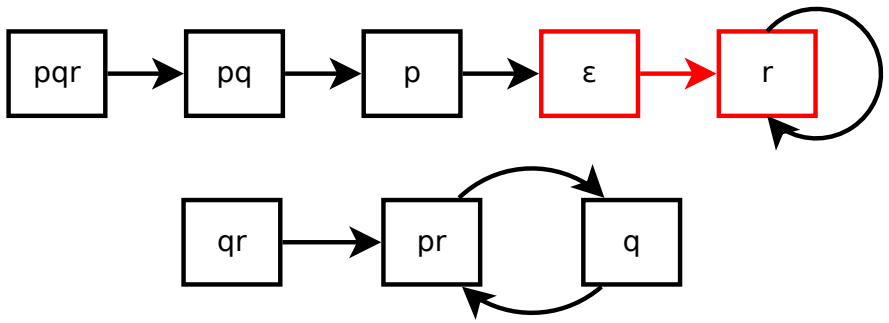


Learned Rules:
$\emptyset$

Knowledge Base:
$p \leftarrow p \wedge q.$
$q \leftarrow p \wedge q \wedge r.$

Running Example

Input State Transitions



Learned Rules:

$r \leftarrow \neg p \wedge \neg q \wedge \neg r.$

Knowledge Base:

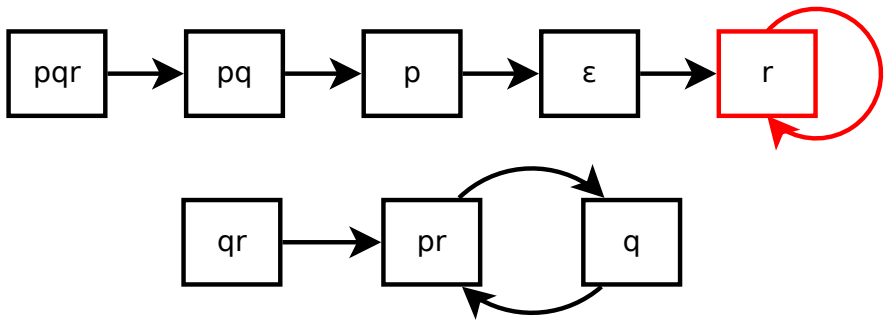
$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge q \wedge r.$

$r \leftarrow \neg p \wedge \neg q \wedge \neg r.$

Running Example

Input State Transitions



Learned Rules:

$r \leftarrow \neg p \wedge \neg q \wedge r.$

Knowledge Base:

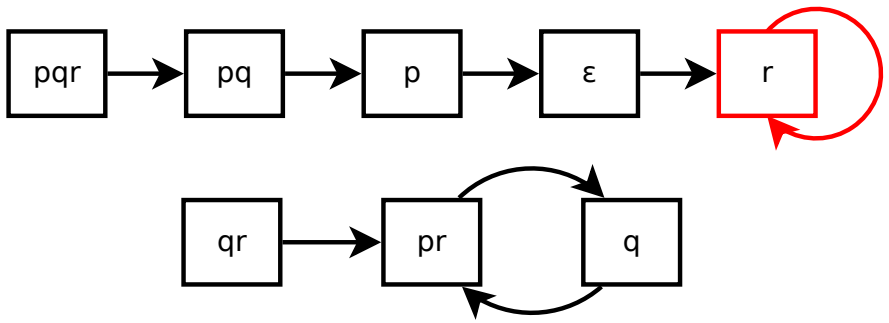
$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge q \wedge r.$

$r \leftarrow \neg p \wedge \neg q \wedge \neg r.$

Running Example

Input State Transitions



Learned Rules:

$r \leftarrow \neg p \wedge \neg q \wedge r.$

Knowledge Base:

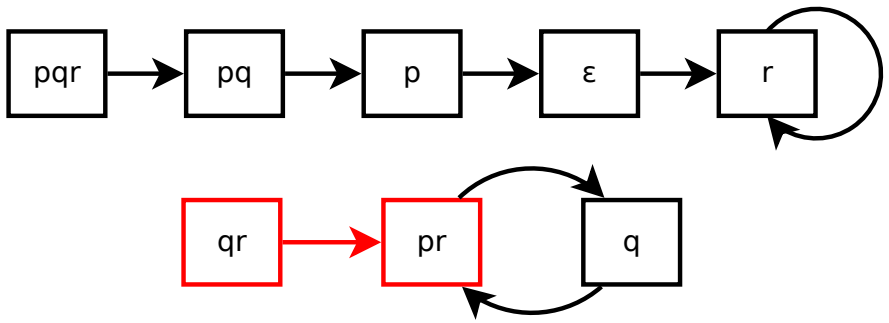
$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge q \wedge r.$

$r \leftarrow \neg p \wedge \neg q.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow \neg p \wedge q \wedge r.$

$r \leftarrow \neg p \wedge q \wedge r.$

Knowledge Base:

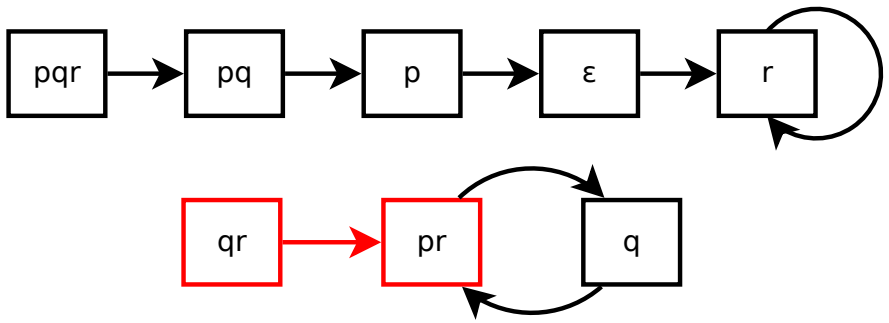
$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge q \wedge r.$

$r \leftarrow \neg p \wedge \neg q.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow \neg p \wedge q \wedge r.$

$r \leftarrow \neg p \wedge q \wedge r.$

Knowledge Base:

$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge q \wedge r.$

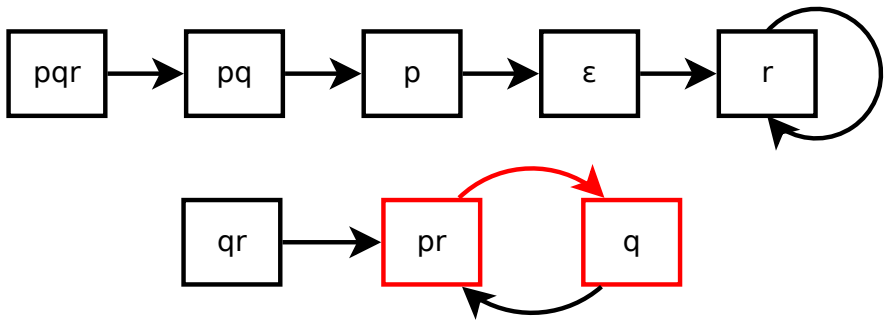
$r \leftarrow \neg p \wedge \neg q.$

$p \leftarrow q \wedge r.$

$r \leftarrow \neg p \wedge r.$

Running Example

Input State Transitions



Learned Rules:

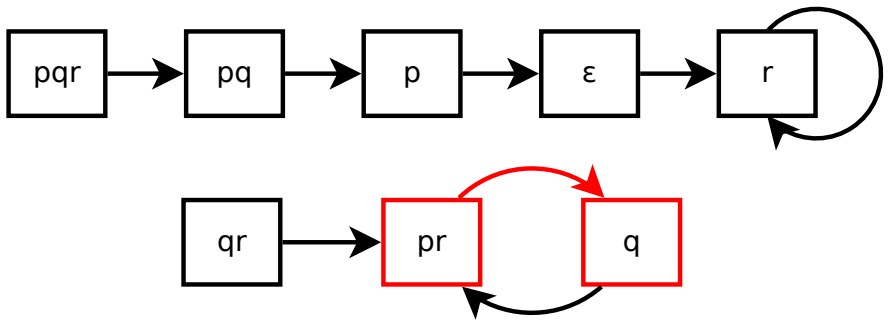
$q \leftarrow p \wedge \neg q \wedge r.$

Knowledge Base:

$p \leftarrow p \wedge q.$
$q \leftarrow p \wedge q \wedge r.$
$r \leftarrow \neg p \wedge \neg q.$
$p \leftarrow q \wedge r.$
$r \leftarrow \neg p \wedge r.$

Running Example

Input State Transitions



Learned Rules:

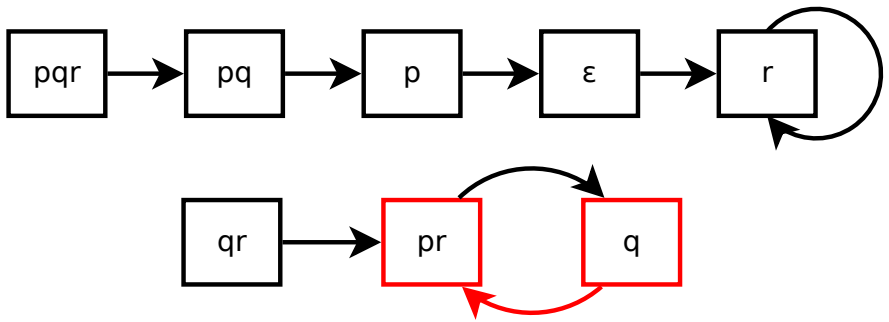
$q \leftarrow p \wedge \neg q \wedge r.$

Knowledge Base:

$p \leftarrow p \wedge q.$
$q \leftarrow p \wedge r.$
$r \leftarrow \neg p \wedge \neg q.$
$p \leftarrow q \wedge r.$
$r \leftarrow \neg p \wedge r.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow \neg p \wedge q \wedge \neg r.$

$r \leftarrow \neg p \wedge q \wedge \neg r.$

Knowledge Base:

$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge r.$

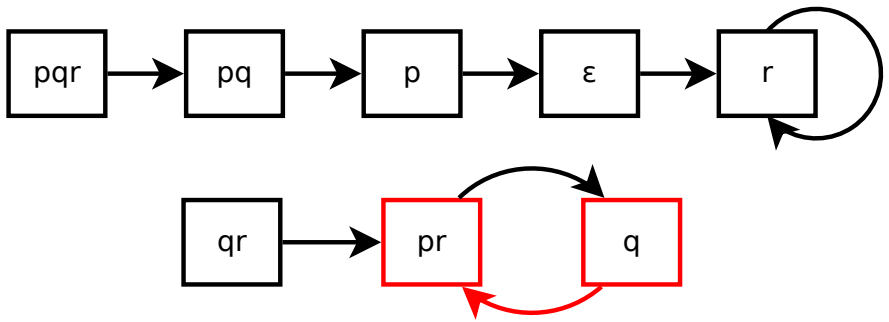
$r \leftarrow \neg p \wedge \neg q.$

$p \leftarrow q \wedge r.$

$r \leftarrow \neg p \wedge r.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow q \wedge \neg r.$

$r \leftarrow \neg p \wedge \neg r.$

Knowledge Base:

$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge r.$

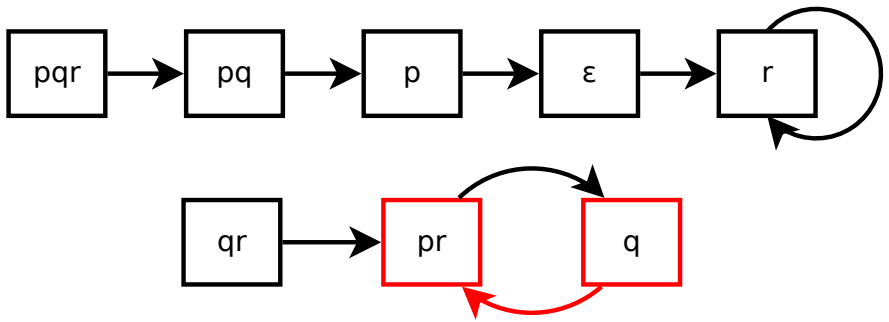
$r \leftarrow \neg p \wedge \neg q.$

$p \leftarrow q \wedge r.$

$r \leftarrow \neg p \wedge r.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow q \wedge \neg r.$

$r \leftarrow \neg p \wedge \neg r.$

Knowledge Base:

$p \leftarrow p \wedge q.$

$q \leftarrow p \wedge r.$

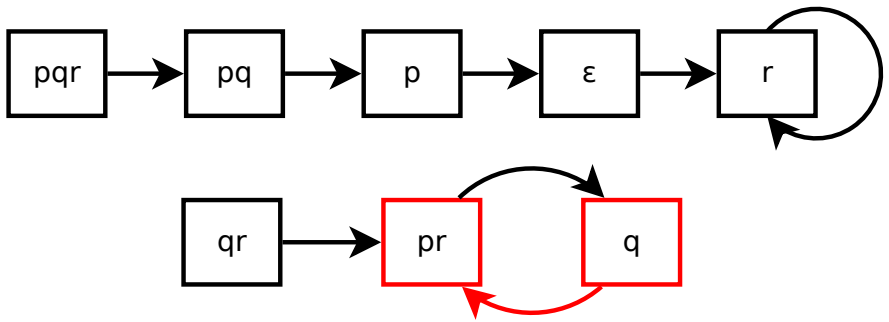
$r \leftarrow \neg p \wedge \neg q.$

$p \leftarrow q.$

$r \leftarrow \neg p.$

Running Example

Input State Transitions



Learned Rules:

$p \leftarrow q \wedge \neg r.$

$r \leftarrow \neg p \wedge \neg r.$

Knowledge Base:

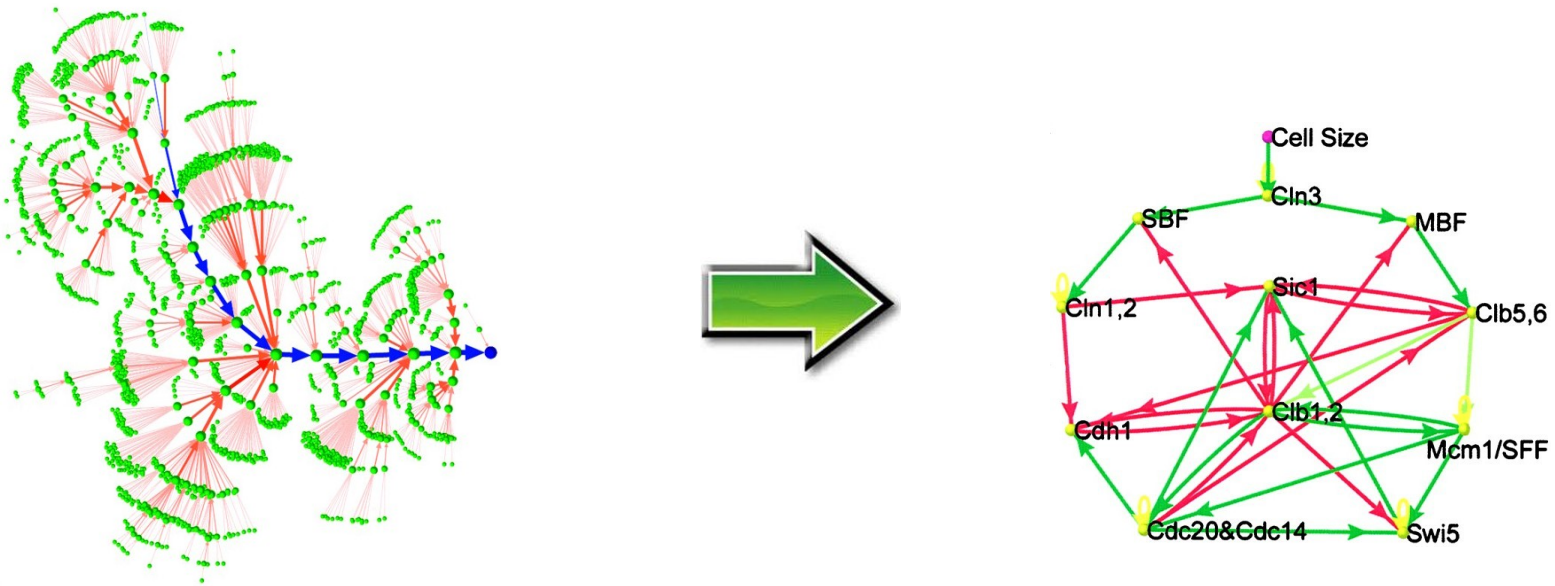
$q \leftarrow p \wedge r.$

$p \leftarrow q.$

$r \leftarrow \neg p.$

Applications in Bioinformatics

Given the state transitions of the *budding yeast* we can learn its gene regulatory networks.



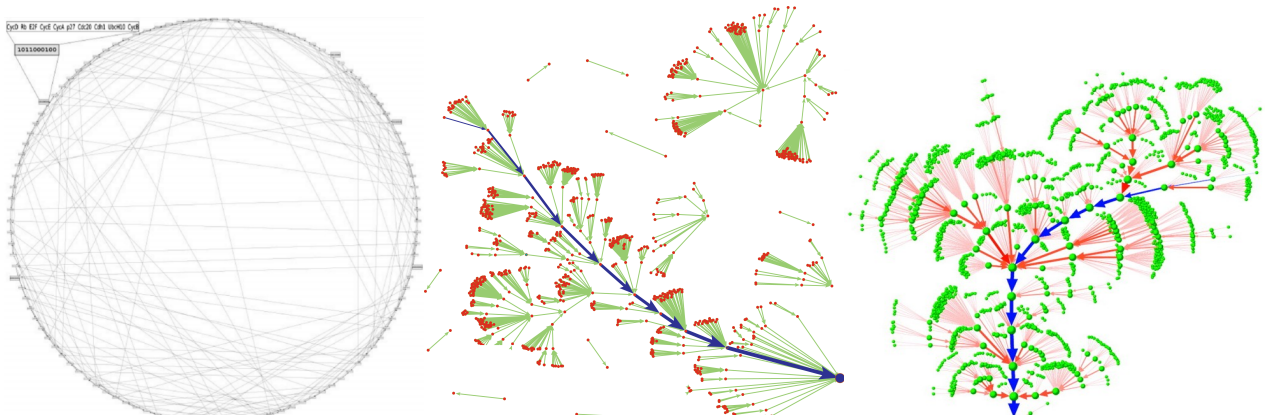
A state transition diagram (left) and the Boolean network of the *budding yeast* (right) (*Li et al. 2004*).

Evaluation

Memory use and learning time of **LF1T** for Boolean networks benchmarks up to 23 nodes

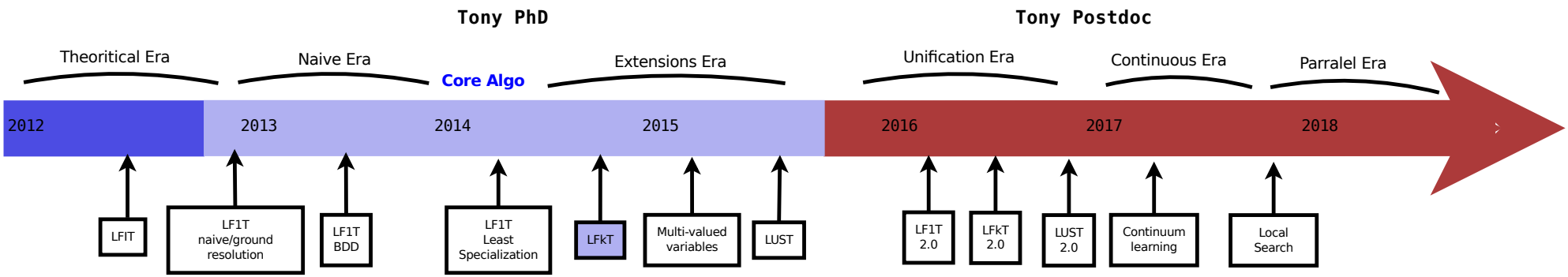
Algorithm	Mammalian (10)	Fission (10)	Budding (12)	Arabidopsis (16)	T helper (23)
Naïve	142 118/4.62s	126 237/3.65s	1 147 124/523s	T.O.	T.O.
Ground	1036/**0.04s**	1218/**0.05s**	21 470/0.26s	271 288/4.25s	T.O.
Ground-BDD	**180**/0.24s	**147**/0.24s	**541**/**0.19s**	**779**/**2.8s**	**611**/**3360s**
Full Naïve	377 539/29.25s	345587/24.03s	T.O.	T.O.	T.O.
Full Ground	1066/0.24s	1178/0.23s	23 738/4.04s	399 469/111s	T.O.
Least Specialization	**375**/**0.06s**	**377**/**0.08s**	**641**/**0.35s**	**2270**/**5.28s**	**3134**/**5263s**

Memory use and learning time of **LF1T** for Boolean networks benchmarks up to 23 nodes



Outline

LFIT Chronology

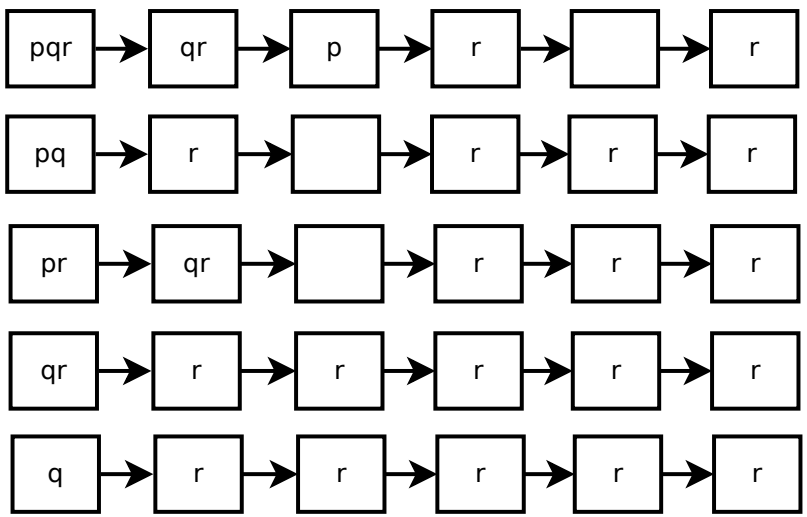




Short paper **ILP 2014/2015**, journal **Frontiers 2015**, long paper **ICMLA 2015**.

LFkT: Learning From k-step Transitions (markov(k) systems)

INPUT:
A set of sequences of interpretations



OUTPUT:
A normal logic program with delay

$$p(t) \leftarrow q(t-1) \wedge q(t-2).$$
$$q(t) \leftarrow p(t-1) \wedge r(t-1).$$
$$r(t) \leftarrow \neg p(t-2).$$

Formalization: Markov(k) into Logic Program

Definition (Timed Herbrand base)

Let $\mathcal{B}$ be the Herbrand base of a program P and k be a natural number. The timed Herbrand base of P (with period k) denoted by $\mathcal{B}_k$, is as follows:

$$\mathcal{B}_k = \bigcup_{i=1}^{k} \{v_{t-i} | v \in \mathcal{B}\}$$

Where t is a constant term which represents the current time step.

Example

If the Herbrand base of a program P is $\mathcal{B} = \{a, b, c\}$ then

- $\mathcal{B}_1 = \{a_{t-1}, b_{t-1}, c_{t-1}\}$
- $\mathcal{B}_2 = \{a_{t-1}, b_{t-1}, c_{t-1}, a_{t-2}, b_{t-2}, c_{t-2}\}$

Markov(k) Systems

A Markov(k) system can be interpreted as a logic program.

Definition (Markov(k) system)

Let P be a logic program, $\mathcal{B}$ be the Herbrand base of P and $\mathcal{B}_k$ be the timed Herbrand base of P with period k. A Markov(k) system S with respect to P is a logic program where for all rules $R \in S$, $h(R) \in \mathcal{B}$ and all atoms appearing in b(R) belong to $\mathcal{B}_k$.

Example

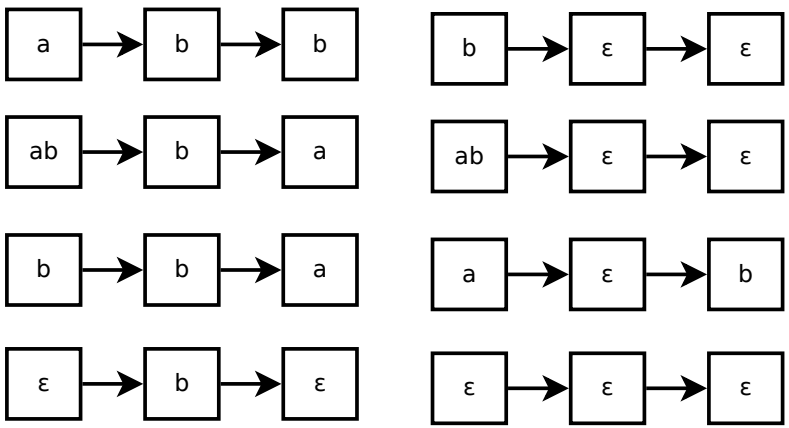
If the Herbrand base of a program P is $\mathcal{B} = \{a, b\}$ then

- $\mathcal{B}_2 = \{a_{t-1}, b_{t-1}, a_{t-2}, b_{t-2}\}$.
- Let $R_1 = a \leftarrow b_{t-1}, b_{t-2}$ and $R_2 = b \leftarrow a_{t-2}, \neg b_{t-2}$.
- $S = \{R_1, R_2\}$ is a Markov(2) system.

Example

Let S be a Markov(k) system as follows:

$$S = \{(a \leftarrow b_{t-1}, b_{t-2}), (b \leftarrow a_{t-2}, \neg b_{t-2})\}$$



Eight traces of executions of the system S

Running Example

Initialization		$a \rightarrow b \rightarrow b$		$ab \rightarrow b \rightarrow a$	
2-step program	1-step program				
a. b.	a. b.				

Running Example

Initialization		$a \to b \to b$		$ab \to b \to a$	
2-step program	1-step program	$(a_2 b_1, b)$			
$a.$ $b.$	$a.$ $b.$	$a \leftarrow \neg \mathbf{a_2}.$ $a \leftarrow \mathbf{b_2}.$ $a \leftarrow \mathbf{a_1}.$ $a \leftarrow \neg \mathbf{b_1}.$ $b.$			

Running Example

Initialization		$a \to b \to b$		$ab \to b \to a$	
2-step program	1-step program	$(a_2 b_1, b)$	(a_1, b)		
$a.$ $b.$	$a.$ $b.$	$a \leftarrow \neg \mathbf{a_2}.$ $a \leftarrow \mathbf{b_2}.$ $a \leftarrow \mathbf{a_1}.$ $a \leftarrow \neg \mathbf{b_1}.$ $b.$	$a \leftarrow \neg \mathbf{a_1}.$ $a \leftarrow \mathbf{b_1}.$ $b.$		

Running Example

Initialization		$a \rightarrow b \rightarrow b$		$ab \rightarrow b \rightarrow a$	
2-step program	1-step program	$(a_2 b_1, b)$	(a_1, b)		
$a.$ $b.$	$a.$ $b.$	$a \leftarrow \neg \mathbf{a_2}.$ $a \leftarrow \mathbf{b_2}.$ $a \leftarrow \mathbf{a_1}.$ $a \leftarrow \neg \mathbf{b_1}.$ $b.$	$a \leftarrow \neg \mathbf{a_1}.$ $a \leftarrow \mathbf{b_1}.$ $b.$		
			$(a_2 b_1, b)$		
			$a \leftarrow \neg a_1, \neg \mathbf{b_1}.$ $a \leftarrow \mathbf{a_1}, b_1.$ $b.$		

Running Example

Initialization		$a \rightarrow b \rightarrow b$		$ab \rightarrow b \rightarrow a$	
2-step program	1-step program	$(a_2 b_1, b)$	(a_1, b)	$(a_2 b_2 b_1, a)$	
$a.$ $b.$	$a.$ $b.$	$a \leftarrow \neg \mathbf{a_2}.$ $a \leftarrow \mathbf{b_2}.$ $a \leftarrow \mathbf{a_1}.$ $a \leftarrow \neg \mathbf{b_1}.$ $b.$	$a \leftarrow \neg \mathbf{a_1}.$ $a \leftarrow \mathbf{b_1}.$ $b.$	$a \leftarrow \neg a_2.$ $a \leftarrow b_2.$ $a \leftarrow a_1.$	
			$(a_2 b_1, b)$	$a \leftarrow \neg b_1.$	
			$a \leftarrow \neg a_1, \neg \mathbf{b_1}.$ $a \leftarrow \mathbf{a_1}, b_1.$ $b.$	$b \leftarrow \neg a_2.$ $b \leftarrow \neg b_2.$ $b \leftarrow a_1.$ $b \leftarrow \neg b_1.$	

Running Example

Initialization		$a \to b \to b$		$ab \to b \to a$	
2-step program	1-step program	$(a_2 b_1, b)$	(a_1, b)	$(a_2 b_2 b_1, a)$	$(a_1 b_1, b)$
$a.$ $b.$	$a.$ $b.$	$a \leftarrow \neg \mathbf{a_2}.$ $a \leftarrow \mathbf{b_2}.$ $a \leftarrow \mathbf{a_1}.$ $a \leftarrow \neg \mathbf{b_1}.$ $b.$	$a \leftarrow \neg \mathbf{a_1}.$ $a \leftarrow \mathbf{b_1}.$ $b.$ $(a_2 b_1, b)$ $a \leftarrow \neg a_1, \neg \mathbf{b_1}.$ $a \leftarrow \mathbf{a_1}, b_1.$ $b.$	$a \leftarrow \neg a_2.$ $a \leftarrow b_2.$ $a \leftarrow a_1.$ $a \leftarrow \neg b_1.$ $b \leftarrow \neg a_2.$ $b \leftarrow \neg b_2.$ $b \leftarrow a_1.$ $b \leftarrow \neg b_1.$	$a \leftarrow \neg a_1, \neg b_1.$ ~~$a \leftarrow a_1, b_1.$~~ $b.$

Running Example

Initialization		$a \rightarrow b \rightarrow b$		$ab \rightarrow b \rightarrow a$	
2-step program	1-step program	$(a_2 b_1, b)$	(a_1, b)	$(a_2 b_2 b_1, a)$	$(a_1 b_1, b)$
$a.$ $b.$	$a.$ $b.$	$a \leftarrow \neg \mathbf{a_2}.$ $a \leftarrow \mathbf{b_2}.$ $a \leftarrow \mathbf{a_1}.$ $a \leftarrow \neg \mathbf{b_1}.$ $b.$	$a \leftarrow \neg \mathbf{a_1}.$ $a \leftarrow \mathbf{b_1}.$ $b.$ $(a_2 b_1, b)$ $a \leftarrow \neg a_1, \neg \mathbf{b_1}.$ $a \leftarrow \mathbf{a_1}, b_1.$ $b.$	$a \leftarrow \neg a_2.$ $a \leftarrow b_2.$ $a \leftarrow a_1.$ $a \leftarrow \neg b_1.$ $b \leftarrow \neg a_2.$ $b \leftarrow \neg b_2.$ $b \leftarrow a_1.$ $b \leftarrow \neg b_1.$	$a \leftarrow \neg a_1, \neg b_1.$ ~~$a \leftarrow a_1, b_1.$~~ $b.$ $(a_2 b_2 b_1, a)$ $a \leftarrow \neg a_1, \neg b_1.$ $b \leftarrow \mathbf{a_1}.$ $b \leftarrow \neg \mathbf{b_1}.$

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$					
$a \leftarrow \neg a_2$.					
$a \leftarrow b_2$.					
$a \leftarrow \neg a_1$.					
$a \leftarrow \neg b_1$.					
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}$.~~					
~~$b \leftarrow \neg a_2, \mathbf{a_1}$.~~					
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}$.~~					
$b \leftarrow \neg b_2$.					
$b \leftarrow a_1$.					
$b \leftarrow \neg b_1$.					

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)				
$a \leftarrow \neg a_2$.	$a \leftarrow \neg a_1, \neg b_1$.				
$a \leftarrow b_2$.	$b \leftarrow a_1$.				
$a \leftarrow \neg a_1$.	$b \leftarrow \neg b_1$.				
$a \leftarrow \neg b_1$.					
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}$.~~					
~~$b \leftarrow \neg a_2, \mathbf{a_1}$.~~					
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}$.~~					
$b \leftarrow \neg b_2$.					
$b \leftarrow a_1$.					
$b \leftarrow \neg b_1$.					

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)				
$a \leftarrow \neg a_2$.	$a \leftarrow \neg a_1, \neg b_1$.				
$a \leftarrow b_2$.	$b \leftarrow a_1$.				
$a \leftarrow \neg a_1$.	$b \leftarrow \neg b_1$.				
$a \leftarrow \neg b_1$.	$(b_2 b_1, a)$				
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}$.~~	$a \leftarrow \neg a_1, \neg b_1$.				
~~$b \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.				
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}$.~~	$b \leftarrow \neg b_1$.				
$b \leftarrow \neg b_2$.					
$b \leftarrow a_1$.					
$b \leftarrow \neg b_1$.					

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)	(ϵ, ϵ)			
$a \leftarrow \neg a_2.$	$a \leftarrow \neg a_1, \neg b_1.$	~~$a \leftarrow \neg a_2, \mathbf{b_2}.$~~			
$a \leftarrow b_2.$	$b \leftarrow a_1.$	~~$a \leftarrow \neg a_2, \mathbf{a_1}.$~~			
$a \leftarrow \neg a_1.$	$b \leftarrow \neg b_1.$	$a \leftarrow \neg a_2, \mathbf{b_1}.$			
$a \leftarrow \neg b_1.$	$(b_2 b_1, a)$	$a \leftarrow b_2.$			
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}.$~~	$a \leftarrow \neg a_1, \neg b_1.$	$a \leftarrow a_1.$			
~~$b \leftarrow \neg a_2, \mathbf{a_1}.$~~	$b \leftarrow a_1.$	$a \leftarrow \mathbf{a_2}, \neg b_1.$			
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}.$~~	$b \leftarrow \neg b_1.$	~~$a \leftarrow \mathbf{b_2}, \neg b_1.$~~			
$b \leftarrow \neg b_2.$		~~$a \leftarrow \mathbf{a_1}, \neg b_1.$~~			
$b \leftarrow a_1.$		$b \leftarrow \mathbf{a_2}, \neg b_2.$			
$b \leftarrow \neg b_1.$		~~$b \leftarrow \neg b_2, \mathbf{a_1}.$~~			
		$b \leftarrow \neg b_2, \mathbf{b_1}.$			
		$b \leftarrow a_1.$			
		$b \leftarrow \mathbf{a_2}, \neg b_1.$			
		$b \leftarrow \mathbf{b_2}, \neg b_1.$			
		~~$b \leftarrow \mathbf{a_1}, \neg b_1.$~~			

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)	(ϵ, ϵ)	(ϵ, ϵ)		
$a \leftarrow \neg a_2.$	$a \leftarrow \neg a_1, \neg b_1.$	~~$a \leftarrow \neg a_2, \mathbf{b_2}.$~~	~~$a \leftarrow \neg a_1, \neg b_1.$~~		
$a \leftarrow b_2.$	$b \leftarrow a_1.$	~~$a \leftarrow \neg a_2, \mathbf{a_1}.$~~	$b \leftarrow a_1.$		
$a \leftarrow \neg a_1.$	$b \leftarrow \neg b_1.$	$a \leftarrow \neg a_2, \mathbf{b_1}.$	~~$b \leftarrow \mathbf{a_1}, \neg b_1.$~~		
$a \leftarrow \neg b_1.$	$(b_2 b_1, a)$	$a \leftarrow b_2.$			
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}.$~~	$a \leftarrow \neg a_1, \neg b_1.$	$a \leftarrow a_1.$			
~~$b \leftarrow \neg a_2, \mathbf{a_1}.$~~	$b \leftarrow a_1.$	$a \leftarrow \mathbf{a_2}, \neg b_1.$			
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}.$~~	$b \leftarrow \neg b_1.$	~~$a \leftarrow \mathbf{b_2}, \neg b_1.$~~			
$b \leftarrow \neg b_2.$		~~$a \leftarrow \mathbf{a_1}, \neg b_1.$~~			
$b \leftarrow a_1.$		$b \leftarrow \mathbf{a_2}, \neg b_2.$			
$b \leftarrow \neg b_1.$		~~$b \leftarrow \neg b_2, \mathbf{a_1}.$~~			
		$b \leftarrow \neg b_2, \mathbf{b_1}.$			
		$b \leftarrow a_1.$			
		$b \leftarrow \mathbf{a_2}, \neg b_1.$			
		$b \leftarrow \mathbf{b_2}, \neg b_1.$			
		~~$b \leftarrow \mathbf{a_1}, \neg b_1.$~~			

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)	(ϵ, ϵ)	(ϵ, ϵ)		
$a \leftarrow \neg a_2$.	$a \leftarrow \neg a_1, \neg b_1$.	~~$a \leftarrow \neg a_2, \mathbf{b_2}$.~~	~~$a \leftarrow \neg a_1, \neg b_1$.~~		
$a \leftarrow b_2$.	$b \leftarrow a_1$.	~~$a \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.		
$a \leftarrow \neg a_1$.	$b \leftarrow \neg b_1$.	$a \leftarrow \neg a_2, \mathbf{b_1}$.	~~$b \leftarrow \mathbf{a_1}, \neg b_1$.~~		
$a \leftarrow \neg b_1$.	$(b_2 b_1, a)$	$a \leftarrow b_2$.	(ϵ, ϵ)		
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}$.~~	$a \leftarrow \neg a_1, \neg b_1$.	$a \leftarrow a_1$.	$b \leftarrow a_1$.		
~~$b \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.	$a \leftarrow \mathbf{a_2}, \neg b_1$.			
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}$.~~	$b \leftarrow \neg b_1$.	~~$a \leftarrow \mathbf{b_2}, \neg b_1$.~~			
$b \leftarrow \neg b_2$.		~~$a \leftarrow \mathbf{a_1}, \neg b_1$.~~			
$b \leftarrow a_1$.		$b \leftarrow \mathbf{a_2}, \neg b_2$.			
$b \leftarrow \neg b_1$.		~~$b \leftarrow \neg b_2, \mathbf{a_1}$.~~			
		$b \leftarrow \neg b_2, \mathbf{b_1}$.			
		$b \leftarrow a_1$.			
		$b \leftarrow \mathbf{a_2}, \neg b_1$.			
		$b \leftarrow \mathbf{b_2}, \neg b_1$.			
		~~$b \leftarrow \mathbf{a_1}, \neg b_1$.~~			

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)	(ϵ, ϵ)	(ϵ, ϵ)	(b_2, ϵ)	
$a \leftarrow \neg a_2.$	$a \leftarrow \neg a_1, \neg b_1.$	~~$a \leftarrow \neg a_2, \mathbf{b_2}.$~~	~~$a \leftarrow \neg a_1, \neg b_1.$~~	$a \leftarrow \neg a_2, b_1.$	
$a \leftarrow b_2.$	$b \leftarrow a_1.$	~~$a \leftarrow \neg a_2, \mathbf{a_1}.$~~	$b \leftarrow a_1.$	$a \leftarrow \mathbf{a_2}, b_2.$	
$a \leftarrow \neg a_1.$	$b \leftarrow \neg b_1.$	$a \leftarrow \neg a_2, \mathbf{b_1}.$	~~$b \leftarrow \mathbf{a_1}, \neg b_1.$~~	~~$a \leftarrow b_2, \mathbf{a_1}.$~~	
$a \leftarrow \neg b_1.$	$(b_2 b_1, a)$	$a \leftarrow b_2.$	(ϵ, ϵ)	$a \leftarrow b_2, \mathbf{b_1}.$	
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}.$~~	$a \leftarrow \neg a_1, \neg b_1.$	$a \leftarrow a_1.$	$b \leftarrow a_1.$	$a \leftarrow a_1.$	
~~$b \leftarrow \neg a_2, \mathbf{a_1}.$~~	$b \leftarrow a_1.$	$a \leftarrow \mathbf{a_2}, \neg b_1.$		$a \leftarrow \mathbf{a_2}, \neg b_1.$	
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}.$~~	$b \leftarrow \neg b_1.$	~~$a \leftarrow \mathbf{b_2}, \neg b_1.$~~		$b \leftarrow \mathbf{a_2}, \neg b_2.$	
$b \leftarrow \neg b_2.$		~~$a \leftarrow \mathbf{a_1}, \neg b_1.$~~		$b \leftarrow \neg b_2, b_1.$	
$b \leftarrow a_1.$		$b \leftarrow \mathbf{a_2}, \neg b_2.$		$b \leftarrow a_1.$	
$b \leftarrow \neg b_1.$		~~$b \leftarrow \neg b_2, \mathbf{a_1}.$~~		$b \leftarrow a_2, \neg b_1.$	
		$b \leftarrow \neg b_2, \mathbf{b_1}.$		~~$b \leftarrow \mathbf{a_2}, b_2, \neg b_1.$~~	
		$b \leftarrow a_1.$		~~$b \leftarrow b_2, \neg b_1, \mathbf{a_1}.$~~	
		$b \leftarrow \mathbf{a_2}, \neg b_1.$			
		$b \leftarrow \mathbf{b_2}, \neg b_1.$			
		~~$b \leftarrow \mathbf{a_1}, \neg b_1.$~~			

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)	(ϵ, ϵ)	(ϵ, ϵ)	(b_2, ϵ)	(b_1, ϵ)
$a \leftarrow \neg a_2$.	$a \leftarrow \neg a_1, \neg b_1$.	~~$a \leftarrow \neg a_2, \mathbf{b_2}$.~~	~~$a \leftarrow \neg a_1, \neg b_1$.~~	$a \leftarrow \neg a_2, b_1$.	$b \leftarrow a_1$.
$a \leftarrow b_2$.	$b \leftarrow a_1$.	~~$a \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.	$a \leftarrow \mathbf{a_2}, b_2$.	
$a \leftarrow \neg a_1$.	$b \leftarrow \neg b_1$.	$a \leftarrow \neg a_2, \mathbf{b_1}$.	~~$b \leftarrow \mathbf{a_1}, \neg b_1$.~~	~~$a \leftarrow b_2, \mathbf{a_1}$.~~	
$a \leftarrow \neg b_1$.	$(b_2 b_1, a)$	$a \leftarrow b_2$.	(ϵ, ϵ)	$a \leftarrow b_2, \mathbf{b_1}$.	
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}$.~~	$a \leftarrow \neg a_1, \neg b_1$.	$a \leftarrow a_1$.	$b \leftarrow a_1$.	$a \leftarrow a_1$.	
~~$b \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.	$a \leftarrow \mathbf{a_2}, \neg b_1$.		$a \leftarrow \mathbf{a_2}, \neg b_1$.	
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}$.~~	$b \leftarrow \neg b_1$.	~~$a \leftarrow \mathbf{b_2}, \neg b_1$.~~		$b \leftarrow \mathbf{a_2}, \neg b_2$.	
$b \leftarrow \neg b_2$.		~~$a \leftarrow \mathbf{a_1}, \neg b_1$.~~		$b \leftarrow \neg b_2, b_1$.	
$b \leftarrow a_1$.		$b \leftarrow \mathbf{a_2}, \neg b_2$.		$b \leftarrow a_1$.	
$b \leftarrow \neg b_1$.		~~$b \leftarrow \neg b_2, \mathbf{a_1}$.~~		$b \leftarrow a_2, \neg b_1$.	
		$b \leftarrow \neg b_2, \mathbf{b_1}$.		~~$b \leftarrow \mathbf{a_2}, b_2, \neg b_1$.~~	
		$b \leftarrow a_1$.		~~$b \leftarrow b_2, \neg b_1, \mathbf{a_1}$.~~	
		$b \leftarrow \mathbf{a_2}, \neg b_1$.			
		$b \leftarrow \mathbf{b_2}, \neg b_1$.			
		~~$b \leftarrow \mathbf{a_1}, \neg b_1$.~~			

Running Example

$b \to b \to a$		$\epsilon \to \epsilon \to \epsilon$		$b \to \epsilon \to \epsilon$	
$(b_2 b_1, a)$	(b_1, b)	(ϵ, ϵ)	(ϵ, ϵ)	(b_2, ϵ)	(b_1, ϵ)
$a \leftarrow \neg a_2$.	$a \leftarrow \neg a_1, \neg b_1$.	~~$a \leftarrow \neg a_2, \mathbf{b_2}$.~~	~~$a \leftarrow \neg a_1, \neg b_1$.~~	$a \leftarrow \neg a_2, b_1$.	$b \leftarrow a_1$.
$a \leftarrow b_2$.	$b \leftarrow a_1$.	~~$a \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.	$a \leftarrow \mathbf{a_2}, b_2$.	
$a \leftarrow \neg a_1$.	$b \leftarrow \neg b_1$.	$a \leftarrow \neg a_2, \mathbf{b_1}$.	~~$b \leftarrow \mathbf{a_1}, \neg b_1$.~~	~~$a \leftarrow b_2, \mathbf{a_1}$.~~	
$a \leftarrow \neg b_1$.	$(b_2 b_1, a)$	$a \leftarrow b_2$.	(ϵ, ϵ)	$a \leftarrow b_2, \mathbf{b_1}$.	$(b_2 \epsilon, \epsilon)$
~~$b \leftarrow \neg a_2, \neg \mathbf{b_2}$.~~	$a \leftarrow \neg a_1, \neg b_1$.	$a \leftarrow a_1$.	$b \leftarrow a_1$.	$a \leftarrow a_1$.	$b \leftarrow a_1$.
~~$b \leftarrow \neg a_2, \mathbf{a_1}$.~~	$b \leftarrow a_1$.	$a \leftarrow \mathbf{a_2}, \neg b_1$.		$a \leftarrow \mathbf{a_2}, \neg b_1$.	
~~$b \leftarrow \neg a_2, \neg \mathbf{b_1}$.~~	$b \leftarrow \neg b_1$.	~~$a \leftarrow \mathbf{b_2}, \neg b_1$.~~		$b \leftarrow \mathbf{a_2}, \neg b_2$.	
$b \leftarrow \neg b_2$.		~~$a \leftarrow \mathbf{a_1}, \neg b_1$.~~		$b \leftarrow \neg b_2, b_1$.	
$b \leftarrow a_1$.		$b \leftarrow \mathbf{a_2}, \neg b_2$.		$b \leftarrow a_1$.	
$b \leftarrow \neg b_1$.		~~$b \leftarrow \neg b_2, \mathbf{a_1}$.~~		$b \leftarrow a_2, \neg b_1$.	
		$b \leftarrow \neg b_2, \mathbf{b_1}$.		~~$b \leftarrow \mathbf{a_2}, b_2, \neg b_1$.~~	
		$b \leftarrow a_1$.		~~$b \leftarrow b_2, \neg b_1, \mathbf{a_1}$.~~	
		$b \leftarrow \mathbf{a_2}, \neg b_1$.			
		$b \leftarrow \mathbf{b_2}, \neg b_1$.			
		~~$b \leftarrow \mathbf{a_1}, \neg b_1$.~~			

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$					
$a \leftarrow \neg a_2, b_1.$					
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~					
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~					
$a \leftarrow b_2, b_1.$					
$a \leftarrow a_1.$					
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$					
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					
$b \leftarrow \mathbf{a_2}, \neg b_2.$					
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$				
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$				
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~					
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~					
$a \leftarrow b_2, b_1.$					
$a \leftarrow a_1.$					
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$					
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					
$b \leftarrow \mathbf{a_2}, \neg b_2.$					
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$				
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$				
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~					
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~	(ϵ, ϵ)				
$a \leftarrow b_2, b_1.$	$b \leftarrow a_1, \neg b_1.$				
$a \leftarrow a_1.$					
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$					
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					
$b \leftarrow \mathbf{a_2}, \neg b_2.$					
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$	(a_2, ϵ)			
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$	$a \leftarrow \neg a_2, b_1.$			
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~		$a \leftarrow b_2, b_1.$			
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~	(ϵ, ϵ)	$a \leftarrow a_1.$			
$a \leftarrow b_2, b_1.$	$b \leftarrow a_1, \neg b_1.$	~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~			
$a \leftarrow a_1.$		$b \leftarrow a_2, \neg b_2.$			
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$		$b \leftarrow \neg b_2, b_1.$			
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~		$b \leftarrow a_1.$			
$b \leftarrow \mathbf{a_2}, \neg b_2.$					
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$	(a_2, ϵ)	(a_1, ϵ)		
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$	$a \leftarrow \neg a_2, b_1.$	~~$b \leftarrow a_1, \neg b_1.$~~		
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~		$a \leftarrow b_2, b_1.$			
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~	(ϵ, ϵ)	$a \leftarrow a_1.$			
$a \leftarrow b_2, b_1.$	$b \leftarrow a_1, \neg b_1.$	~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~			
$a \leftarrow a_1.$		$b \leftarrow a_2, \neg b_2.$			
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$		$b \leftarrow \neg b_2, b_1.$			
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~		$b \leftarrow a_1.$			
$b \leftarrow \mathbf{a_2}, \neg b_2.$					
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$	(a_2, ϵ)	(a_1, ϵ)		
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$	$a \leftarrow \neg a_2, b_1.$	~~$b \leftarrow a_1, \neg b_1.$~~		
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~		$a \leftarrow b_2, b_1.$			
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~	(ϵ, ϵ)	$a \leftarrow a_1.$	(ϵ, b)		
$a \leftarrow b_2, b_1.$	$b \leftarrow a_1, \neg b_1.$	~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~	$\emptyset$		
$a \leftarrow a_1.$		$b \leftarrow a_2, \neg b_2.$			
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$		$b \leftarrow \neg b_2, b_1.$			
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~		$b \leftarrow a_1.$			
$b \leftarrow \mathbf{a_2}, \neg b_2.$					
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$	(a_2, ϵ)	(a_1, ϵ)	(b_1, ϵ)	
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$	$a \leftarrow \neg a_2, b_1.$	~~$b \leftarrow a_1, \neg b_1.$~~	~~$a \leftarrow \neg a_2, \mathbf{b_2} b_1.$~~	
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~		$a \leftarrow b_2, b_1.$		~~$a \leftarrow \neg a_2, \mathbf{a_1} b_1.$~~	
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~	(ϵ, ϵ)	$a \leftarrow a_1.$	(ϵ, b)	$a \leftarrow b_2, b_1.$	
$a \leftarrow b_2, b_1.$	$b \leftarrow a_1, \neg b_1.$	~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~	$\emptyset$	$a \leftarrow a_1.$	
$a \leftarrow a_1.$		$b \leftarrow a_2, \neg b_2.$		$b \leftarrow a_2, \neg b_2.$	
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$		$b \leftarrow \neg b_2, b_1.$		~~$b \leftarrow \mathbf{a_2}, \neg b_2, b_1.$~~	
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~		$b \leftarrow a_1.$		~~$b \leftarrow \neg b_2, \mathbf{a_1}, b_1.$~~	
$b \leftarrow \mathbf{a_2}, \neg b_2.$				$b \leftarrow a_1.$	
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$		$a \to \epsilon \to b$		$\epsilon \to b \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$	(a_2, ϵ)	(a_1, ϵ)	(b_1, ϵ)	(ϵ, b)
$a \leftarrow \neg a_2, b_1.$	$b \leftarrow a_1, \neg \mathbf{b_1}.$	$a \leftarrow \neg a_2, b_1.$	~~$b \leftarrow a_1, \neg b_1.$~~	~~$a \leftarrow \neg a_2, \mathbf{b_2} b_1.$~~	$\emptyset$
~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~		$a \leftarrow b_2, b_1.$		~~$a \leftarrow \neg a_2, \mathbf{a_1} b_1.$~~	
~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~	(ϵ, ϵ)	$a \leftarrow a_1.$	(ϵ, b)	$a \leftarrow b_2, b_1.$	
$a \leftarrow b_2, b_1.$	$b \leftarrow a_1, \neg b_1.$	~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~	$\emptyset$	$a \leftarrow a_1.$	
$a \leftarrow a_1.$		$b \leftarrow a_2, \neg b_2.$		$b \leftarrow a_2, \neg b_2.$	
$a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$		$b \leftarrow \neg b_2, b_1.$		~~$b \leftarrow \mathbf{a_2}, \neg b_2, b_1.$~~	
~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~		$b \leftarrow a_1.$		~~$b \leftarrow \neg b_2, \mathbf{a_1}, b_1.$~~	
$b \leftarrow \mathbf{a_2}, \neg b_2.$				$b \leftarrow a_1.$	
$b \leftarrow \neg b_2, b_1.$					
$b \leftarrow a_1.$					
~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~					
~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~					

Running Example

$ab \to \epsilon \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$
$a \leftarrow \neg a_2, b_1.$ ~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~ ~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ $a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$ ~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~ $b \leftarrow \mathbf{a_2}, \neg b_2.$ $b \leftarrow \neg b_2, b_1.$ $b \leftarrow a_1.$ ~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~ ~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~	$b \leftarrow a_1, \neg \mathbf{b_1}.$ (ϵ, ϵ) $b \leftarrow a_1, \neg b_1.$

$a \to \epsilon \to b$	
(a_2, ϵ)	(a_1, ϵ)
$a \leftarrow \neg a_2, b_1.$ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ ~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~ $b \leftarrow a_2, \neg b_2.$ $b \leftarrow \neg b_2, b_1.$ $b \leftarrow a_1.$	~~$b \leftarrow a_1, \neg b_1.$~~ (ϵ, b) $\emptyset$

$\epsilon \to b \to \epsilon$	
(b_1, ϵ)	(ϵ, b)
~~$a \leftarrow \neg a_2, \mathbf{b_2} b_1.$~~ ~~$a \leftarrow \neg a_2, \mathbf{a_1} b_1.$~~ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ $b \leftarrow a_2, \neg b_2.$ ~~$b \leftarrow \mathbf{a_2}, \neg b_2, b_1.$~~ ~~$b \leftarrow \neg b_2, \mathbf{a_1}, b_1.$~~ $b \leftarrow a_1.$	$\emptyset$ (b_1, ϵ) $\emptyset$

Running Example

$ab \to \epsilon \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$
$a \leftarrow \neg a_2, b_1.$ ~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~ ~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ $a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$ ~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~ $b \leftarrow \mathbf{a_2}, \neg b_2.$ $b \leftarrow \neg b_2, b_1.$ $b \leftarrow a_1.$ ~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~ ~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~	$b \leftarrow a_1, \neg \mathbf{b_1}.$ (ϵ, ϵ) $b \leftarrow a_1, \neg b_1.$

$a \to \epsilon \to b$	
(a_2, ϵ)	(a_1, ϵ)
$a \leftarrow \neg a_2, b_1.$ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ ~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~ $b \leftarrow a_2, \neg b_2.$ $b \leftarrow \neg b_2, b_1.$ $b \leftarrow a_1.$	~~$b \leftarrow a_1, \neg b_1.$~~ (ϵ, b) $\emptyset$

$\epsilon \to b \to \epsilon$	
(b_1, ϵ)	(ϵ, b)
~~$a \leftarrow \neg a_2, \mathbf{b_2} b_1.$~~ ~~$a \leftarrow \neg a_2, \mathbf{a_1} b_1.$~~ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ $b \leftarrow a_2, \neg b_2.$ ~~$b \leftarrow \mathbf{a_2}, \neg b_2, b_1.$~~ ~~$b \leftarrow \neg b_2, \mathbf{a_1}, b_1.$~~ $b \leftarrow a_1.$	$\emptyset$ (b_1, ϵ) $\emptyset$

Merging of the programs	OUTPUT
$a \leftarrow b_2, b_1.$ ~~$a \leftarrow a_1.$~~ $b \leftarrow a_2, \neg b_2.$ ~~$b \leftarrow a_1.$~~	

Running Example

$ab \to \epsilon \to \epsilon$	
$(a_2 b_2, \epsilon)$	$(a_1 b_1, \epsilon)$
$a \leftarrow \neg a_2, b_1.$ ~~$a \leftarrow a_2, b_2, \mathbf{a_1}.$~~ ~~$a \leftarrow a_2, b_2, \mathbf{b_1}.$~~ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ $a \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$ ~~$a \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~ $b \leftarrow \mathbf{a_2}, \neg b_2.$ $b \leftarrow \neg b_2, b_1.$ $b \leftarrow a_1.$ ~~$b \leftarrow a_2, \neg \mathbf{b_2}, \neg b_1.$~~ ~~$b \leftarrow a_2, \mathbf{a_1}, \neg b_1.$~~	$b \leftarrow a_1, \neg \mathbf{b_1}.$ (ϵ, ϵ) $b \leftarrow a_1, \neg b_1.$

$a \to \epsilon \to b$	
(a_2, ϵ)	(a_1, ϵ)
$a \leftarrow \neg a_2, b_1.$ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ ~~$a \leftarrow a_2, \neg b_2, \mathbf{a_1} \neg b_1.$~~ $b \leftarrow a_2, \neg b_2.$ $b \leftarrow \neg b_2, b_1.$ $b \leftarrow a_1.$	~~$b \leftarrow a_1, \neg b_1.$~~ (ϵ, b) $\emptyset$

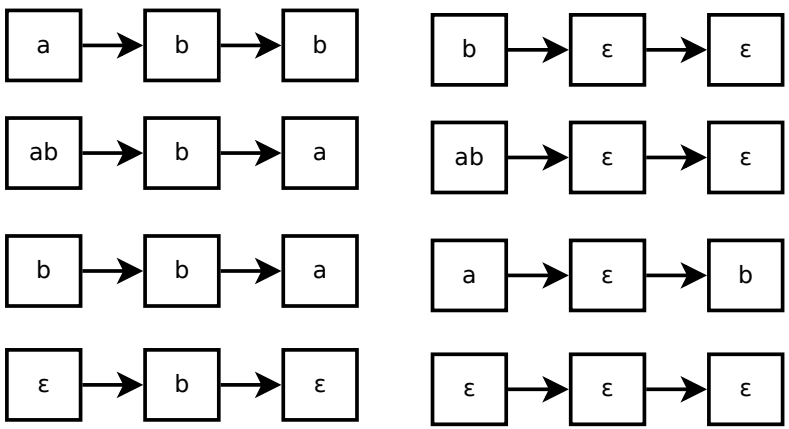
$\epsilon \to b \to \epsilon$	
(b_1, ϵ)	(ϵ, b)
~~$a \leftarrow \neg a_2, \mathbf{b_2} b_1.$~~ ~~$a \leftarrow \neg a_2, \mathbf{a_1} b_1.$~~ $a \leftarrow b_2, b_1.$ $a \leftarrow a_1.$ $b \leftarrow a_2, \neg b_2.$ ~~$b \leftarrow \mathbf{a_2}, \neg b_2, b_1.$~~ ~~$b \leftarrow \neg b_2, \mathbf{a_1}, b_1.$~~ $b \leftarrow a_1.$	$\emptyset$ (b_1, ϵ) $\emptyset$

Merging of the programs	OUTPUT
$a \leftarrow b_2, b_1.$ ~~$a \leftarrow a_1.$~~ $b \leftarrow a_2, \neg b_2.$ ~~$b \leftarrow a_1.$~~	$a \leftarrow b_2, b_1.$ $b \leftarrow a_2, \neg b_2.$

How to know the Delay?

Let S be a Markov(k) system as follows:

$$S = \{(a \leftarrow b_{t-1}, b_{t-2}), (b \leftarrow a_{t-2}, \neg b_{t-2})\}$$

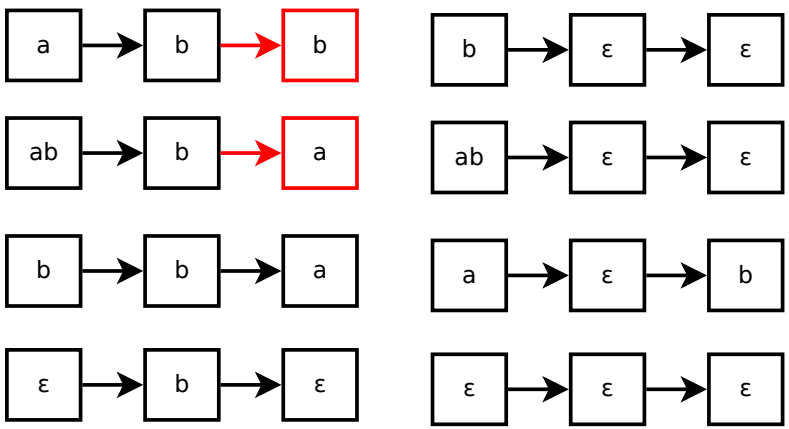


Eight traces of executions of the system S

How to know the Delay?

Let S be a Markov(k) system as follows:

$$S = \{(a \leftarrow b_{t-1}, b_{t-2}), (b \leftarrow a_{t-2}, \neg b_{t-2})\}$$

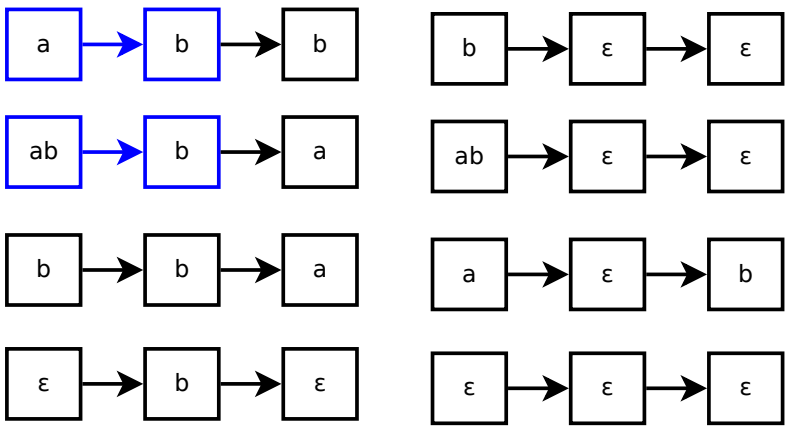


Eight traces of executions of the system S

How to know the Delay?

Let S be a Markov(k) system as follows:

$$S = \{(a \leftarrow b_{t-1}, b_{t-2}), (b \leftarrow a_{t-2}, \neg b_{t-2})\}$$

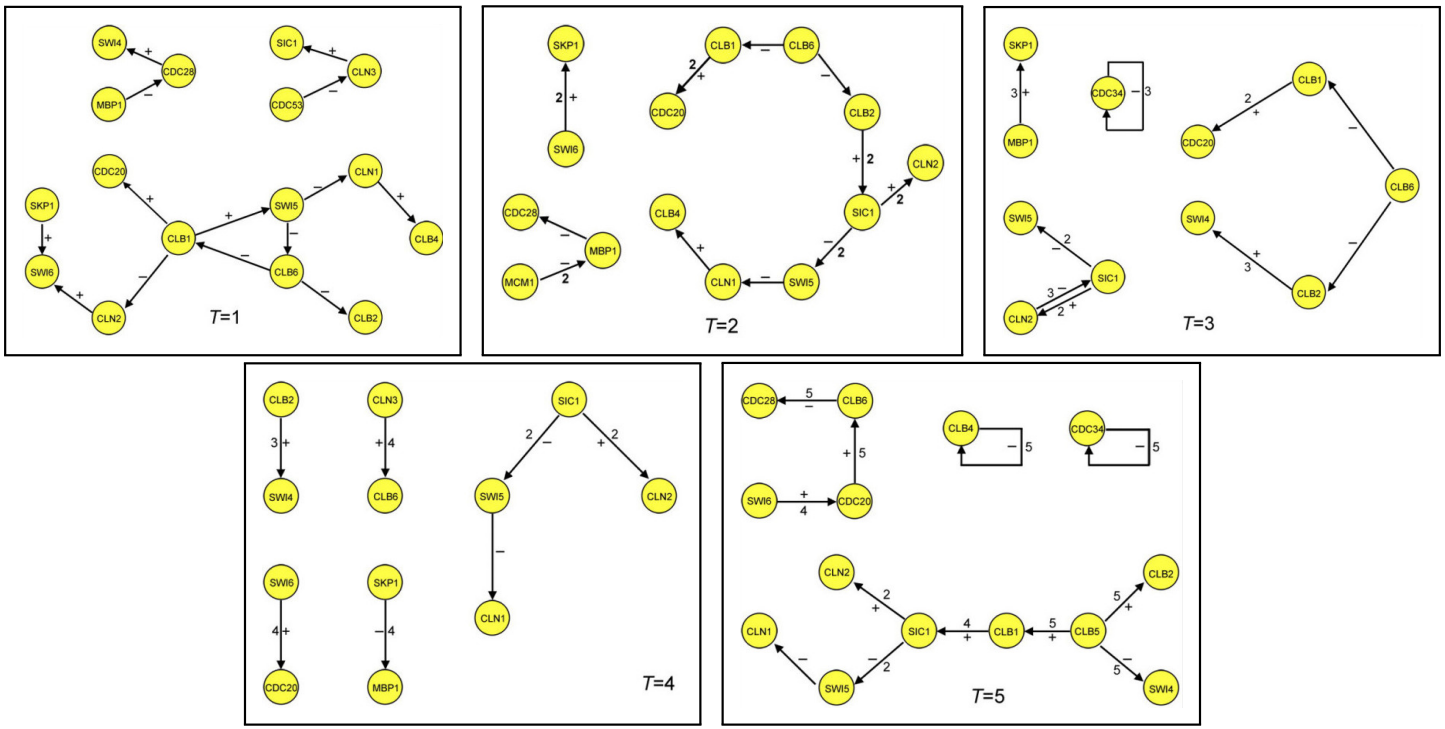


Eight traces of executions of the system S

Experiments

We evaluated LFkT on learning 5 real biological benchmarks.

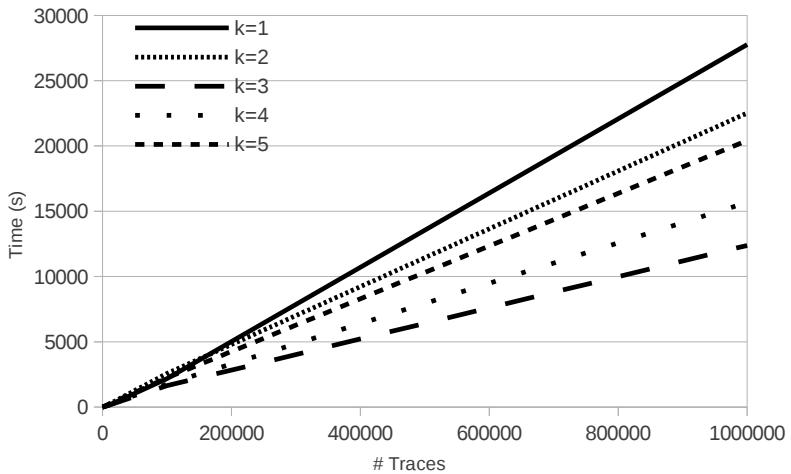
- Benchmark are Markov(1) to Markov(5) systems
- LFkT input goes from 10 to 1 million traces



The time-delayed gene regulatory networks of the human HeLa cell cycling (Li et. al 2006).

Evaluation

Run time (# of seconds)						
# Traces	k=1	k=2	k=3	k=4	k=5	k=7
10	0.23s	0.27s	**0.16s**	0.28s	0.31s	2.8s
100	1.87s	2.49s	**1.63s**	2.37s	2.84s	27s
1,000	15.3s	18.5s	**13.3s**	20.1s	23.8s	249s
10,000	**146s**	218s	147s	201s	234s	3,621s
100,000	2,177s	2,577s	**1,643s**	1,764s	2,243s	39,973s
1,000,000	27,768s	22,517s	12,384s	15,670s	20,413s	327,480s



LFkT Run time varying the input size (number of traces)

Evaluation

Runtime and output size of **LFkT** on learning the benchmark from (*Dubrova and Teslenko 2011*), with regard to different delays.

Runtime (in seconds)/Output size (# of rules)				
Delay	Mammalian (10)	Fission (10)	Budding (12)	Arabidopsis (15)
k=1	0.06s / 23	0.07s / 24	0.42s / 54	5.87s / 28
k=2	6.92s / 3,922	2.5s / 2,003	213s / 10,551	6,902s / 4,981
k=3	752s / 54,692	49.39s / 11,312	20,970s / 60,568	T.O.
k=4	12,448s / 293,020	302s / 32,581	T.O.	T.O.
k=5	T.O.	1,197s / 70,322	T.O.	T.O.
k=6	T.O.	3,585s / 129,603	T.O.	T.O.
k=7	T.O.	9,401s / 216,212	T.O.	T.O.

The complexity of learning a markov(k) system S from a set of observations O with **LFkT** is respectively:
$O(2^{nk})$ for memory and $O(\sum_{T \in O} |T| \cdot n2^{nk})$ for runtime.

Evaluation

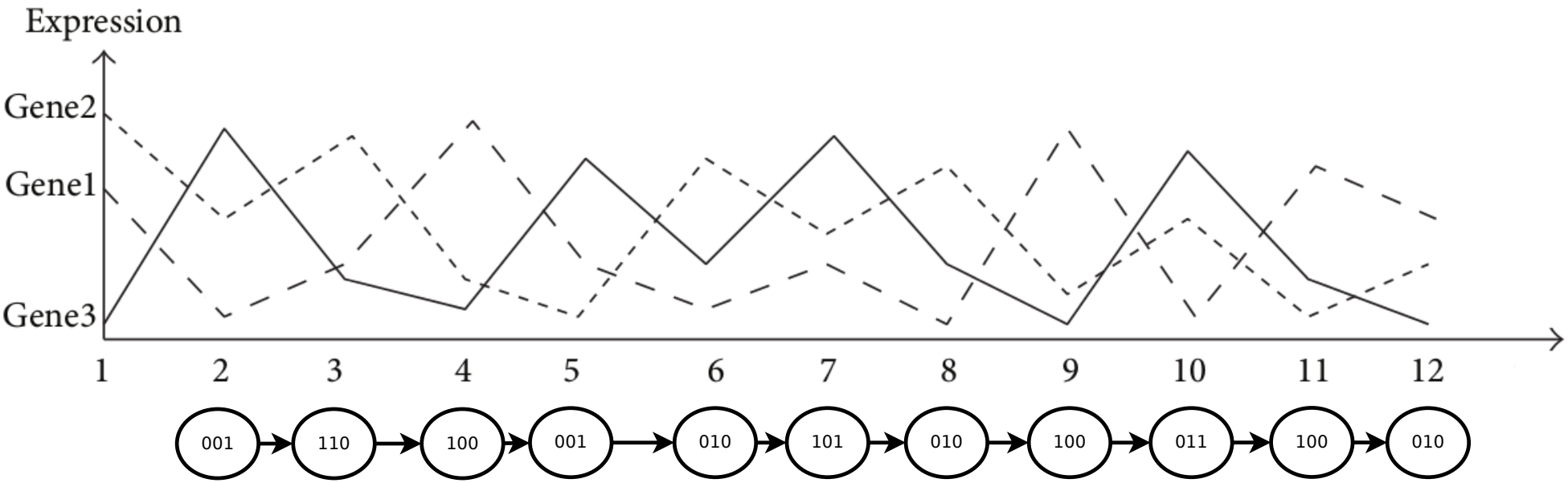
Runtime and output size of **LFkT** on learning the benchmark from (*Dubrova and Teslenko 2011*), with regard to different delays.

Runtime (in seconds)/Output size (# of rules)				
Delay	Mammalian (10)	Fission (10)	Budding (12)	Arabidopsis (15)
k=1	0.06s / 23	0.07s / 24	0.42s / 54	5.87s / 28
k=2	6.92s / 3,922	2.5s / 2,003	213s / 10,551	6,902s / 4,981
k=3	752s / 54,692	49.39s / 11,312	20,970s / 60,568	T.O.
k=4	12,448s / 293,020	302s / 32,581	T.O.	T.O.
k=5	T.O.	1,197s / 70,322	T.O.	T.O.
k=6	T.O.	3,585s / 129,603	T.O.	T.O.
k=7	T.O.	9,401s / 216,212	T.O.	T.O.

The complexity of learning a markov(k) system S from a set of observations O with **LFkT** is respectively:
$O(2^{nk})$ for memory and $O(\sum_{T \in O} |T| \cdot n2^{nk})$ for runtime.

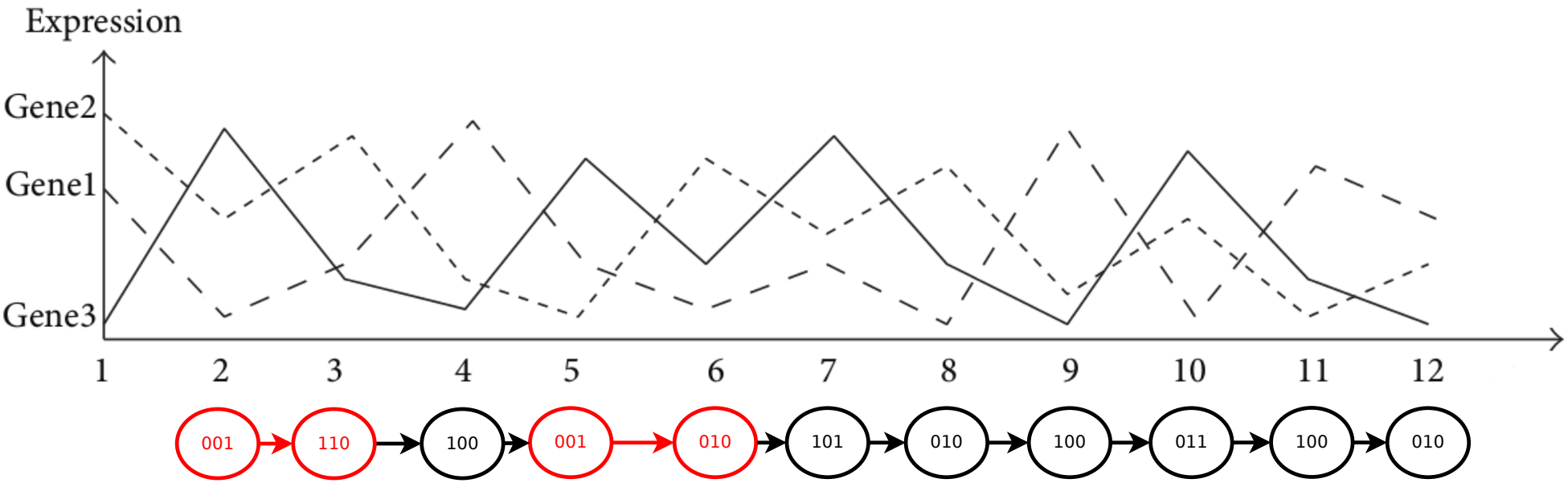
Construction of Temporal Boolean networks with LFkT

Represent the time series as a sequence of state transitions.



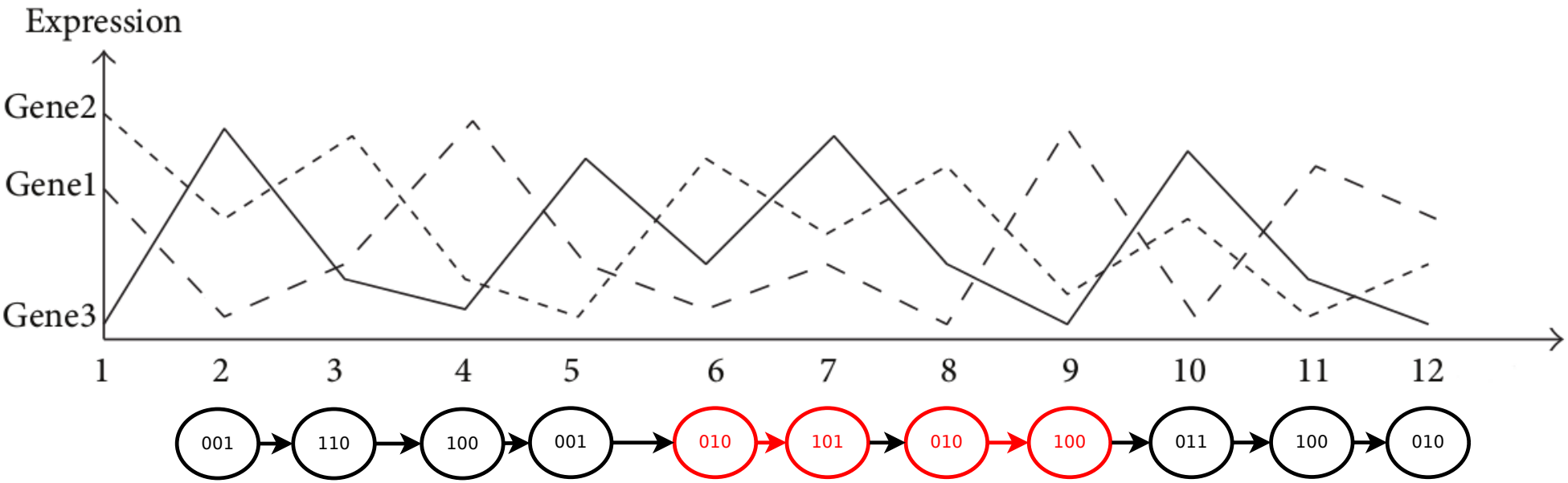
Construction of Temporal Boolean networks with LFkT

The algorithm explains inconsistency by using delays.



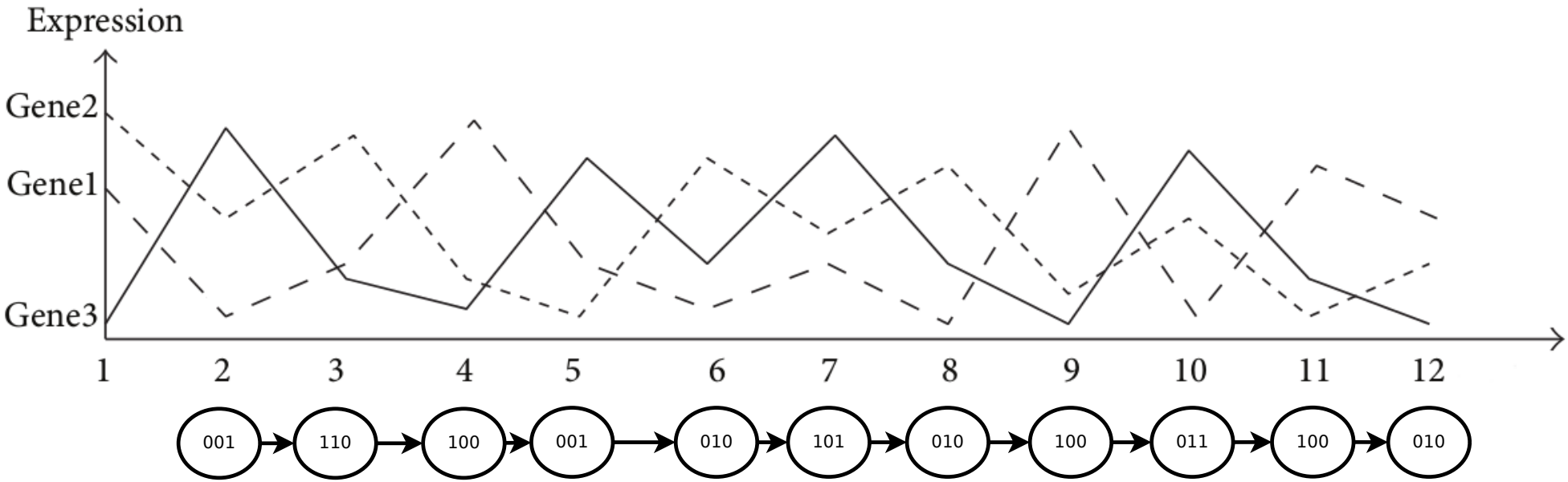
Construction of Temporal Boolean networks with LFkT

The algorithm explains inconsistency by using delays.

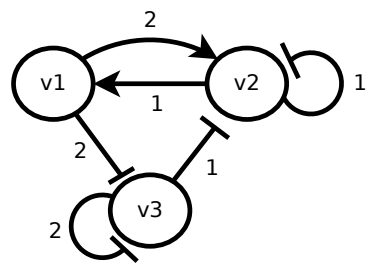


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

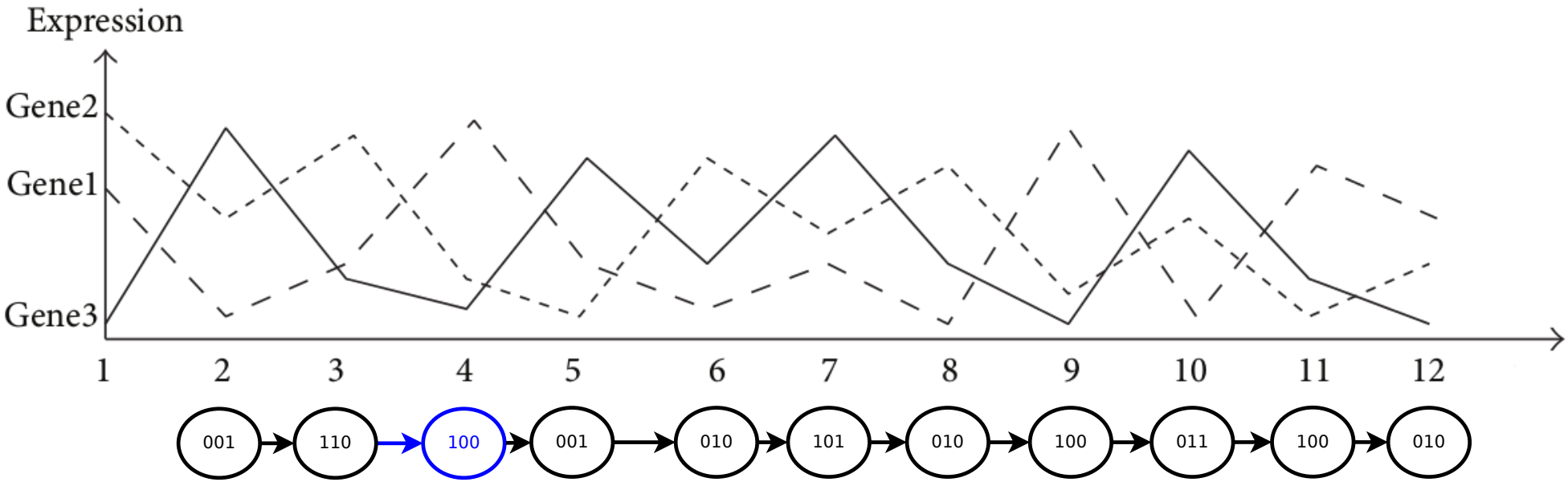



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

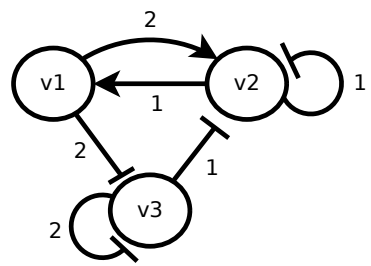


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

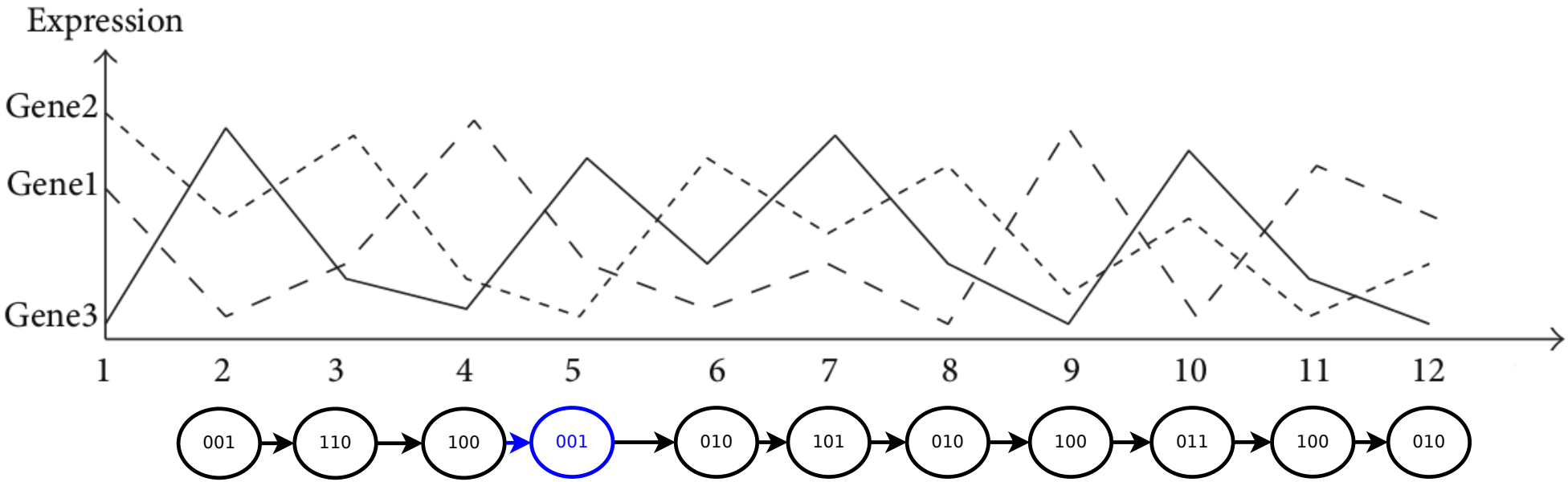



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

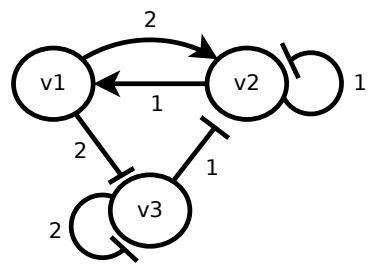


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

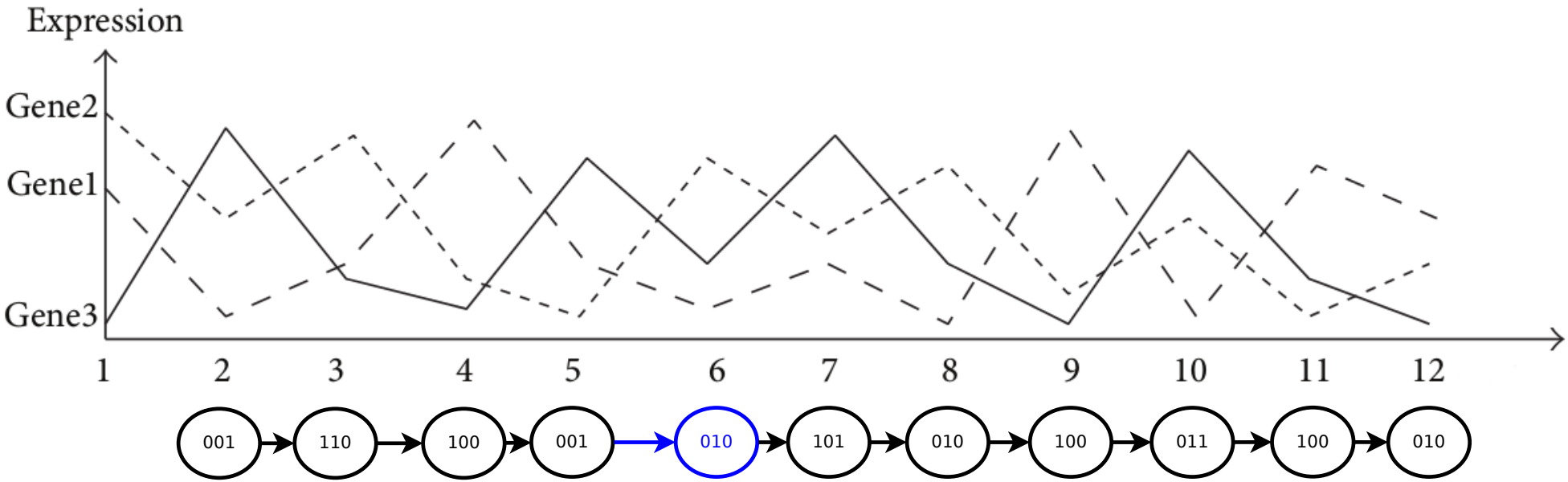



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

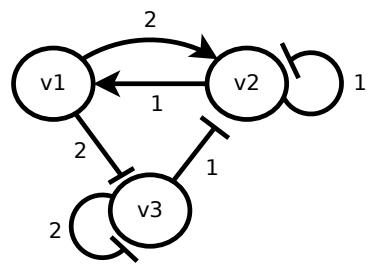


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

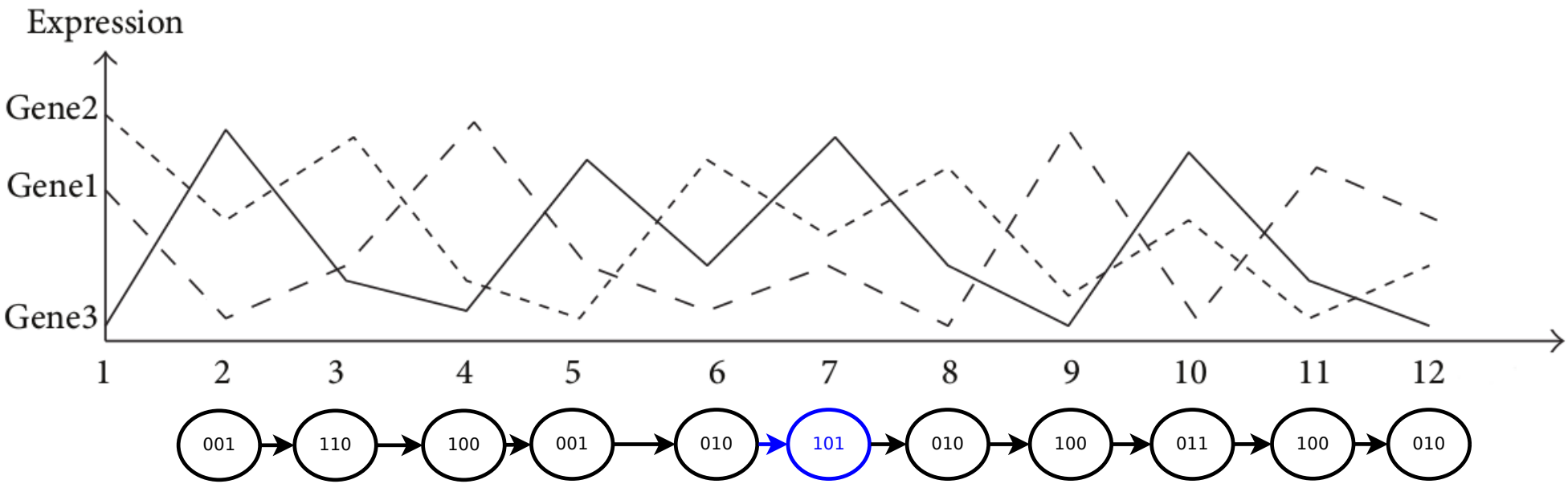



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

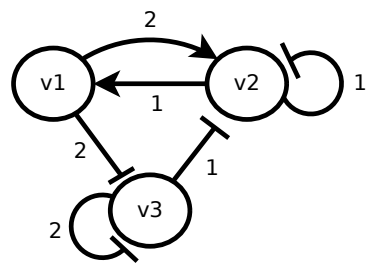


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

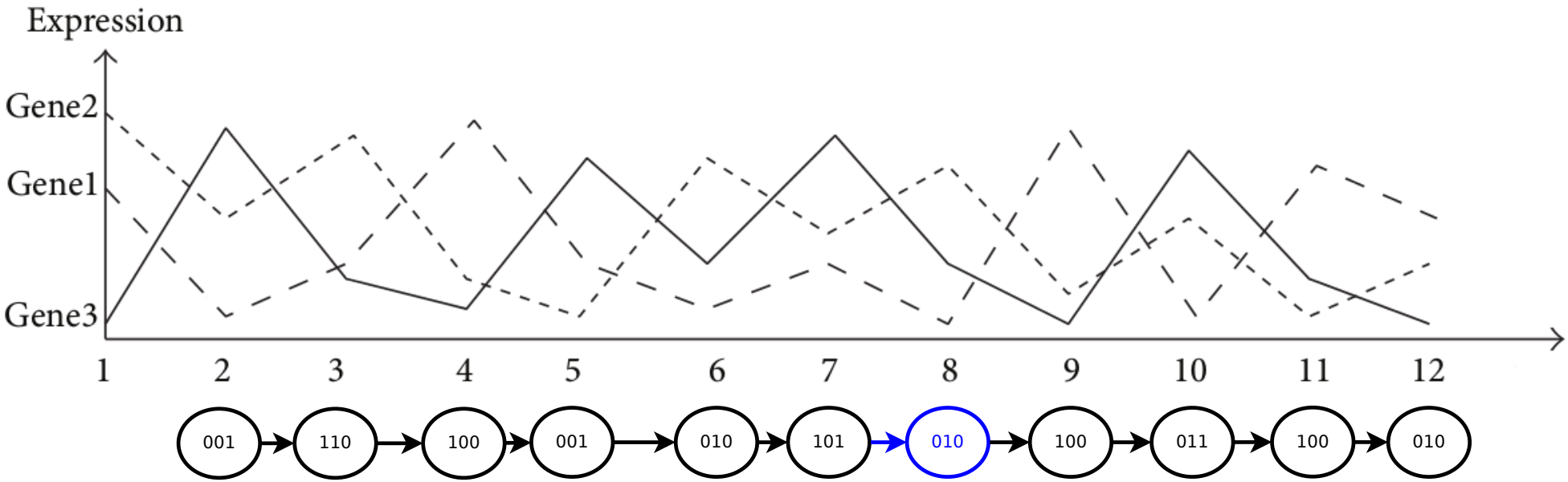



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

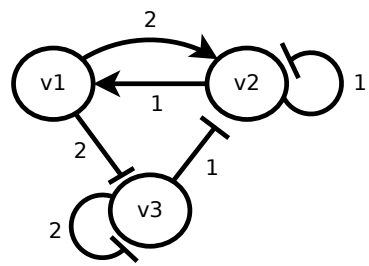


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

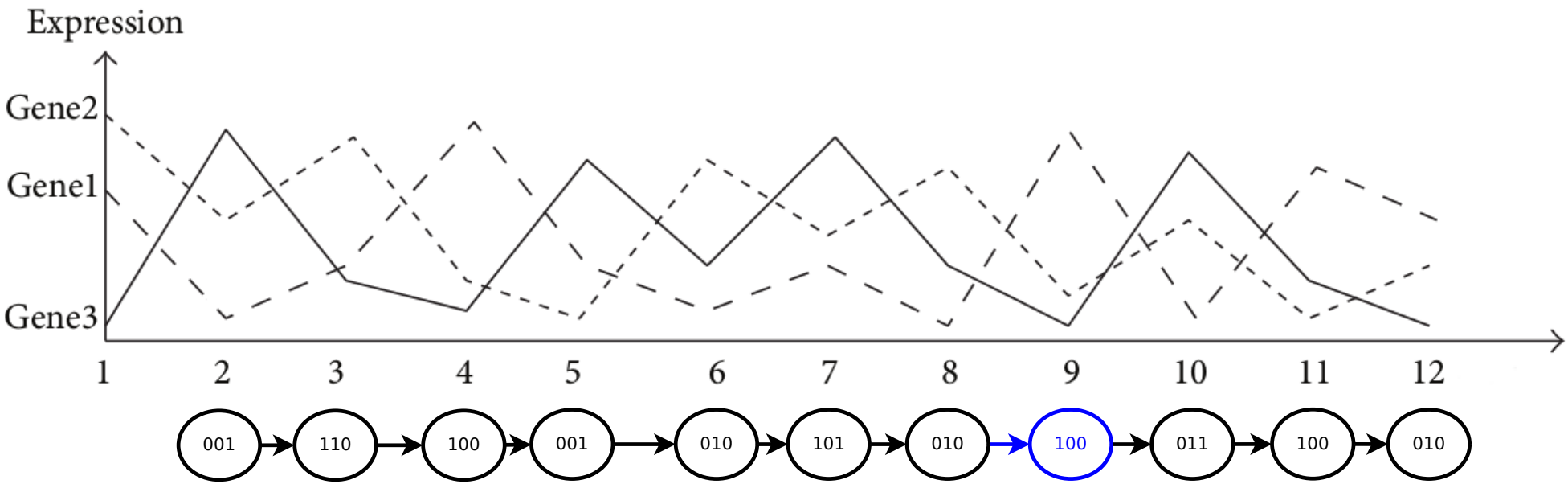



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

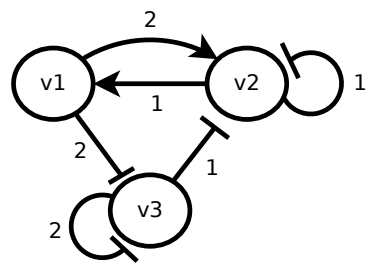


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

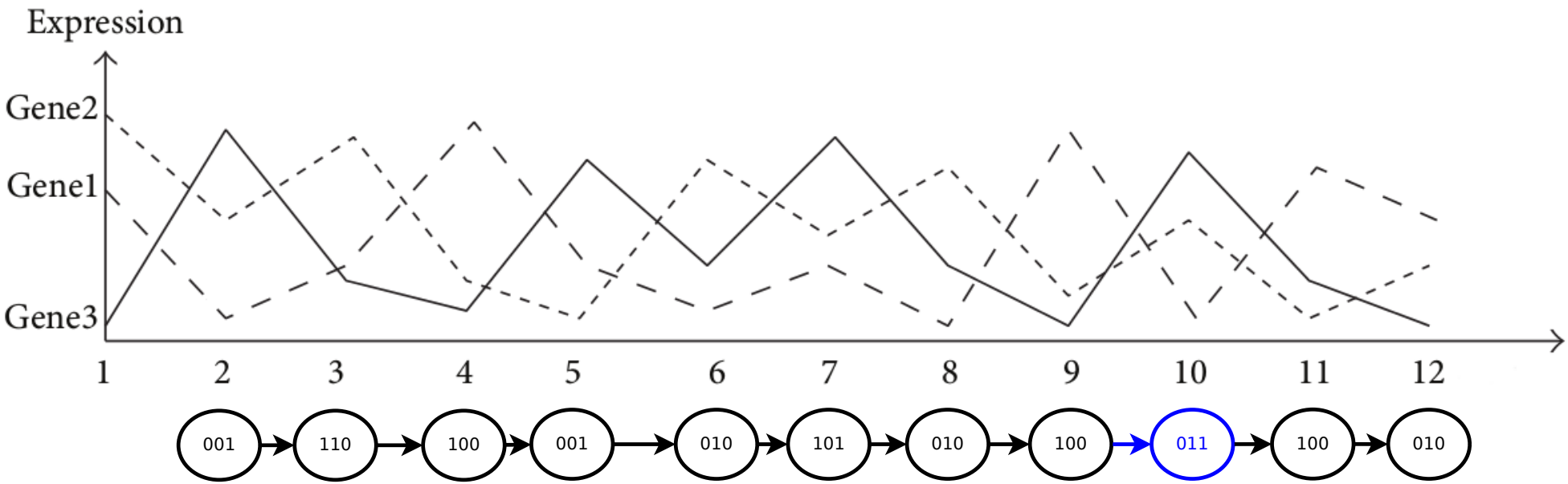



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

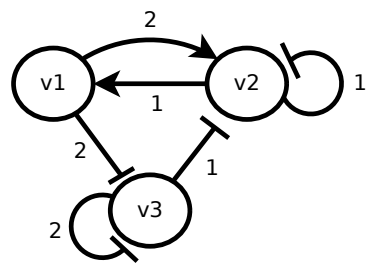


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

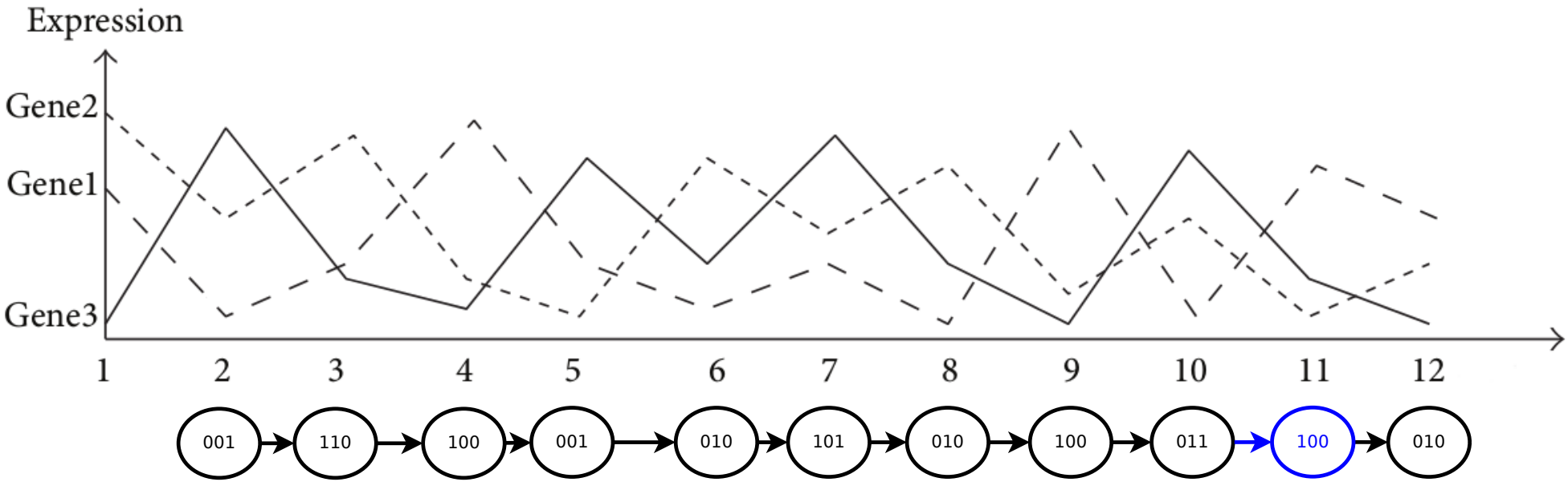



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

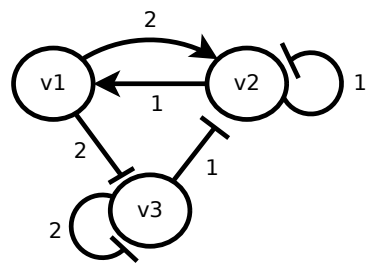


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

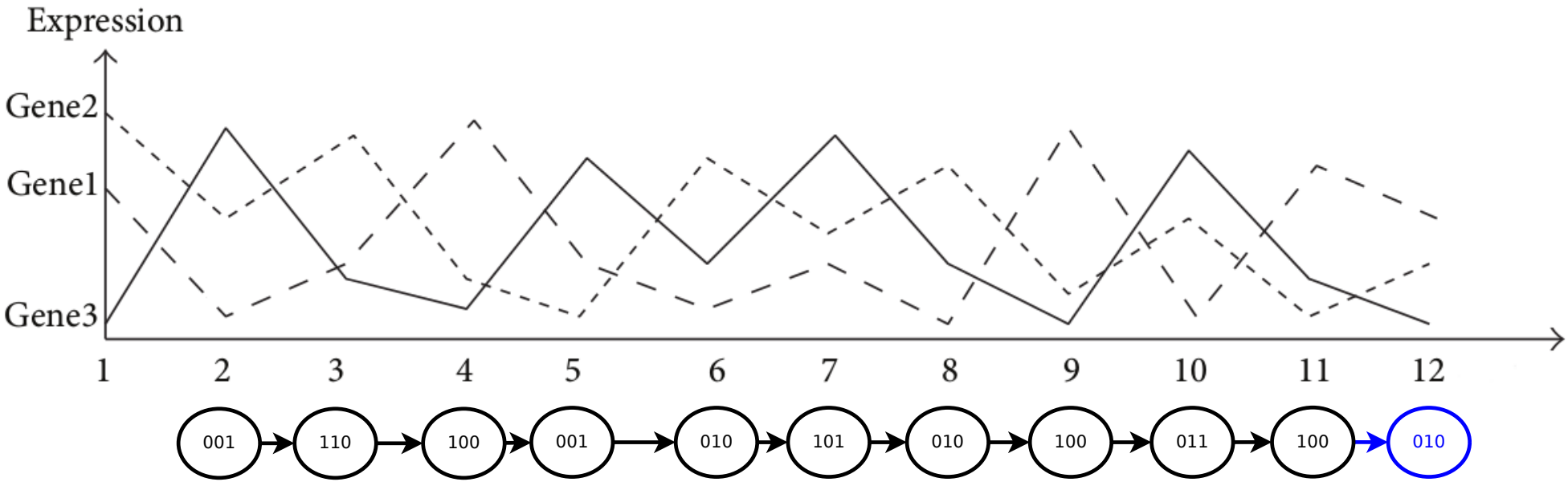



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

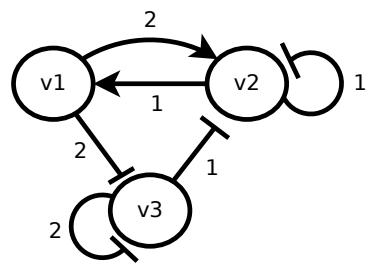


Construction of Temporal Boolean networks with LFkT

Construct one logic program with LFkT that explains the entire trace and compute the corresponding Timed Boolean network.

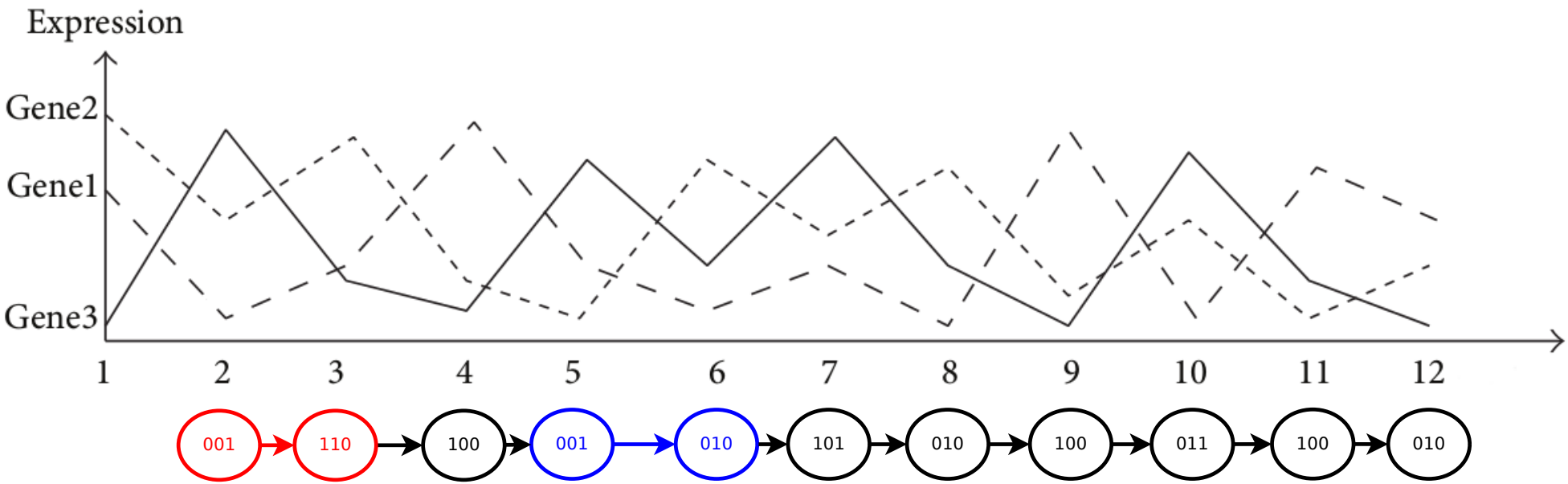



```
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).
```

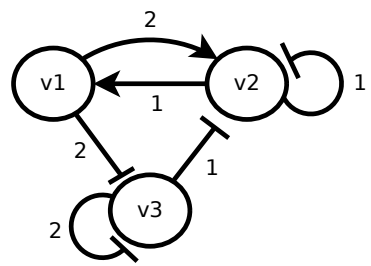


Construction of Temporal Boolean networks with LFkT

But the beginning of the trace could not be captured by the model because of the delay.



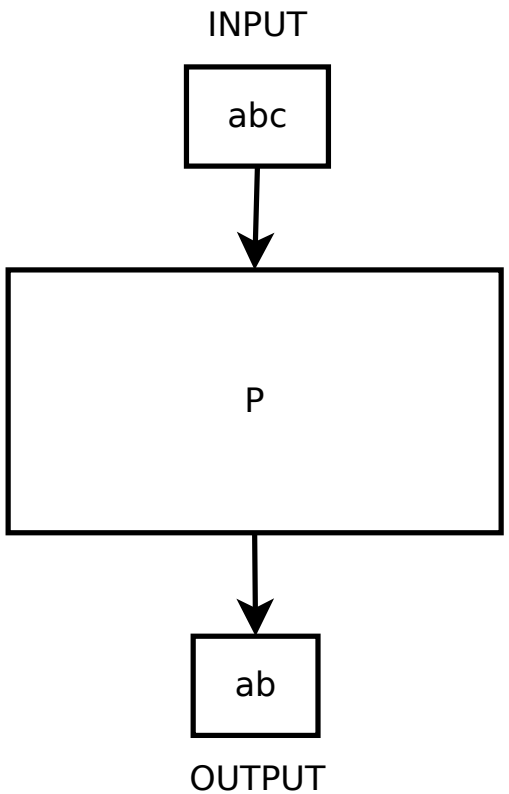
v1(t) :- v2(t-1).
v2(t) :- not v2(t-1), not v1(t-2).
v2(t) :- not v2(t-1), v3(t-1).
v3(t) :- not v3(t-1), not v3(t-2).
v3(t) :- not v1(t-1), not v1(t-2).



Outline

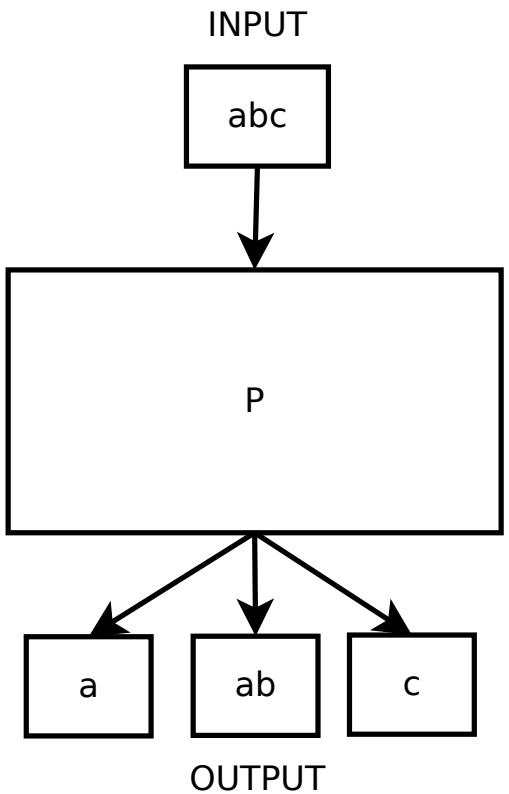
Non-deterministic System

Deterministic system: from the same input we always have the same output.



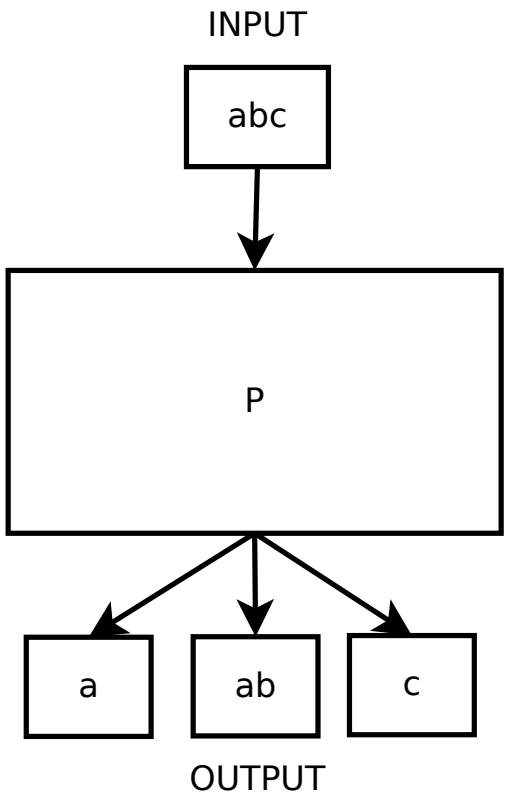
Non-deterministic System

Non-deterministic system: from the same input we can have a different output.



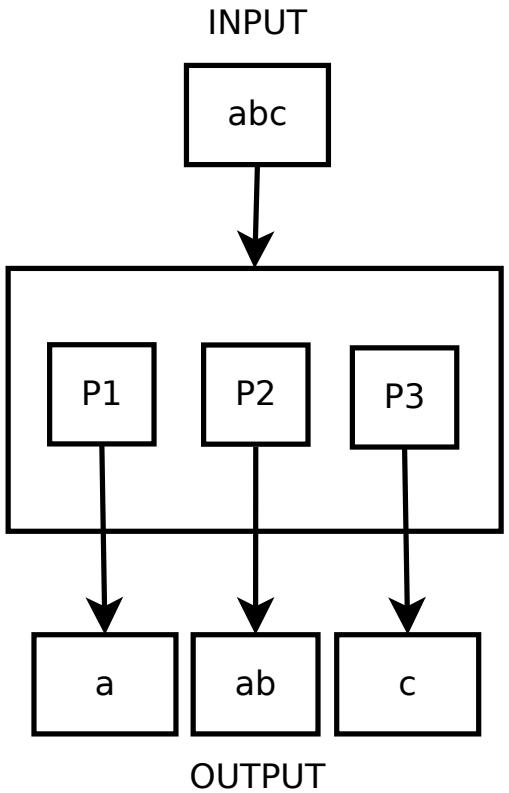
Non-deterministic System

We represent non-deterministic system by a set of logic programs. For an input state, each program provides one possible next state.

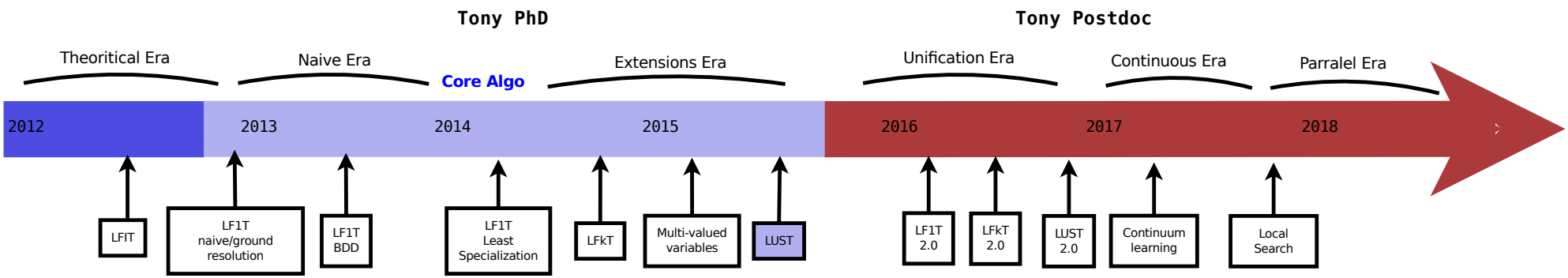


Non-deterministic System

We represent non-deterministic system by a set of logic programs. For an input state, each program provides one possible next state.



LFIT Chronology

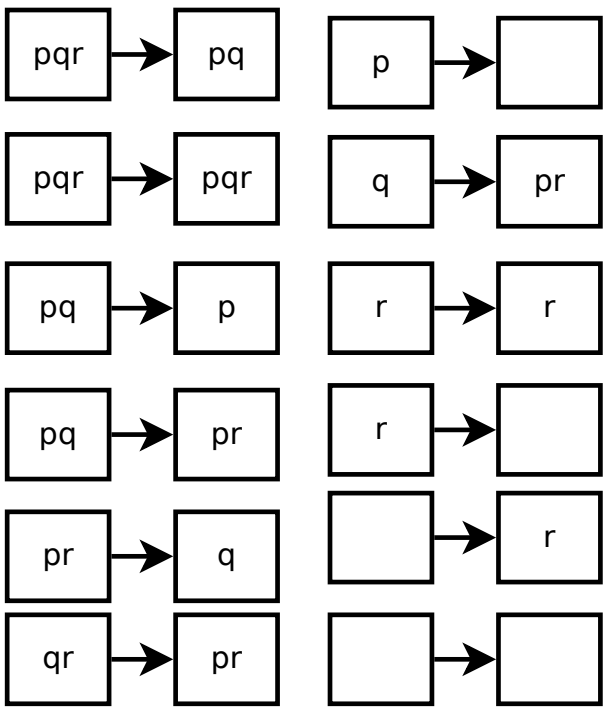




Tech. com **ICLP 2015**, long paper **ICAPS 2016**, journal **JMLR 2017**.

LUST: Learning from Uncertain State Transitions (non-deterministic systems)

INPUT:
A set of pairs of interpretations



OUTPUT:
A set of normal logic programs

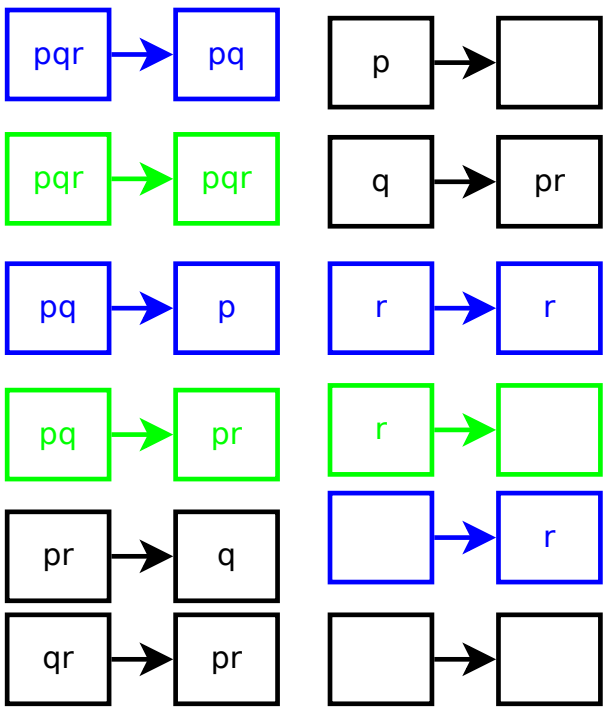
$$p(t) \leftarrow q(t-1).$$
$$q(t) \leftarrow p(t-1) \wedge r(t-1).$$
$$r(t) \leftarrow \neg p(t-1).$$

$$\oplus$$

$$p(t) \leftarrow q(t-1).$$
$$q(t) \leftarrow p(t-1) \wedge r(t-1).$$
$$r(t) \leftarrow q(t-1).$$

LUST: Learning from Uncertain State Transitions (non-deterministic systems)

INPUT:
A set of pairs of interpretations



OUTPUT:
A set of normal logic programs

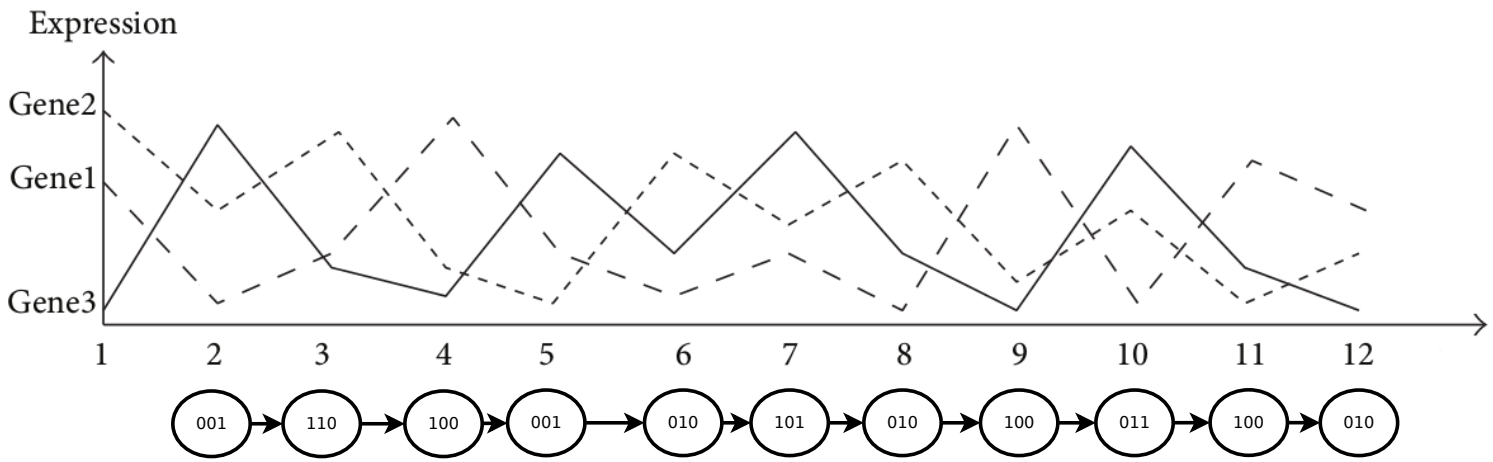
$$p(t) \leftarrow q(t-1).$$
$$q(t) \leftarrow p(t-1) \wedge r(t-1).$$
$$r(t) \leftarrow \neg p(t-1).$$

$$\oplus$$

$$p(t) \leftarrow q(t-1).$$
$$q(t) \leftarrow p(t-1) \wedge r(t-1).$$
$$r(t) \leftarrow q(t-1).$$

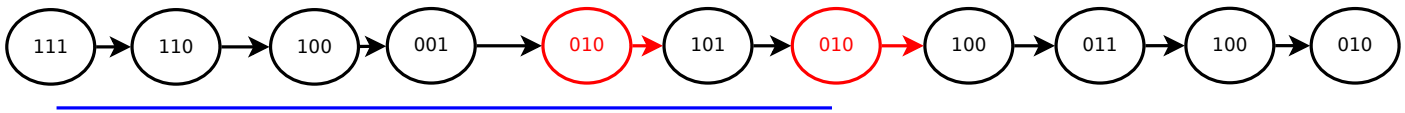
LUST: Learning From Uncertain State Transition

How to learn this set of logic programs from uncertain state transitions ?



LUST: Learning From Uncertain State Transition

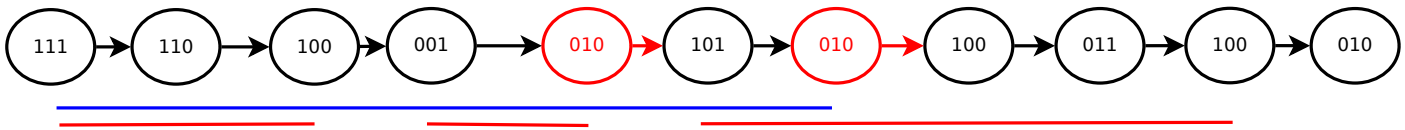
The algorithm starts by learning one program until non-determinism is encountered.




```
v1 :- v2.
v2 :- not v2, v3.
v2 :- v1, v3.
v3 :- not v1, not v3.
v3 :- not v2, not v3.
```

LUST: Learning From Uncertain State Transition

The first program is copied and rules are added to the copy to realize the new transition.

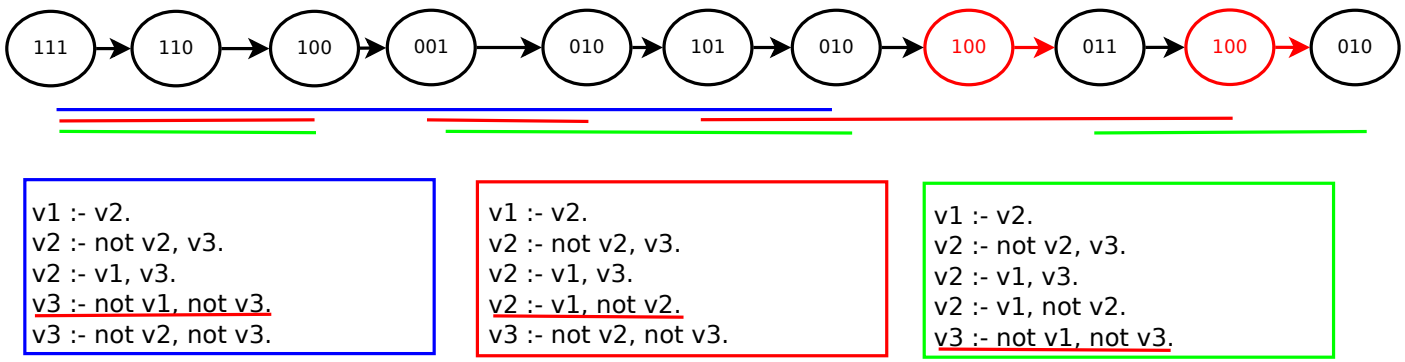



```
v1 :- v2.
v2 :- not v2, v3.
v2 :- v1, v3.
v3 :- not v1, not v3.
v3 :- not v2, not v3.
```

```
v1 :- v2.
v2 :- not v2, v3.
v2 :- v1, v3.
v2 :- v1, not v2.
v3 :- not v2, not v3.
```

LUST: Learning From Uncertain State Transition

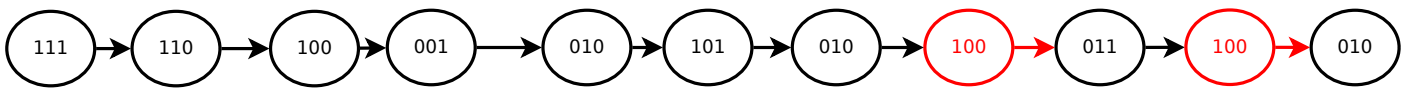
For each transition that is not realized by any program, the same process is performed.



Complexity: $O(d. \sum_{T \in O} |T| \cdot n2^n)$ for run time and $O(d.2^n)$ for memory, with d the maximal out-degree of an observed state.

LUST: Learning From Uncertain State Transition

Counting the number of times a rule matches correctly provides its probability.



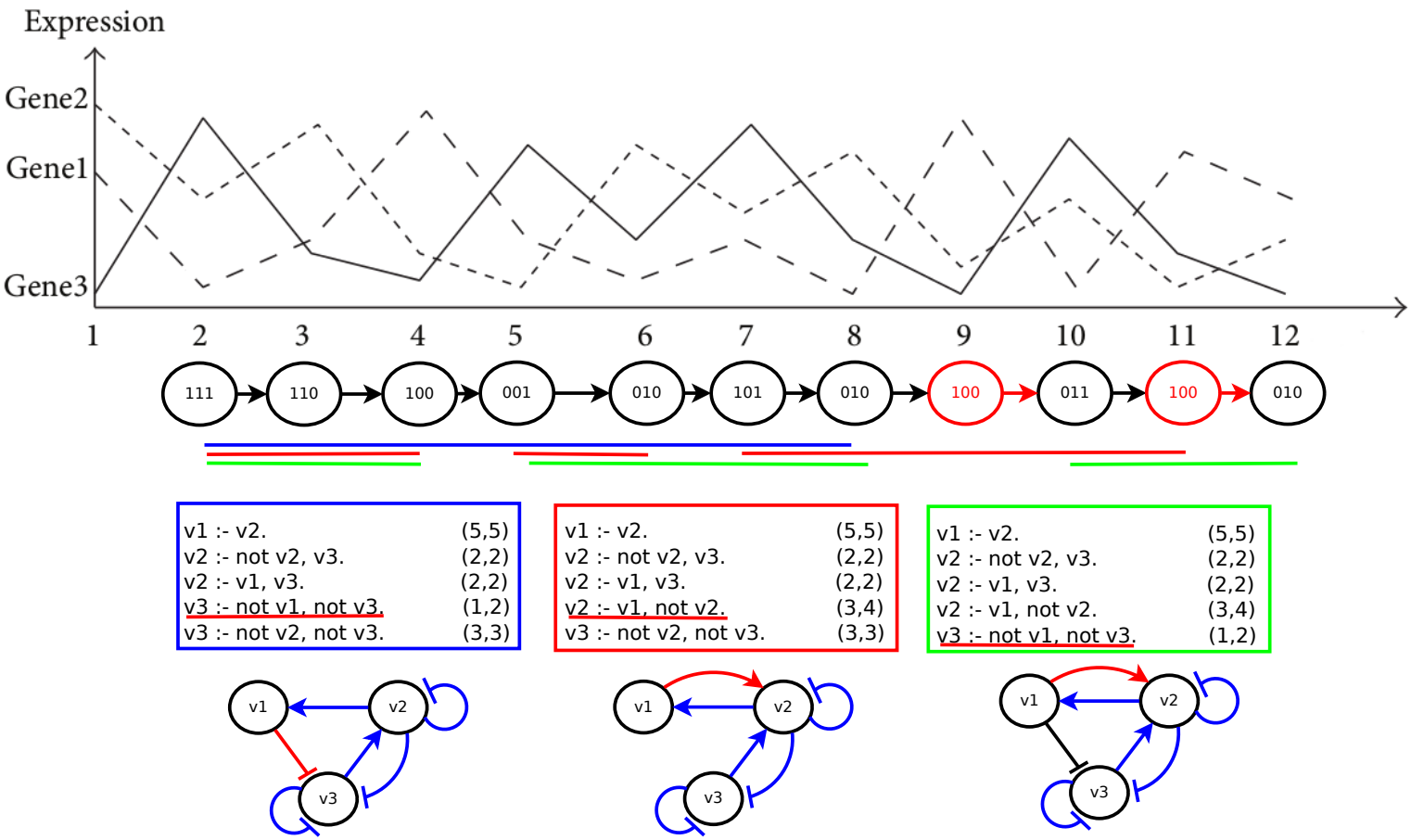
v1 :- v2.	(5,5)
v2 :- not v2, v3.	(2,2)
v2 :- v1, v3.	(2,2)
v3 :- not v1, not v3.	(1,2)
v3 :- not v2, not v3.	(3,3)

v1 :- v2.	(5,5)
v2 :- not v2, v3.	(2,2)
v2 :- v1, v3.	(2,2)
v2 :- v1, not v2.	(3,4)
v3 :- not v2, not v3.	(3,3)

v1 :- v2.	(5,5)
v2 :- not v2, v3.	(2,2)
v2 :- v1, v3.	(2,2)
v2 :- v1, not v2.	(3,4)
v3 :- not v1, not v3.	(1,2)

LUST: Learning From Uncertain State Transition

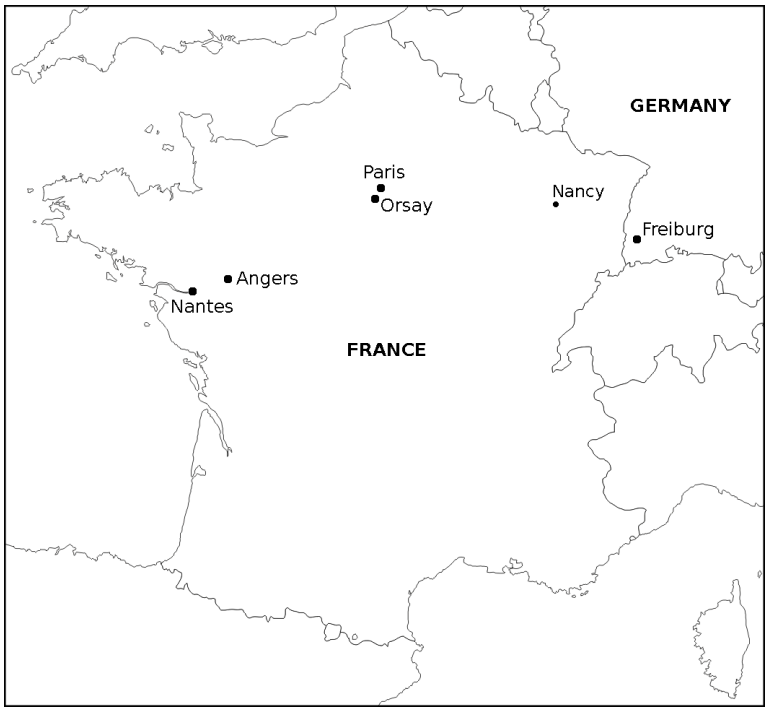
Application: model gene regulatory network from noisy data.

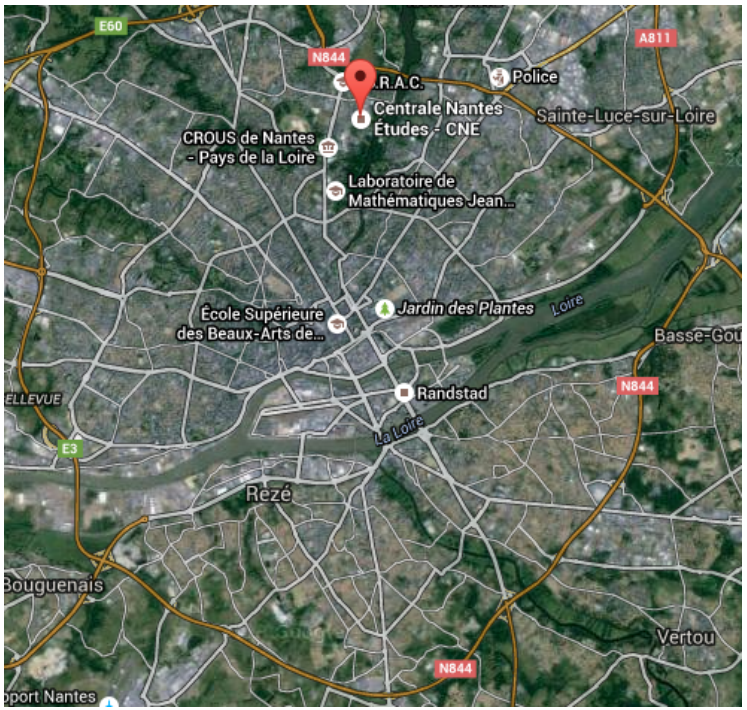


Outline

Laboratory

2 years and 3 months of Postdoc in the MeForBio team of IRCCyN/LS2N at École Centrale de Nantes.





Laboratory

2 years and 3 months of Postdoc in the MeForBio team of IRCCyN/LS2N at École Centrale de Nantes.

Team MeForBio of IRCCyN

1 Professor: Olivier Roux


2 Associate Professors: Morgan Magnin and Carito Guziolowski



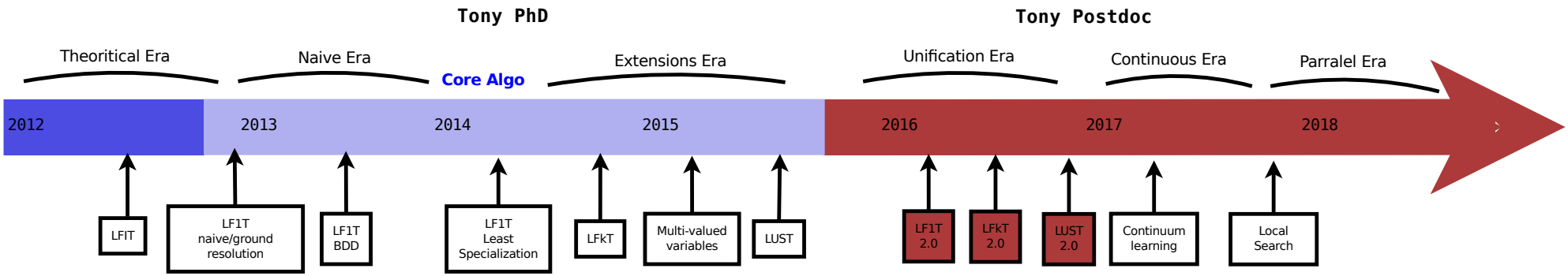

4 PhD: Louis Fippo Fitime, Bertrand Miannay, Emna Ben Abdallah, Misbah Razzak

Motivation: biological system modeling

Projects

- Hyclock: influences modeling
- DREAM Challenge 11: predictions
- Biology Institute of Valrose: gene level of expression

LFIT Chronology



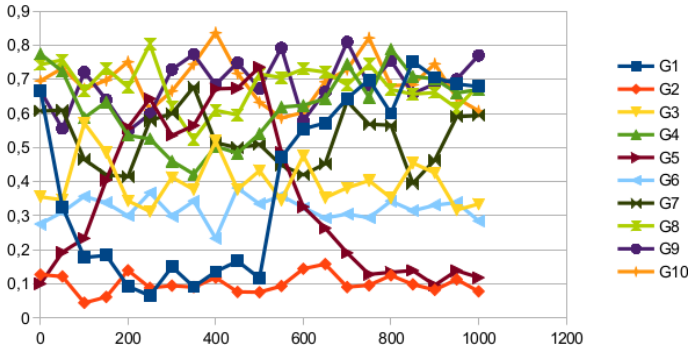
Outline

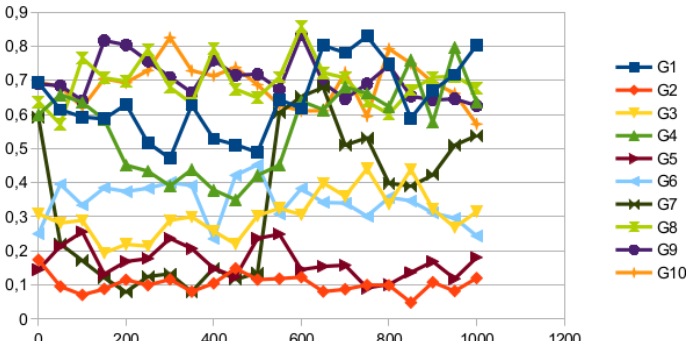
Hyclock project: modeling the mammalian circadian clock

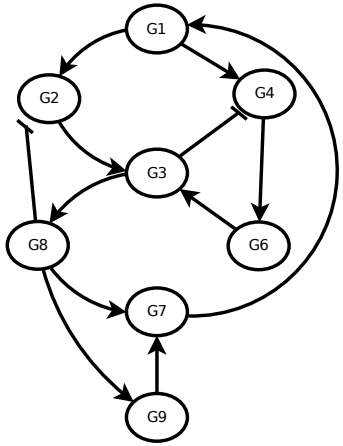
Data

- 45,000 variables
- 2 time series of 30h
- 1 data point per hour for each variable

Goal: extract gene influences







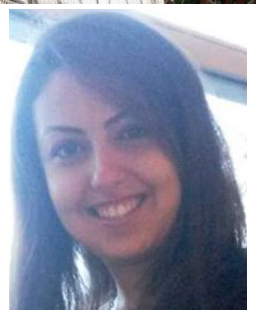
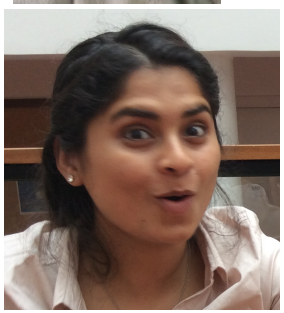
Outline

Team Neo Naoned







3 Professors: Olivier Roux, Paco Chinesta, Katsumi Inoue
2 Associate Professors: Morgan Magnin and Carito Guziolowski
2 Postdocs: Tony Ribeiro and Domenico Borzachiello
3 PhD: Bertrand Miannay, Emna Ben Abdallah, Misbah Razzak

DREAM Challenge 11: prediction

Data

- 120 patients, 7 viruses
- Time series over 22 000 variables (gene probes)
- Meta data: shedding, symptomatic and symptoms

Specificities

- Irregular time points over 240h
- Different sizes of the series

Challenge

- 25 test patients
- Subchallenge 1: predict probability of shedding
- Subchallenge 2: predict probability of symptomatic
- Subchallenge 3: predict logsymptoms

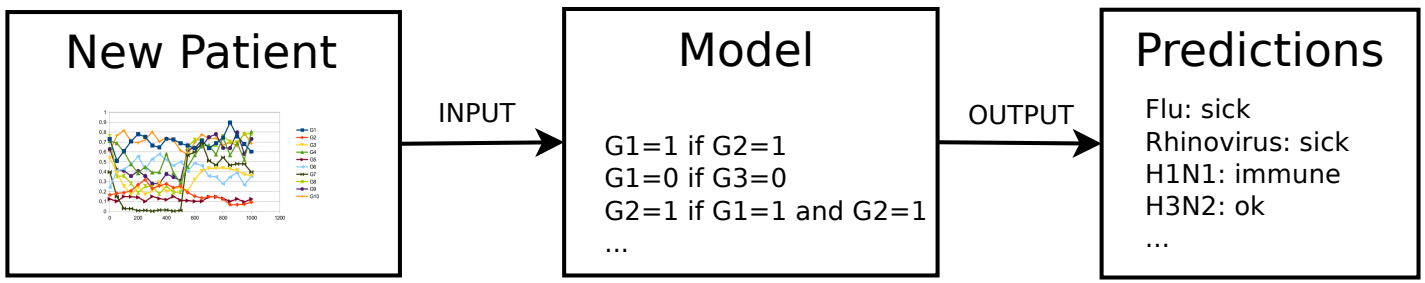
DREAM Challenge 11: prediction

Data

- 120 patients, 7 viruses
- Time series over 22 000 variables (gene probes)
- Meta data: shedding, symptomatic and symptoms

Specificities

- Irregular time points over 240h
- Different sizes of the series



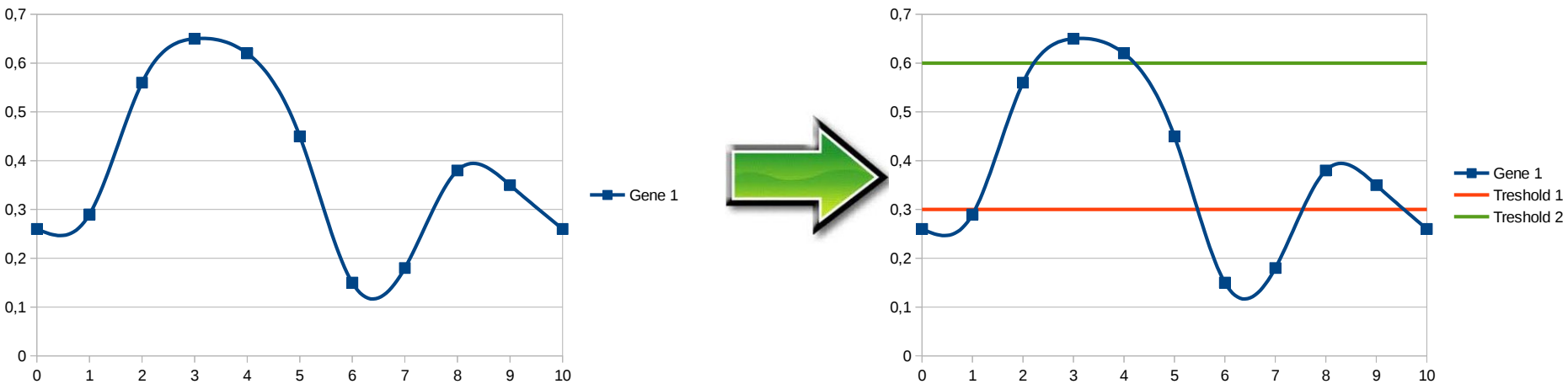
Outline

Identification of gene level of expression

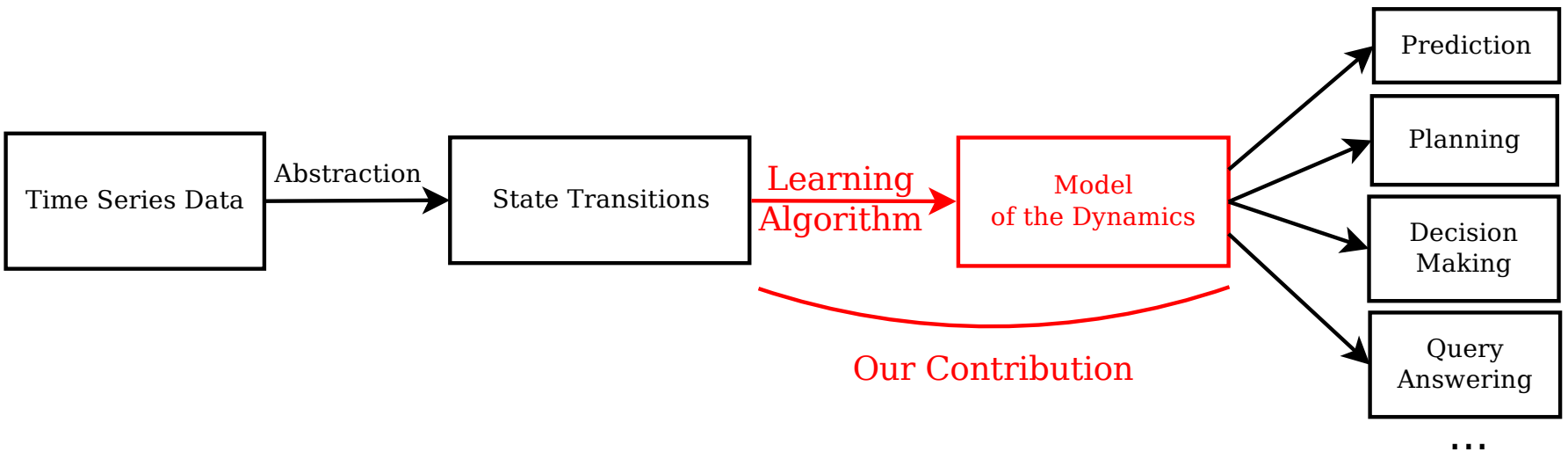
Data

- 2 genes
- time series on demand
- precision on demand

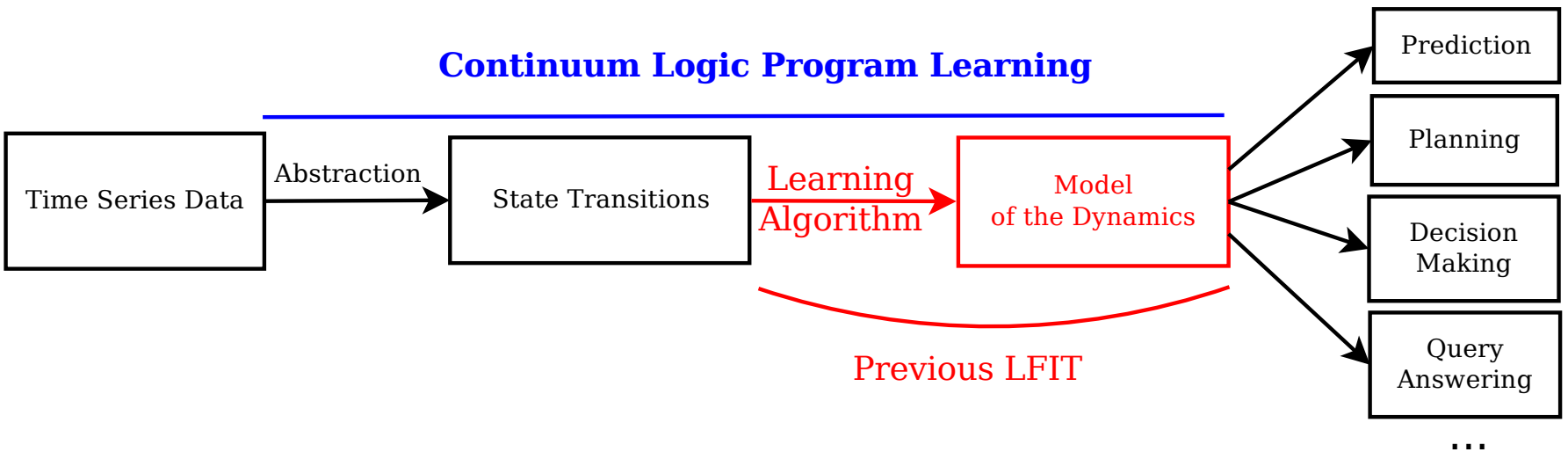
Goal: extract the levels of expression of both gene



LFIT process

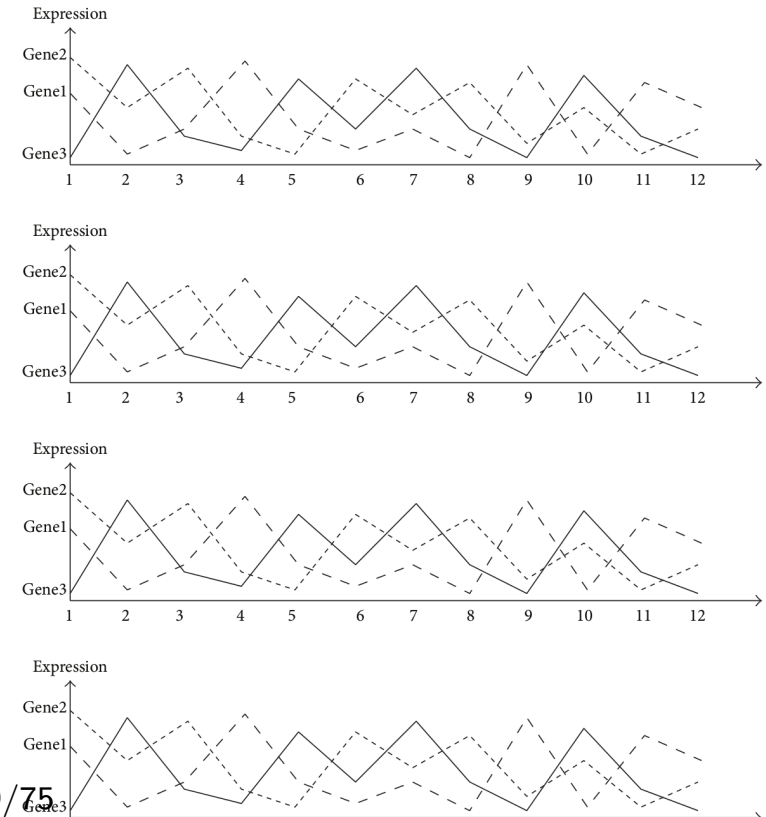


LFIT process



Continuum Logic Program

INPUT:
A set of time series data

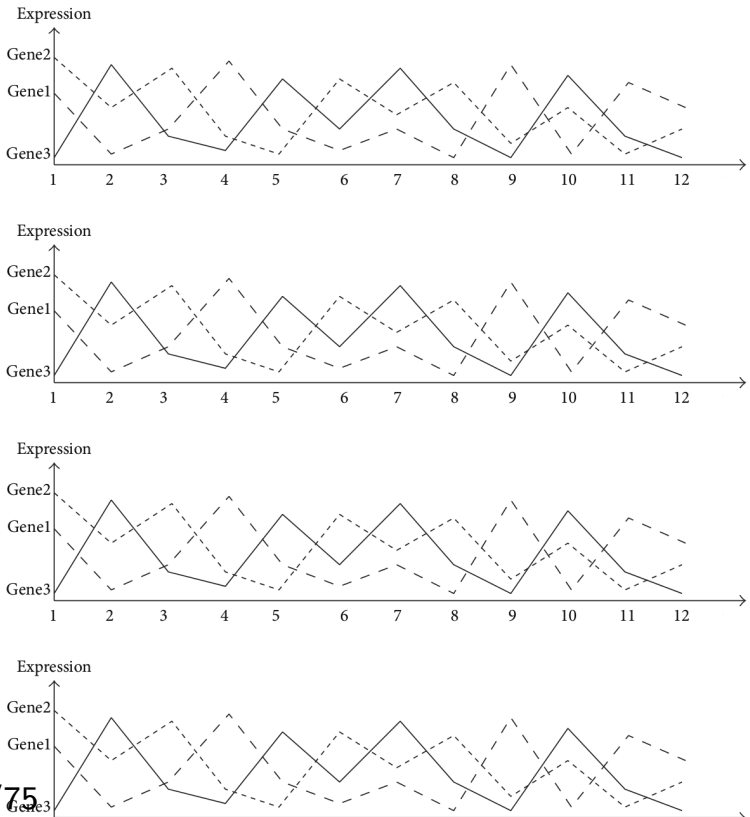


OUTPUT:
A continuum logic program

$$p([0,0.5],t) \leftarrow q([0,0.5],t-1).$$
$$p([0.5,1],t) \leftarrow q([0.5,1],t-1).$$
$$q([0,0.5],t) \leftarrow p([0,0.5],t-1) \wedge r([0.5,1],t-1).$$
$$q([0.5,1],t) \leftarrow p([0.5,1],t-1) \wedge r([0.5,1],t-1).$$
$$r([0,0.5],t) \leftarrow p([0.5,1],t-1).$$
$$r([0.5,1],t) \leftarrow p([0,0.5],t-1).$$

Continuum Logic Program

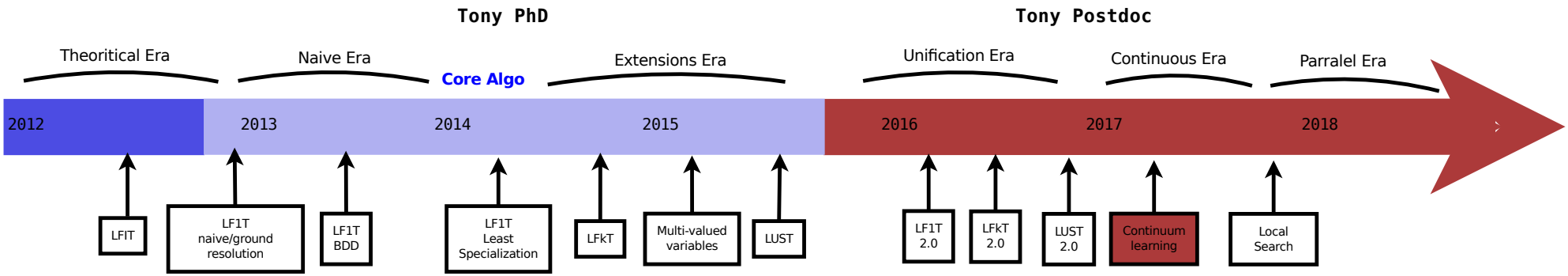
INPUT:
A set of time series data



OUTPUT:
A continuum logic program

$$p([0,0.5],t) \leftarrow q([0,0.5],t-1).$$
$$p([0.5,1],t) \leftarrow q([0.5,1],t-1).$$
$$q([0,0.5],t) \leftarrow p([0,0.5],t-1) \wedge r([0.32,1],t-1).$$
$$q([0.5,1],t) \leftarrow p([0.5,1],t-1) \wedge r([0.32,1],t-1).$$
$$r([0,0.32],t) \leftarrow p([0.5,1],t-1).$$
$$r([0.32,1],t) \leftarrow p([0,0.5],t-1).$$

LFIT Chronology



Long paper **ILP 2017**.

Outline

Conclusion

LFIT contribution summary

- Genes influences (LF1T, LFkT, LUST)
- Gene's level of expression (ACEDIA)
- Prediction (NN-LFIT)

Ongoing

- Local search LFIT
- Temporal logic constraints / Continuous Time Learning
- Learning asynchronous/general semantics

Conclusion

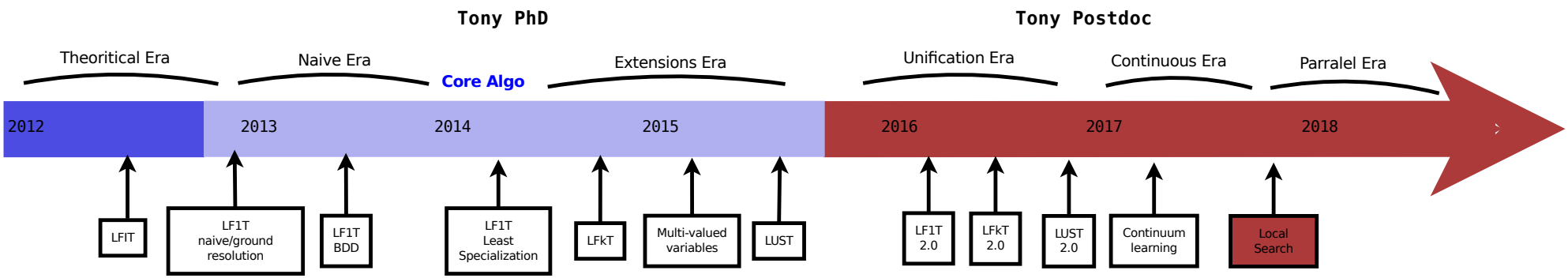
LFIT contribution summary

- Genes influences (LF1T, LFkT, LUST)
- Gene's level of expression (ACEDIA)
- Prediction (NN-LFIT)

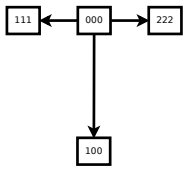
Ongoing

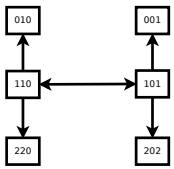
- Local search LFIT
- Temporal logic constraints / Continuous Time Learning
- Learning asynchronous/general semantics

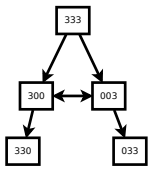
LFIT Chronology

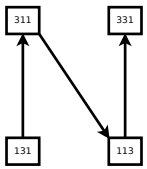


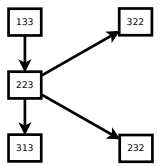
Submitted Journal **MLJ 2018**.

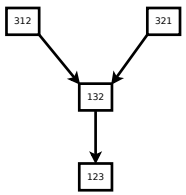

111
000
222
100

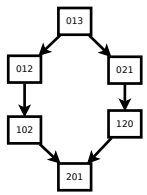

010
001
110
101
220
202

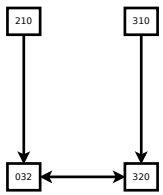

333
300
003
330
033


311
331
131
113


133
322
223
313
232


312
321
132
123


013
012
021
102
120
201


210
310
032
320

Questions?