

# LFIT: Learning from Interpretation Transition

(LFIT: 解釈遷移からの学習)

Katsumi Inoue  
井上 克巳

National Institute of Informatics  
国立情報学研究所

SOKENDAI (The Graduate University for Advanced Studies)  
総合研究大学院大学

Tokyo Institute of Technology  
東京工業大学

*The 100th JSAI SIG-FPAI Meeting*

人工知能学会 第100回人工知能基本問題研究会

*Kumamoto, March 27<sup>th</sup>, 2016*

# Members Involved



Katsumi Inoue (NII/SOKENDAI/Tokyo Tech, Japan)



Chiaki Sakama (Wakayama University, Japan)



Tony Ribeiro (SOKENDAI → École Centrale de Nantes, France)



Morgan Magnin (École Centrale de Nantes/NII)



David Martínez (Institut de Robòtica i Informàtica Industrial, Spain)



Enguerrand Gentet (Université Paris-Sud, France)

and their colleagues

# Contents

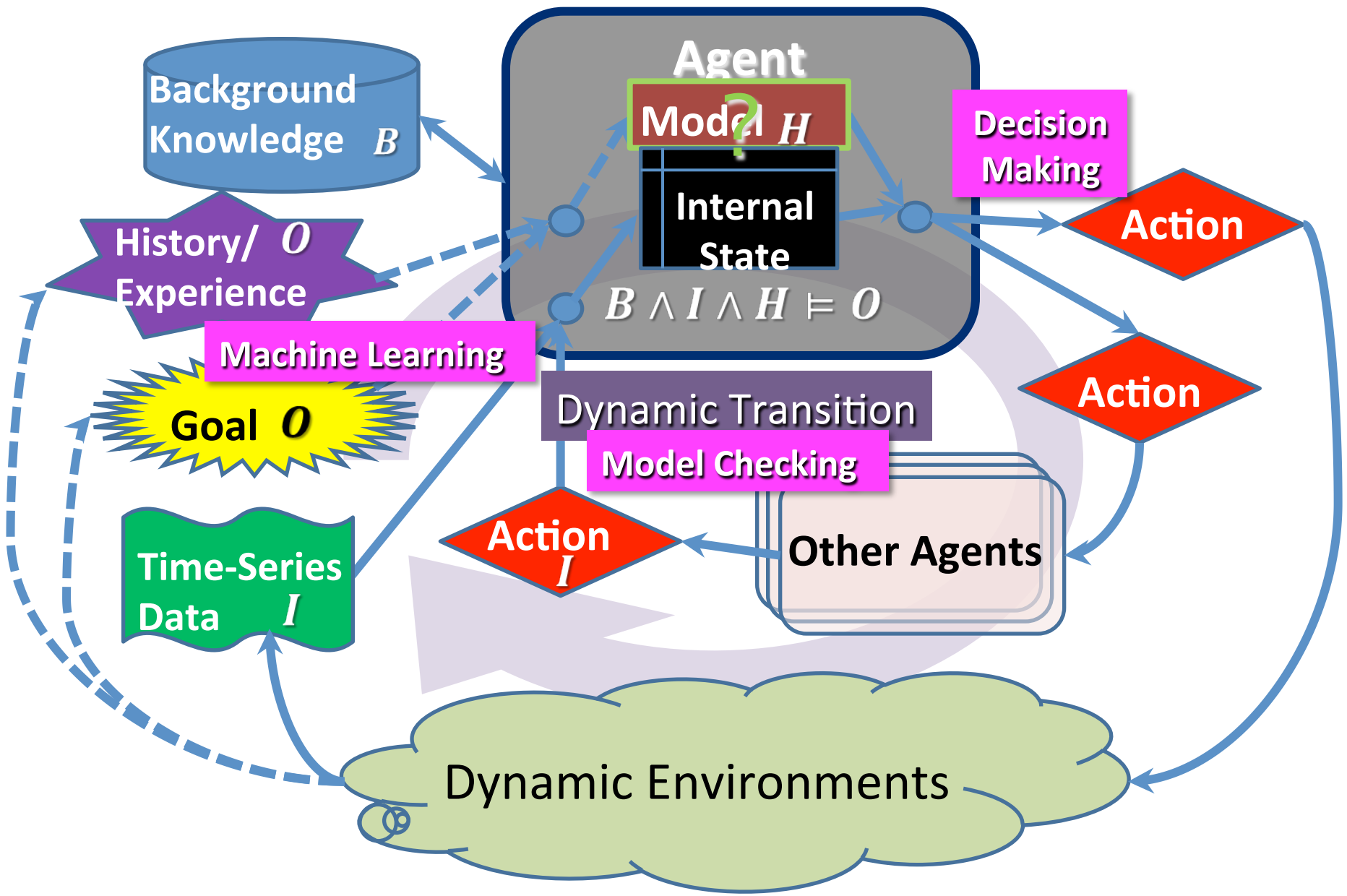
- **Motivation:** Modeling in Dynamic Environments
- **Principles:** LF1T (Learning from 1-Step Transitions)
  - Bottom-Up Algorithm (generalization)
  - BDD Optimization
  - Top-Down Algorithm (specialization)
- **Extensions:** LF $k$ T (Learning Markov( $k$ ) Systems), Multi-Valued/Asynchronous/Probabilistic Extensions
- **Applications:** Biology, Robotics, AGI, etc.
- **Ongoing Work:** DREAM Challenges, NN-LFIT

# Contents

- **Motivation:**
  - Modeling in Dynamic Environments
- **Principles**
- **Extensions**
- **Applications**
- **Ongoing Work**

# AI in Dynamic Environments

- Identifying the **model** of a system in **dynamic** environments in order to achieve tasks even when unknown situations are encountered.
  - **Phase 1:** The internal model is constructed by learning from environment and interaction with other systems.
  - **Phase 2:** The model is used for choosing the next action.
  - **Phase Update:** The effect/result of an action affects the environment and updates time-series data, history, experience and goals. Then the model is updated accordingly.
  - The agent also interacts with other agents, and its internal model is refined by such interactions.

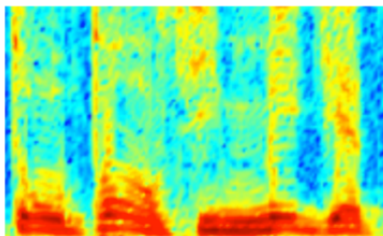
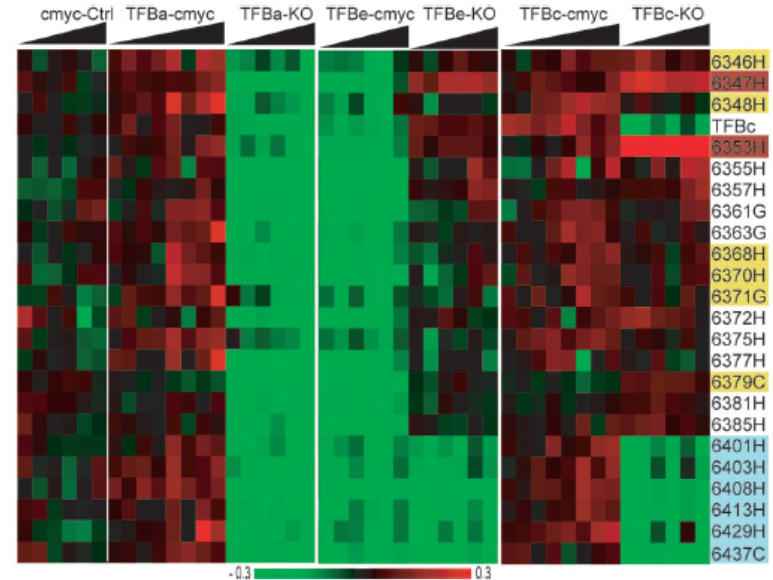


# Vector Representations of Worlds

$$(x_1, \dots, x_n) = (a_1, \dots, a_n) : \textit{static}$$

$$(x_1^t, \dots, x_n^t) = (a_1^t, \dots, a_n^t) : \textit{dynamic}$$

**Heat map:** Transcriptome data for a subset of eukaryotic TF II B (TFB) perturbations (Facciotti, M., *et al.*, *PNAS*, Vo.104, 2007)



Audio Spectrogram

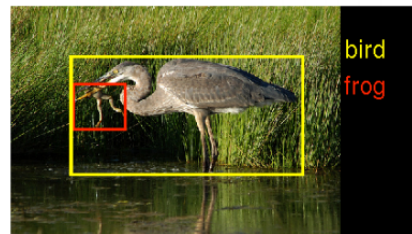
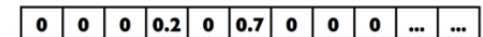


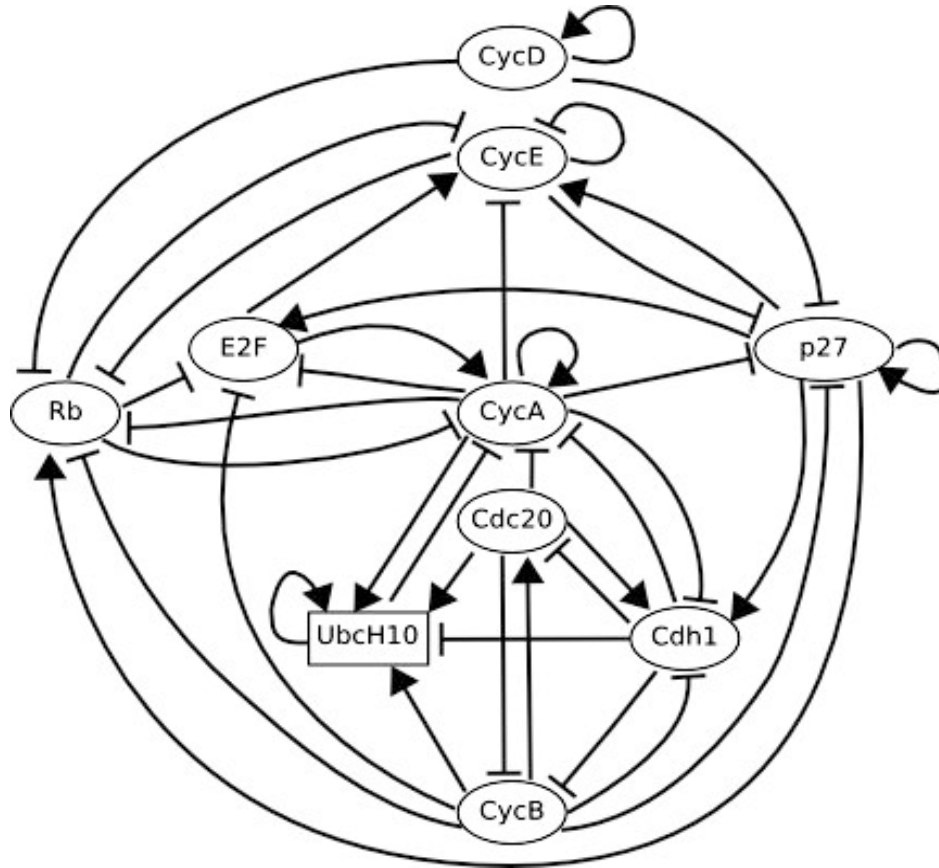
Image pixels



Word, context, or document vectors

# Network Representations of Models

- Mammalian Cell Cycle Network



Product ← Rules leading to activity

<i>CycD</i>	$\overline{CycD}$
<i>Rb</i>	$(\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{CycD} \wedge \overline{CycB})$
<i>E2F</i>	$(\overline{Rb} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{Rb} \wedge \overline{CycB})$
<i>CycE</i>	$(E2F \wedge \overline{Rb})$
<i>CycA</i>	$(E2F \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge (\overline{Cdh1} \wedge \overline{Ubc})) \vee (CycA \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge (\overline{Cdh1} \wedge \overline{Ubc}))$
<i>p27</i>	$(\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge (\overline{CycE} \wedge \overline{CycA}) \wedge \overline{CycB} \wedge \overline{CycD})$
<i>Cdc20</i>	$\overline{CycB}$
<i>Cdh1</i>	$(\overline{CycA} \wedge \overline{CycB}) \vee (Cdc20) \vee (p27 \wedge \overline{CycB})$
<i>UbcH10</i>	$(\overline{Cdh1}) \vee (Cdh1 \wedge \overline{Ubc} \wedge (\overline{Cdc20} \vee \overline{CycA} \vee \overline{CycB}))$
<i>CycB</i>	$(\overline{Cdc20} \wedge \overline{Cdh1})$

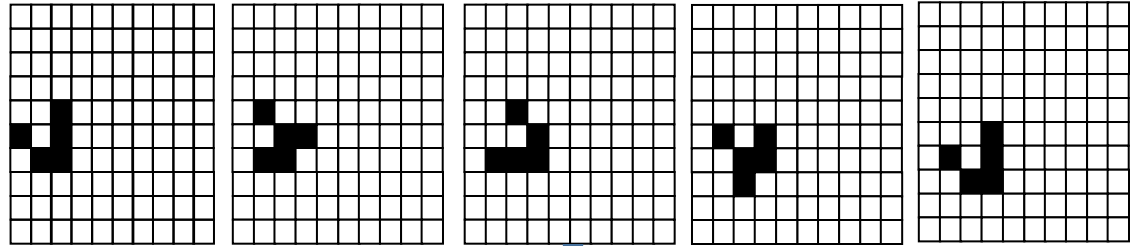
Each node represents the activity of a key regulatory element, whereas the edges represent cross-regulations. Blunt arrows stand for inhibitory effects, normal arrows for activations.

(Fauré et al., *Bioinformatics*, **22**, 2006)



# Logical Representations of Models

- Boolean Networks
- Cellular Automata



Game of Life

$$c(x_i, y_j, t+1) \leftarrow c(x_i, y_j, t), \mathbf{2} \{ c(x_{i-1}, y_{j-1}, t), c(x_i, y_{j-1}, t), c(x_{i+1}, y_{j-1}, t), c(x_{i-1}, y_j, t), c(x_{i+1}, y_j, t), c(x_{i-1}, y_{j+1}, t), c(x_i, y_{j+1}, t), c(x_{i+1}, y_{j+1}, t) \} \mathbf{3}.$$

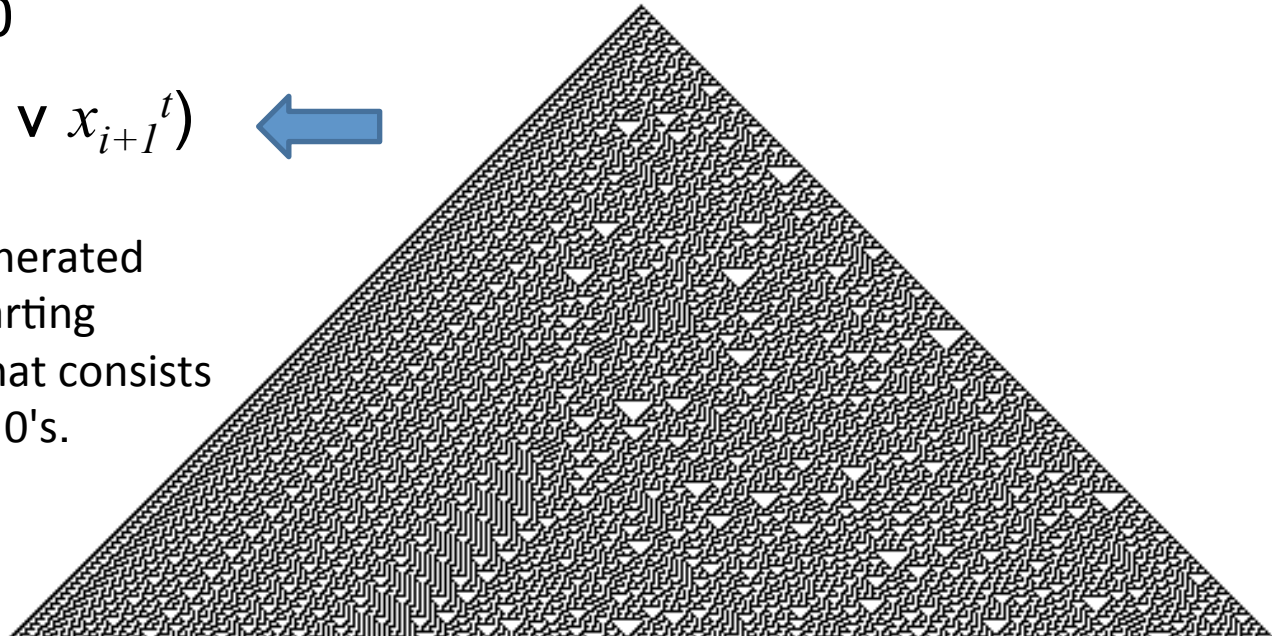
$$C(x_i, y_j, t+1) \leftarrow \mathbf{not} c(x_i, y_j, t), \mathbf{3} \{ c(x_{i-1}, y_{j-1}, t), c(x_i, y_{j-1}, t), c(x_{i+1}, y_{j-1}, t), c(x_{i-1}, y_j, t), c(x_{i+1}, y_j, t), c(x_{i-1}, y_{j+1}, t), c(x_i, y_{j+1}, t), c(x_{i+1}, y_{j+1}, t) \} \mathbf{3}.$$

Wolfram's Rule 30

$$x_i^{t+1} = x_{i-1}^t \oplus (x_i^t \vee x_{i+1}^t)$$



The history of the generated patterns with the starting configuration (top) that consists of a 1 surrounded by 0's.

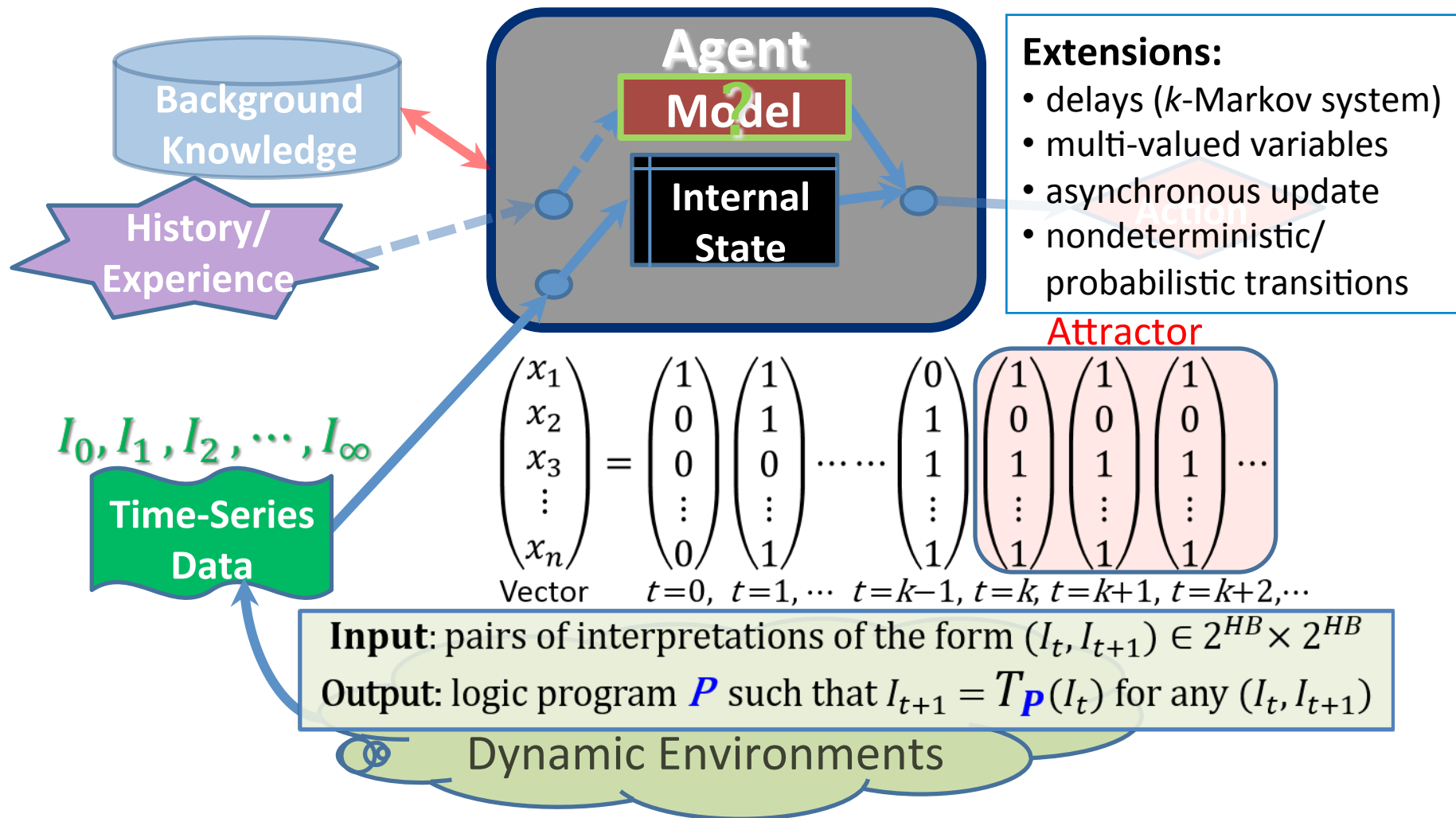




# Contents

- **Motivation**
- **Principles:**
  - LF1T (Learning from 1-Step Transitions)
    - Bottom-Up Algorithm (generalization)
    - BDD Optimization
    - Top-Down Algorithm (specialization)
- **Extensions**
- **Applications**
- **Ongoing Work**

# Learning From Interpretation Transition (LFIT)



- Inoue, K., Ribeiro, T., Sakama, C.: "Learning from Interpretation Transition", *Machine Learning*, 94(1):51-79, 2014.

# Normal Logic Programs

- A *normal logic program* (NLP)  $P$  is a set of rules:

$$H \leftarrow A_1 \wedge \dots \wedge A_m \wedge \neg B_1 \wedge \dots \wedge \neg B_n \quad (m, n \geq 0)$$

where  $H$ ,  $A_i$  and  $B_j$  are atoms and  $\neg$  is (*default*) *negation*.

- $P$  is *definite* if  $n = 0$  for every rule in  $P$ .
- $ground(P)$  : the set of ground instances of all rules in  $P$ .
- The *Herbrand base*  $\mathbf{HB}_P$  is the set of ground atoms in language( $P$ ).
- An (*Herbrand*) *interpretation*  $I$  of an NLP  $P$ :  $I \subseteq \mathbf{HB}_P$  ( $I \in 2^{\mathbf{HB}_P}$ ).
- $I \in 2^{\mathbf{HB}_P}$  *satisfies* ( $\models$ ) a ground rule of the form:

$$H \leftarrow A_1 \wedge \dots \wedge A_m \wedge \neg B_1 \wedge \dots \wedge \neg B_n$$

iff  $\forall i. A_i \in I$  and  $\forall j. B_j \notin I$  imply that  $H \in I$ .

- $I$  is an (*Herbrand*) *model* of  $P$  if  $I$  satisfies all rules in  $ground(P)$ .

# $T_P$ operator

- $T_P(I) := \{ H \mid H \leftarrow L_1 \wedge \dots \wedge L_n \in \text{ground}(P), I \models L_1 \wedge \dots \wedge L_n \}$ .

- When  $P$  is a *definite* program,  $I \models A_1 \wedge \dots \wedge A_m$  iff  $\forall i. A_i \in I$ .

In this case,  $T_P$  operator is *monotone*, and the sequence

$$I_0 = \{\}, \quad I_{n+1} = T_P(I_n) \quad (n=0, \dots)$$

reaches the *least fixpoint* of  $T_P$ , denoted as  $I^* = T_P \uparrow \omega$ :  $I^* = T_P(I^*)$ .

$T_P \uparrow \omega$  is the *least model* of  $P$  (van Emden & Kowalski, 1976).

- When  $P$  is a *normal* program,

$$I \models A_1 \wedge \dots \wedge A_m \wedge \neg B_1 \wedge \dots \wedge \neg B_n \text{ iff } \forall i. A_i \in I \text{ and } \forall j. B_j \notin I.$$

In this case,  $T_P$  is *non-monotone* (Apt, Blair & Walker, 1988).

- The *orbit* of  $I$  wrt  $P$  (Blair *et al.*, 1997) is  $\langle T_P^k(I) \rangle_{k=0,1,2,\dots}$ ,

where  $T_P^0(I) = I$ ,  $T_P^{k+1}(I) = T_P(T_P^k(I))$  for  $k = 0, 1, 2, \dots$ .

# $T_P$ operator, supportedness, completion

- An interpretation  $I$  is **supported** (Apt, Blair & Walker, 1988) if  $\forall A \in I. \exists (A \leftarrow A_1 \wedge \dots \wedge A_m \wedge \neg B_1 \wedge \dots \wedge \neg B_n) \in \text{ground}(P)$  such that  $\forall i. A_i \in I$  and  $\forall j. B_j \notin I$ .
- **Prop.:** An interpretation  $I$  is a model of  $P$  iff  $T_P(I) \subseteq I$ .
- **Prop.:**  $I$  is supported iff  $I \subseteq T_P(I)$ .
- **Cor.:**  $I$  is a *supported model* of  $P$  iff  $I = T_P(I)$ .
- **Prop.:**  $I$  is a model of  $\text{Comp}(P)$  iff  $I = T_P(I)$ , where  $\text{Comp}(P)$  is Clark's completion of  $P$ .
- **Cor.:**  $I$  is a supported model of  $P$  iff  $I$  is a model of  $\text{Comp}(P)$ .

# $T_p$ operator for NLPs

- $p \leftarrow q.$
  - $p \leftarrow \neg r.$
  - $r \leftarrow p \wedge \neg q.$
1.  $\{\}$
  2.  $\{p\}$
  3.  $\{p,r\}$
  4.  $\{r\}$
  5.  $\{\}$
  6. repeat 2—5
- $T_p$  is *non-monotone*.
  - No fixpoint is reached in general.
  - No supported model exists here.



# Learning Logical Dynamics of Systems

- Learning action theories in ILP
  - Event calculus: Moyle & Muggleton (1997), Moyle (2003)
  - Logic programs: with situation calculus: Otero (2003, 2005)
  - Action languages: Inoue *et al.* (2005), Tran & Baral (2009)
  - Probabilistic logic programs: Corapi *et al.* (2011)
- Relational reinforcement learning
  - Logic programs: Džeroski *et al.* (2001)
- Abductive action learning
  - Abductive event calculus: Eshghi (1988), Shanahan (2000)
- Learning Petri nets : Srinivasan *et al.* (2015)
- Active learning of action models
  - STRIPS-like: Rodrigues *et al.* (2011)
- These works suppose applications to robotics and bioinformatics.
- However, it is hard to infer *rules of systems dynamics* due to presence of positive and negative feedbacks.

# LFIT: Learning from Interpretation Transitions

(Inoue, Ribeiro & Sakama, *Machine Learning*, 2014)

- Herbrand interpretation  $I$ : a state of the world
- Logic program  $P$ : a *state transition system*, which maps an Herbrand interpretation into another interpretation
- Next state  $T_P(I)$ : where  $T_P$  is the immediate consequence operator ( $T_P$  operator).
- Learning setting:
  - Given: a set of pairs of Herbrand interpretations  $(I, J)$  such that  $J = T_P(I)$ ,
  - Induce a normal logic program  $P$ .
- c.f. learning from interpretations (LFI)
  - Given: a set  $S$  of Herbrand interpretations,
  - Induce a program  $P$  whose models are exactly  $S$ .

# Subsumption, least generalization

- For two rules  $R_1, R_2$  with the same head,  $R_1$  *subsumes*  $R_2$  if there is a substitution  $\theta$  s.t.  $b^+(R_1)\theta \subseteq b^+(R_2)$  and  $b^-(R_1)\theta \subseteq b^-(R_2)$ .
- A rule  $R$  is the *least (general) generalization (lg)* of  $R_1$  and  $R_2$ , written as  $R = lg(R_1, R_2)$ , if  $R$  subsumes both  $R_1$  and  $R_2$  and is subsumed by any rule that subsumes both  $R_1$  and  $R_2$ .
- The lg of two atoms  $p(s_1, \dots, s_n)$  and  $q(t_1, \dots, t_n)$  is *undefined* if  $p \neq q$ ; and is  $p(lg(s_1, t_1), \dots, lg(s_n, t_n))$  if  $p = q$ .
- The lg of two rules  $lg(R_1, R_2)$  is then written as:

$$lg(h(R_1), h(R_2)) \leftarrow \bigwedge_{L \in b^+(R_1), K \in b^+(R_2)} lg(L, K) \quad \wedge \quad \bigwedge_{L \in b^-(R_1), K \in b^-(R_2)} \neg lg(L, K).$$

# LF1T: Learning from 1-Step Transitions

- **Input:**  $E \subseteq 2^{\text{HB}} \times 2^{\text{HB}}$ : (positive) examples/observations,  
 $P$ : an (initial) NLP;
- **Output:** NLP  $P$  s.t.  $J = T_p(I)$  holds for any  $(I, J) \in E$

## Bottom-Up Algorithm (Inoue *et al.*, 2012;2014)

1. If  $E = \emptyset$ , then output  $P$  and stop;
2. Pick  $(I, J) \in E$ ; put  $E := E \setminus \{(I, J)\}$ ;
3. For each  $A \in J$ , let

$$R'_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in \text{HB} \setminus I} \neg C;$$

1. If  $R'_A$  is not subsumed by any rule in  $P$ , then  $P := P \cup \{R'_A\}$  and simplify  $P$  by generalizing some rules in  $P$  and removing all clauses subsumed by them;
2. Return to 1.

# Resolution as Generalization

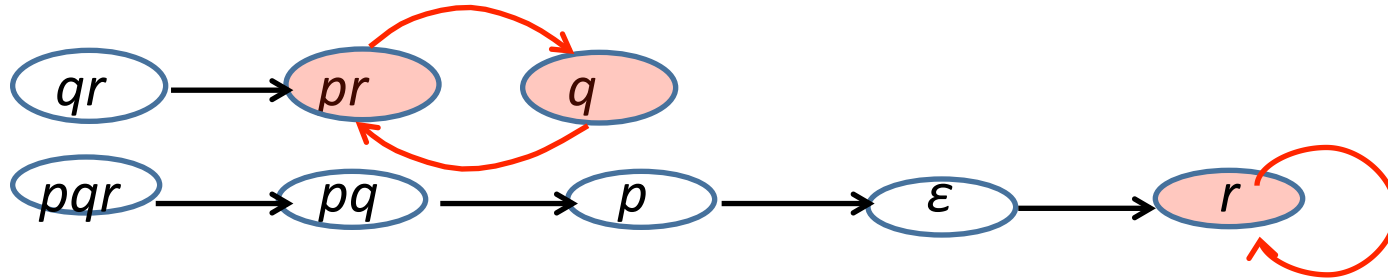
- **(naïve/ground resolution)** Let  $R_1$  and  $R_2$  be two ground rules, and  $l$  be a literal such that  $h(R_1) = h(R_2)$ ,  $l \in b(R_1)$  and  $\bar{l} \in b(R_2)$ . If  $(b(R_2) \setminus \{\bar{l}\}) \subseteq (b(R_1) \setminus \{l\})$  then the *ground resolution* of  $R_1$  and  $R_2$  (upon  $l$ ) is defined as

$$res(R_1, R_2) := h(R_1) \leftarrow \bigwedge_{K \in b(R_1) \setminus \{l\}} K$$

In particular, if  $(b(R_2) \setminus \{\bar{l}\}) = (b(R_1) \setminus \{l\})$  then the ground resolution is called the *naïve resolution* of  $R_1$  and  $R_2$  (upon  $l$ ).

- **Example.**  $R_1 = (p \leftarrow q \wedge r)$ ,  $R_2 = (p \leftarrow \neg q \wedge r)$ ,  $R_3 = (p \leftarrow \neg q)$ :  
 $res(R_1, R_2) = res(R_1, R_3) = (p \leftarrow r)$ .
- **Proposition:** The naïve resolution of  $R_1$  and  $R_2$  is the least generalization of them, e.g.,  $lg(R_1, R_2) = res(R_1, R_2)$ .

# LF1T (naïve resolution) $[R^I_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in \text{HBV}} \neg C]$



Step	$I \rightarrow J$	Operation	Rule	ID	$P$	$P_{old}$
1	$qr \rightarrow pr$	$R^{qr}_p$	$p \leftarrow \neg p \wedge q \wedge r$	1	1	{}
		$R^{qr}_r$	$r \leftarrow \neg p \wedge q \wedge r$	2	1,2	
2	$pr \rightarrow q$	$R^{pr}_q$	$q \leftarrow p \wedge \neg q \wedge r$	3	1,2,3	
3	$q \rightarrow pr$	$R^q_p$	$p \leftarrow \neg p \wedge q \wedge \neg r$	4		
		$res(4,1)$	$p \leftarrow \neg p \wedge q$	5	2,3,5	+1,4
		$R^q_r$	$r \leftarrow \neg p \wedge q \wedge \neg r$	6		
		$res(6,2)$	$r \leftarrow \neg p \wedge q$	7	3,5,7	+2,6
4	$pqr \rightarrow pq$	$R^{pqr}_p$	$p \leftarrow p \wedge q \wedge r$	8		
		$res(8,1)$	$p \leftarrow q \wedge r$	9	3,5,7,9	+8
		$R^{pqr}_q$	$q \leftarrow p \wedge q \wedge r$	10		
		$res(10,3)$	$q \leftarrow p \wedge r$	11	5,7,9,11	+3,10

# Cont. (naïve resolution) $[R'_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in \text{HBV}} \neg C]$

Step	$I \rightarrow J$	Operation	Rule	ID	$P$	$P_{old}$
5	$pq \rightarrow p$	$R_p^{pq}$	$p \leftarrow p \wedge q \wedge \neg r$	12		
		$res(12,5)$	$p \leftarrow q \wedge \neg r$	13	5,7,9,11,13	+12
		$res(13,9)$	$p \leftarrow q$	14	7,11,14	+5,9,13
6	$p \rightarrow \varepsilon$					
7	$\varepsilon \rightarrow r$	$R_r^\varepsilon$	$r \leftarrow \neg p \wedge \neg q \wedge \neg r$	15		
		$res(15,6)$	$r \leftarrow \neg p \wedge \neg r$	16	7,11,14,16	+15
8	$r \rightarrow r$	$R_r^r$	$r \leftarrow \neg p \wedge \neg q \wedge r$	17		
		$res(17,15)$	$r \leftarrow \neg p \wedge \neg q$	18	7,11,14,16,18	+17
		$res(18,7)$	$r \leftarrow \neg p$	19	11,14,19	+7,16,18

$$p \leftarrow q.$$

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

$$r \leftarrow \neg p.$$

propositional program

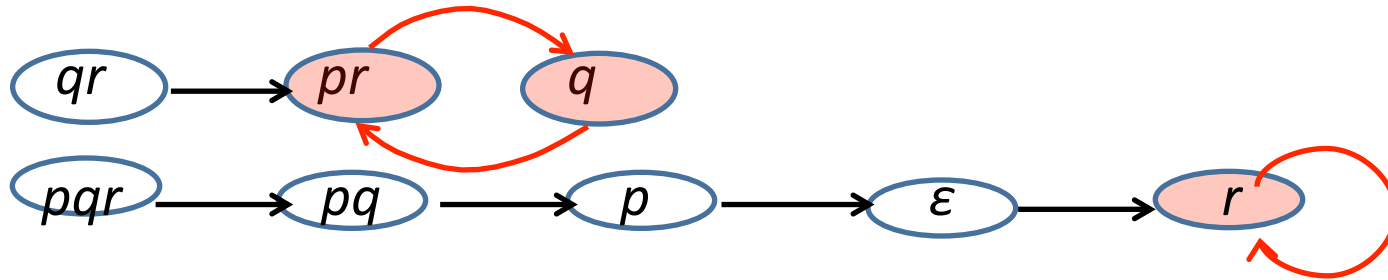
$$p(t+1) \leftarrow q(t).$$

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

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

first-order program

# LF1T (ground resolution) $[R'_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in HB \setminus I} \neg C]$



Step	$I \rightarrow J$	Operation	Rule	ID	$P$
1	$qr \rightarrow pr$	$R^{qr}_p$	$p \leftarrow \neg p \wedge q \wedge r$	1	1
		$R^{qr}_r$	$r \leftarrow \neg p \wedge q \wedge r$	2	1,2
2	$pr \rightarrow q$	$R^{pr}_q$	$q \leftarrow p \wedge \neg q \wedge r$	3	1,2,3
3	$q \rightarrow pr$	$R^q_p$	$p \leftarrow \neg p \wedge q \wedge \neg r$	4	
		$res(4,1)$	$p \leftarrow \neg p \wedge q$	5	2,3,5
		$R^q_r$	$r \leftarrow \neg p \wedge q \wedge \neg r$	6	
		$res(6,2)$	$r \leftarrow \neg p \wedge q$	7	3,5,7
4	$pqr \rightarrow pq$	$R^{pqr}_p$	$p \leftarrow p \wedge q \wedge r$	8	
		$res(8,5)$	$p \leftarrow q \wedge r$	9	3,5,7,9
		$R^{pqr}_q$	$q \leftarrow p \wedge q \wedge r$	10	
		$res(10,3)$	$q \leftarrow p \wedge r$	11	5,7,9,11



# Cont. (ground resolution) $[R'_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in \text{HB} \setminus I} \neg C]$

Step	$I \rightarrow J$	Operation	Rule	ID	$P$
5	$pq \rightarrow p$	$R_p^{pq}$	$p \leftarrow p \wedge q \wedge \neg r$	12	
		$\text{res}(12,5)$	$p \leftarrow q \wedge \neg r$	13	5,7,9,11,13
		$\text{res}(13,9)$	$p \leftarrow q$	14	7,11,14
6	$p \rightarrow \varepsilon$				
7	$\varepsilon \rightarrow r$	$R_r^\varepsilon$	$r \leftarrow \neg p \wedge \neg q \wedge \neg r$	15	
		$\text{res}(15,7)$	$r \leftarrow \neg p \wedge \neg r$	16	7,11,14,16
8	$r \rightarrow r$	$R_r^r$	$r \leftarrow \neg p \wedge \neg q \wedge r$	17	
		$\text{res}(17,7)$	$r \leftarrow \neg p \wedge \neg q$	18	7,11,14,16,18
		$\text{res}(18,16)$	$r \leftarrow \neg p$	19	11,14,19

$$p \leftarrow q.$$

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

$$r \leftarrow \neg p.$$

propositional program

$$p(t+1) \leftarrow q(t).$$

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

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

first-order program

# Worst-Case Complexity

- **Theorem:** Using naïve resolution, the memory use of the LF1T algorithm is bounded by  $O(n \cdot 3^n)$ , and the time complexity of learning is bounded by  $O(n^2 \cdot 9^n)$ , where  $n = |\mathbf{HB}|$ . On the other hand, with ground resolution, the memory use is bounded by  $O(2^n)$ , which is the maximum size of  $P$ , and the time complexity is bounded by  $O(4^n)$ .
- **Corollary:** Given the set  $E$  of complete state transitions, which has the size  $O(2^n)$ , the complexity of  $\mathbf{LF1T}(E, \emptyset)$  with ground resolution is bounded by  $O(|E|^2)$ . On the other hand, the worst-case complexity of learning with naïve resolution is  $O(n^2 \cdot |E|^{4.5})$ .

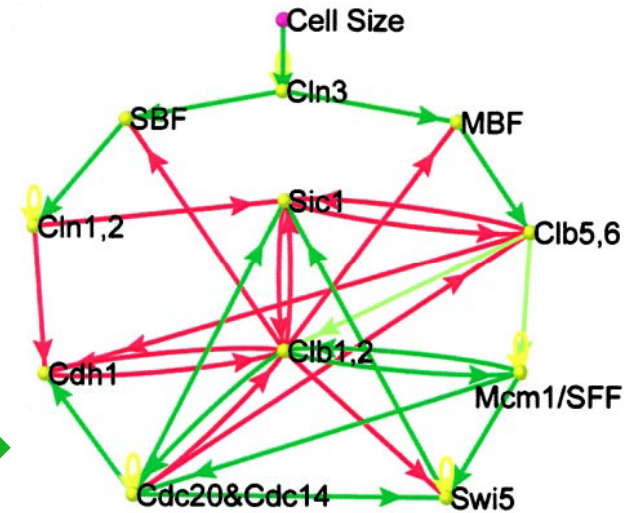
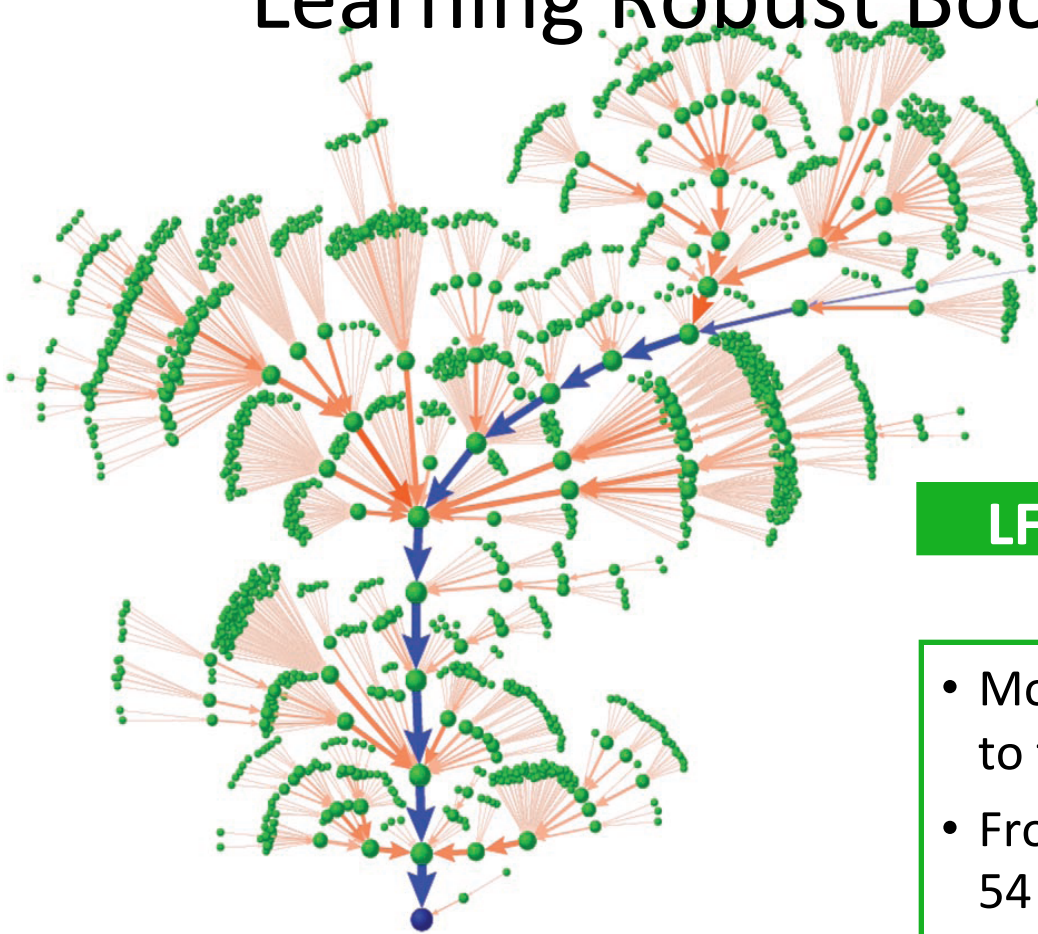
# Learning Boolean Networks

- Benchmarks of Boolean networks are taken from (Dubrova and Teslenko, 2011).
- All possible 1-step state transitions of  $N$  from all  $2^{|HB|}$  possible initial states  $I^0$ 's are computed from the benchmarks by firstly computing all stable models of  $\tau(N) \cup I^0$  using the answer-set solver **clasp**, then by running LF1T with these state transitions.
- Environment: Intel Core I7 (3610QM, 2.3GHz). Time limit: 1 hour.
- Boosting is effective to reduce the size/number of rules.

**Table 3** Learning time of LF1T for Boolean networks up to 15 nodes

Name	# nodes	# $\times$ length of attractor	# rules (org./LFIT)	Naïve	Ground
<i>Arabidopsis thaliana</i>	15	$10 \times 1$	28 / 241	T.O.	13.825s
Budding yeast	12	$7 \times 1$	54 / 54	6m01s	0.820s
Fission yeast	10	$13 \times 1$	23 / 24	5.208s	0.068s
Mammalian cell	10	$1 \times 1, 1 \times 7$	22 / 22	5.756s	0.076s

# Learning Robust Boolean Networks



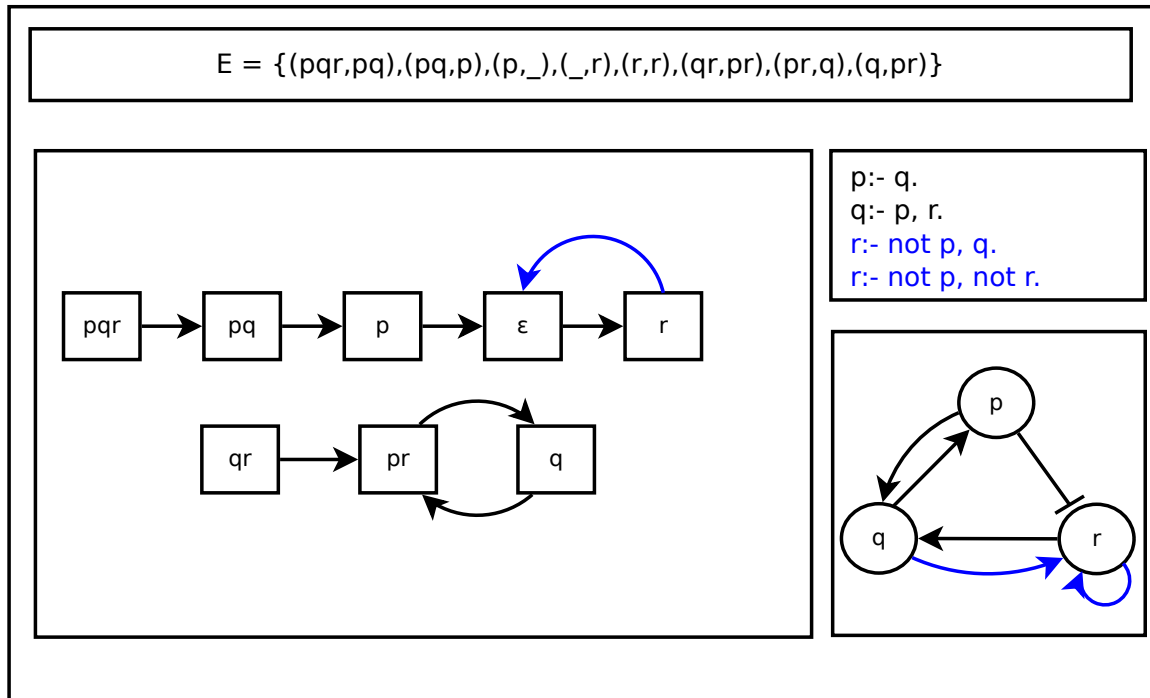
Li, F. *et al.*: The yeast cell-cycle network is robustly designed, *PNAS*, 101(14), 2004.

- Most transitions from  $2^{12}$  states belong to the same *basin of attraction*.
- From this state transition, LFIT learned 54 state transition rules in 0.8 sec.
- An improved learning algorithm using BDD learned the same rules in 0.18 sec.

- Inoue, K., Ribeiro T., Sakama, C.: “Learning from Interpretation Transition”, *Machine Learning*, 94(1):51-79, 2014.
- Ribeiro, T., Inoue, K., Sakama, C.: “A BDD-Based Algorithm for Learning from Interpretation Transition”, *Post-Proc. ILP 2013, LNAI*, Vol.8812, pp.47-63, 2014.

# Demonstration

- Bottom-Up Algorithm (Version 1; 2014.03)



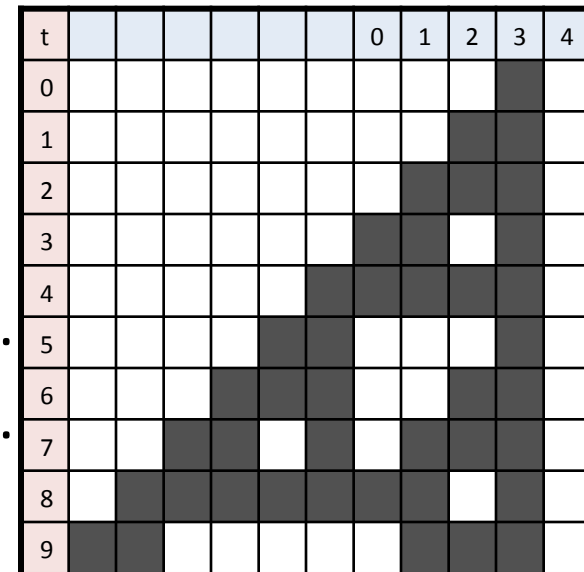
# Cellular Automata (CA)

- A CA consists of a regular grid of **cells**.
- A cell has a finite number of possible **states**.
- The state of each cell changes synchronously in discrete time steps according to local and identical **transition rules**.
- The state of a cell in the next time step is determined by its current state and the states of its surrounding cells (**neighborhood**).
- 2-state CA is regarded as an instance of Boolean networks.
- CA is a model of **emergence** and **self-organization**, which are two important features of the nature/real-life as a complex system.
- 1-dimensional 2-state CA can simulate Turing Machine (Wolfram).
- **Multi-state CA**: Disease Spreading Model—0 (healthy), 1 (infected), values in between (gradually more ill)

# Wolfram's Rule 110

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

- $c(x,t+1) \leftarrow c(x-1,t) \wedge c(x,t) \wedge \neg c(x+1,t).$
- $c(x,t+1) \leftarrow c(x-1,t) \wedge \neg c(x,t) \wedge c(x+1,t).$
- $c(x,t+1) \leftarrow \neg c(x-1,t) \wedge c(x,t) \wedge c(x+1,t).$
- $c(x,t+1) \leftarrow \neg c(x-1,t) \wedge c(x,t) \wedge \neg c(x+1,t).$
- $c(x,t+1) \leftarrow \neg c(x-1,t) \wedge \neg c(x,t) \wedge c(x+1,t).$



- Rule 110 is known to be Turing-complete.
- The logic program is *acyclic* (Apt & Bezem, 1990).

# Incorporating Background Theories

- Torus world: length 4
- $c(0, t) \leftarrow c(4, t)$ .
- $c(5, t) \leftarrow c(1, t)$ .

$c(3)$

→  $c(2), c(3)$

→  $c(1), c(2), c(3)$

→  $c(1), c(3), c(4)$  } attractor

→  $c(1), c(2), c(3) \rightarrow \dots$

t	(4)	1	2	3	4	(1)
0				■		
1			■	■		
2		■	■	■		■
3	■	■		■	■	■
4		■	■	■		■
5	■	■		■	■	■
6		■	■	■		■

learning rules:  $0 \rightarrow 1$  (4),  $1 \rightarrow 2$  (2),  $2 \rightarrow 3$  (2).

learning positive rules: (2), (2), (1).



# Incorporating Inductive Bias

- Bias I: The body of each rule exactly contains 3 neighbor literals.
- Bias II: The rules are universal for every time step and any position.
- Biases I and II imply that *anti-instantiation* (AI) can be applied immediately instead of least generalization.

Step	$I \rightarrow J$	Op.	Rule	ID	$P$
1	0010→0110	$R^3_2$	$c(2) \leftarrow \neg c(1) \wedge \neg c(2) \wedge c(3)$	1	
		AI(1)	$c(x) \leftarrow \neg c(x-1) \wedge \neg c(x) \wedge c(x+1)$	2	2
		$R^3_3$	$c(3) \leftarrow \neg c(2) \wedge c(3) \wedge \neg c(4)$	3	
		AI(3)	$c(x) \leftarrow \neg c(x-1) \wedge c(x) \wedge \neg c(x+1)$	4	2,4
2	0110→1110	$R^2_1$	$c(1) \leftarrow \neg c(0) \wedge \neg c(1) \wedge c(2)$	5	
		$R^{23}_2$	$c(2) \leftarrow \neg c(1) \wedge c(2) \wedge c(3)$	6	
		AI(6)	$c(x) \leftarrow \neg c(x-1) \wedge c(x) \wedge c(x+1)$	7	
		res(7,2)	$c(x) \leftarrow \neg c(x-1) \wedge c(x+1)$	8	4,8
		res(7,4)	$c(x) \leftarrow \neg c(x-1) \wedge c(x)$	9	8,9

# Incorporating Inductive Bias (Cont.)

Step	$I \rightarrow J$	Op.	Rule	ID	$P$
2	0110→1110	$R_{3}^{23}$	$c(3) \leftarrow c(2) \wedge c(3) \wedge \neg c(4)$	10	
		AI(10)	$c(x) \leftarrow c(x-1) \wedge c(x) \wedge \neg c(x+1)$	11	
		res(11,9)	$c(x) \leftarrow c(x) \wedge \neg c(x+1)$	12	8,9,12
3	1110→1011	$R_{1}^{01}$	$c(1) \leftarrow \neg c(0) \wedge c(1) \wedge c(2)$	13	
		$R_{4}^{34}$	$c(4) \leftarrow c(3) \wedge \neg c(4) \wedge c(5)$	14	
		AI(14)	$c(x) \leftarrow c(x-1) \wedge \neg c(x) \wedge c(x+1)$	15	
		res(15,8)	$c(x) \leftarrow \neg c(x) \wedge c(x+1)$	16	8,9,12,16

- $c(x,t+1) \leftarrow \neg c(x-1,t) \wedge c(x+1,t). \quad (8)$
- $c(x,t+1) \leftarrow \neg c(x-1,t) \wedge c(x,t). \quad (9)$
- $c(x,t+1) \leftarrow c(x,t) \wedge \neg c(x+1,t). \quad (12)$
- $c(x,t+1) \leftarrow \neg c(x,t) \wedge c(x+1,t). \quad (16)$

These are simpler than the original 5 rules, but still have one redundant rule.

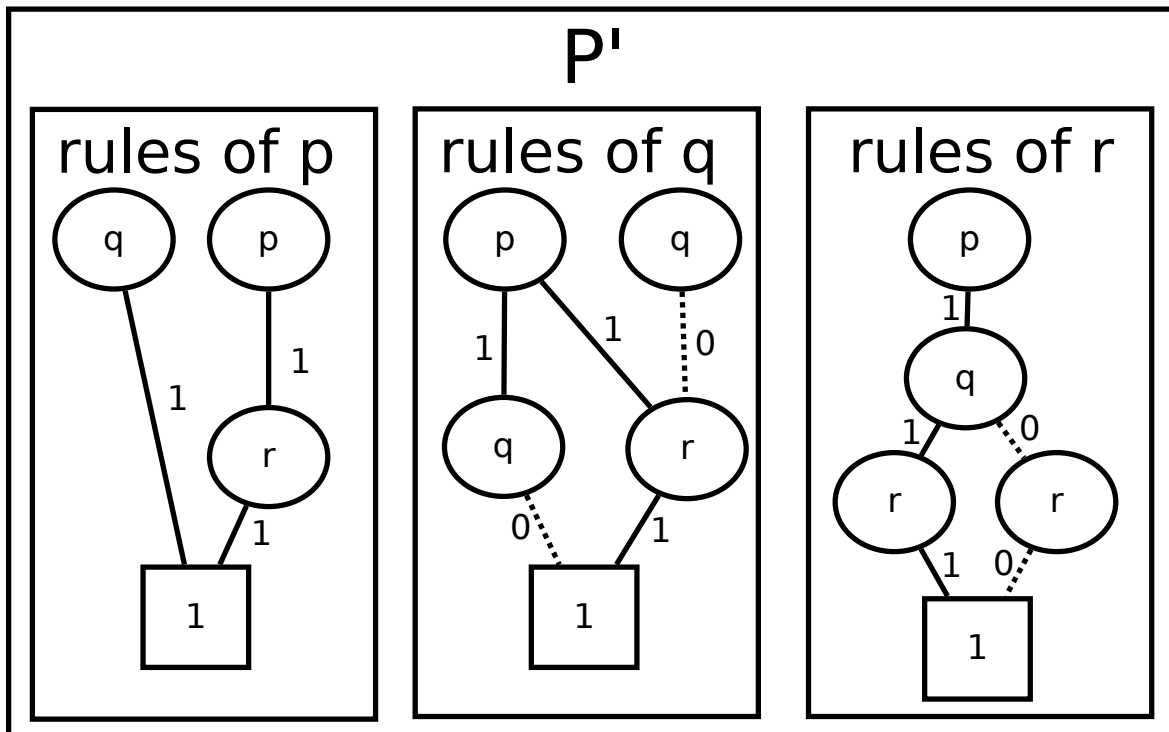
# LF1T-BDD

(Ribeiro, Inoue and Sakama, ILP 2013)

**Idea:** use **BDD** techniques to use less memory and learn faster.

**Contribution:** extend the scalability and the efficiency of LF1T.

- Naïve resolution  $\doteq$  symmetric reduction.
- New operation was devised for Ground resolution.



NLP P

$p \leftarrow q$

$p \leftarrow p \wedge r$

$q \leftarrow p \wedge \neg q$

$q \leftarrow p \wedge r$

$q \leftarrow \neg q \wedge r$

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

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

# Top-Down LF1T

(Ribeiro and Inoue, ILP 2014)

- **Idea:** Generate rules by specialization from the most general ones until no negative example is covered.
- **Merit:** Learn all minimal (*prime implicant*) rules.
  - Unique output, Efficient on real networks
- TD-LF1T starts with an initial program  $P_0 = \{a. \mid a \in \mathbf{HB}\}$ .
- For each transition  $(I, J) \in E$ , for each variable  $A$  that does **not** appear in  $J$ ,  $A$  is associated with the set of its *anti-support*
$$K'_A := I \cup \{\neg B \mid B \in \mathbf{HB} \setminus I\};$$
- Each rule with head  $A$  is then specialized by the rules each of which is formed by adding one literal from  $K'_A$  to its body.
- **Example:** from the state transition  $(bc, ac)$ , the rule  $b.$  is replaced by three rules:  $b \leftarrow a$ ;  $b \leftarrow \neg b$ ;  $b \leftarrow \neg c$ .

# Running example

$$E = \{ (abc, ab), (ab, a), (a, \epsilon), (\epsilon, c), (c, c), (bc, ac) (ac, b) (b, ac) \}$$

Init.	$abc \rightarrow ab$	$ab \rightarrow a$	$a \rightarrow \epsilon$	$\epsilon \rightarrow c$	$c \rightarrow c$
a.	a.	a.	$a \leftarrow \neg a.$	$a \leftarrow b.$	$a \leftarrow b.$
b.	b.	$b \leftarrow \neg a.$	$a \leftarrow b.$	$a \leftarrow c.$	$a \leftarrow a \wedge c.$
c.	$c \leftarrow \neg a.$	$b \leftarrow \neg b.$	$a \leftarrow c.$	<del><math>a \leftarrow \neg a \wedge b.</math></del>	<del><math>a \leftarrow b \wedge \epsilon.</math></del>
	$c \leftarrow \neg b.$	$b \leftarrow c.$	$b \leftarrow \neg a.$	<del><math>a \leftarrow \neg a \wedge \epsilon.</math></del>	$b \leftarrow \neg a \wedge b.$
	$c \leftarrow \neg c.$	$c \leftarrow \neg a.$	$b \leftarrow c.$	$b \leftarrow c.$	$b \leftarrow a \wedge c.$
		<del><math>c \leftarrow \neg b.</math></del>	<del><math>b \leftarrow \neg a \wedge \neg b.</math></del>	$b \leftarrow \neg a \wedge b.$	$b \leftarrow b \wedge c.$
		<del><math>\epsilon \leftarrow \neg a \wedge \neg \epsilon.</math></del>	<del><math>b \leftarrow \neg b \wedge \epsilon.</math></del>	<del><math>b \leftarrow \neg a \wedge \epsilon.</math></del>	$c \leftarrow \neg a.$
		<del><math>\epsilon \leftarrow \neg b \wedge \neg \epsilon.</math></del>	$c \leftarrow \neg a.$	$c \leftarrow \neg a.$	$c \leftarrow \neg b \wedge c.$
		<del><math>\epsilon \leftarrow \neg a \wedge \neg b.</math></del>	<del><math>\epsilon \leftarrow \neg a \wedge \neg b.</math></del>	$c \leftarrow \neg b \wedge c.$	
		$c \leftarrow \neg b \wedge c.$			

$bc \rightarrow ac$	$ac \rightarrow b$	$b \rightarrow ac$
$a \leftarrow b.$	$a \leftarrow b.$	$a \leftarrow b.$
$a \leftarrow a \wedge c.$	<del><math>a \leftarrow a \wedge b \wedge \epsilon.</math></del>	$b \leftarrow a \wedge c.$
$b \leftarrow a \wedge c.$	$b \leftarrow a \wedge c.$	$c \leftarrow \neg a.$
$b \leftarrow \neg a \wedge b \wedge \neg c.$	$b \leftarrow \neg a \wedge b \wedge \neg c.$	
<del><math>b \leftarrow a \wedge b \wedge \epsilon.</math></del>	$c \leftarrow \neg a.$	
$c \leftarrow \neg a.$	<del><math>\epsilon \leftarrow \neg a \wedge \neg b \wedge \epsilon.</math></del>	
$c \leftarrow \neg b \wedge c.$		
		$b \leftarrow \neg a \wedge b \wedge \neg c.$
		is removed because it cannot be specialized

# Comparison of 3 algorithms

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

## Properties of the different LF1T algorithms

Algorithm	Complexity		Scalability	Performances		Output	Variables	Transitions
	Run Time	Memory	# Variables	Run Time	Memory	Sensitivity	Ordering	Ordering
Naïve	$O(n^2 \cdot 9^n)$	$O(n \cdot 3^n)$	12	--	---	-	-	-
Ground	$O(4^n)$	$O(2^n)$	16	++	++	--	--	--
Ground-BDD	$O(4^n)$	$O(2^n)$	23	+++	+++	--	--	--
Full Naïve	$O(n^3 \cdot 9^n)$	$O(n^2 \cdot 3^n)$	10	---	---	+++	+++	+++
Full Ground	$O(n \cdot 4^n)$	$O(2^n)$	16	--	-	+++	+++	---
Least Specialization	$O(n \cdot 4^n)$	$O(2^n)$	23	++	++	+++	+++	---

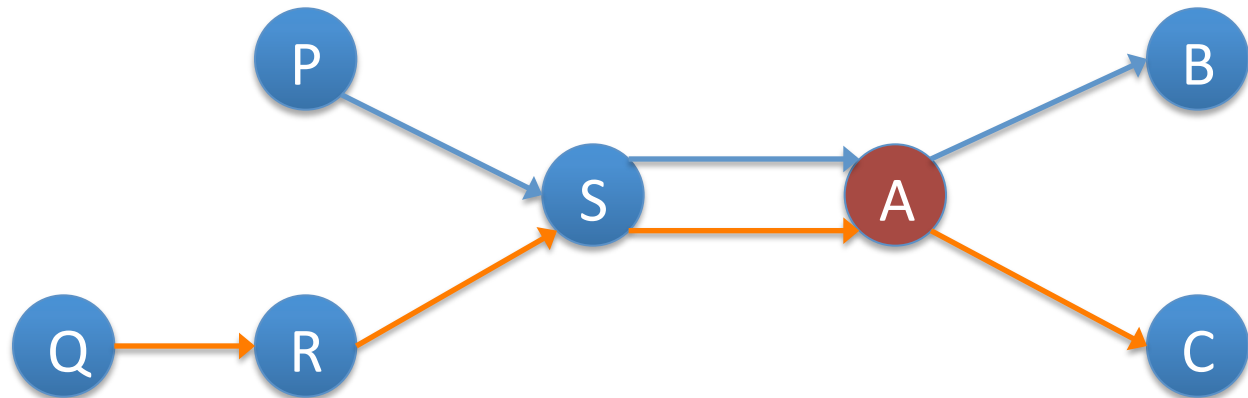
# Contents

- **Motivation**
- **Principles**
- **Extensions:**
  - LFkT (Learning Markov( $k$ ) Systems)
  - Multi-Valued LFIT
  - Asynchronous LFIT
  - Probabilistic LFIT
- **Applications**
- **Ongoing Work**

# What is the Nondeterminism? (A possible answer)

*When different transitions are possible, starting from a single state, the pathway followed depends on the values of the corresponding transition **delays**.*

(Thomas and D'Ari, 1990)



t = -2	t = -1	t = 0	t = 1	t = 2
--	P	S	A	B
Q	R	S	A	C



# Markov( $k$ ) System

(Ribeiro, Magnin, Inoue and Sakama, FiBioE 2015)

- Let  $\mathbf{HB}$  be the Herbrand base of a program  $P$  and  $k$  a natural number. The *timed Herbrand base*  $\mathbf{HB}_k$  of  $P$  (with period  $k$ ) is:

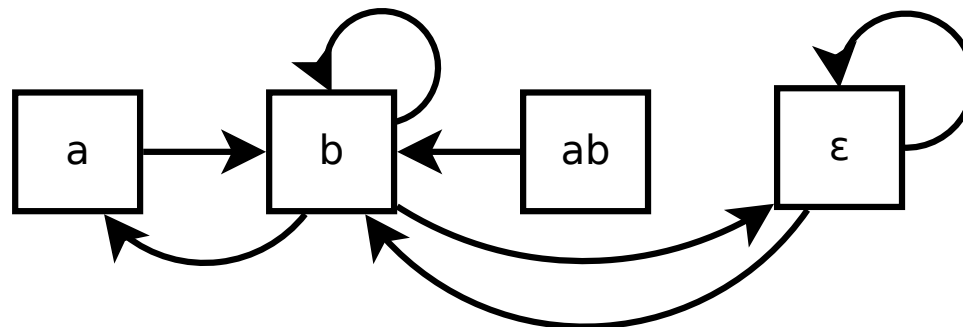
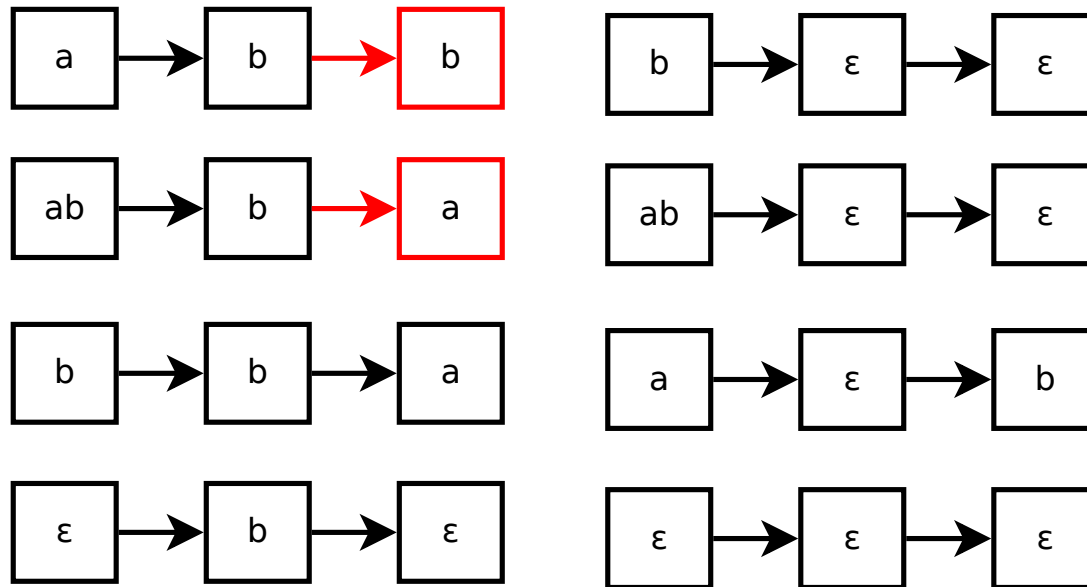
$$\mathbf{HB}_k = \bigcup_{i=1}^k \{v_{t-i} \mid v \in \mathbf{HB}\}$$

where  $t$  is a constant term (the current time step).

- A *Markov( $k$ ) system*  $S$  with respect to  $P$  is a logic program satisfying that, for any rule  $R \in S$ ,  $h(R) \in \mathbf{HB}$  and all atoms appearing in  $b(R)$  belong to  $\mathbf{HB}_k$ .
- Example:**  $S = \{ a \leftarrow b_t \wedge b_{t-1}; b \leftarrow a_{t-2} \wedge \neg b_{t-2} \}$  is a Markov(2) system.

# Markov(2) system

- $S = \{a \leftarrow b_t \wedge b_{t-1}; b \leftarrow a_{t-2} \wedge \neg b_{t-2}\}$



# (Top-down) LFkT

**Input:** A set of traces of executions  $O$  of a Markov( $k$ ) system  $S$ . Note that the optimum delay is unknown at this point.

**Step 1:** Initialize a logic program  $P$  with the fact rules.

**Step 2:** Pick a trace  $T$  from  $O$  and update the delay  $k$  accordingly:

- a. Initialize a program with the fact rules for each new delay  $k$ .
- b. Revise these logic programs with all previous traces.

**Step 3:** Convert the trace into interpretation transitions and revise the programs using least specialization.

**Step 4:** If there is a remaining trace in  $O$ , go back to step 2.

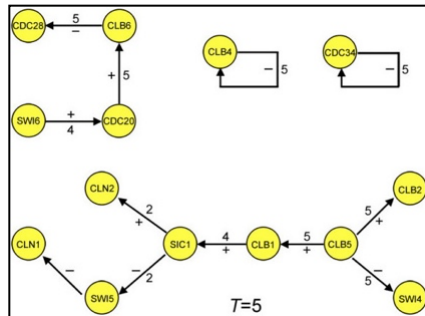
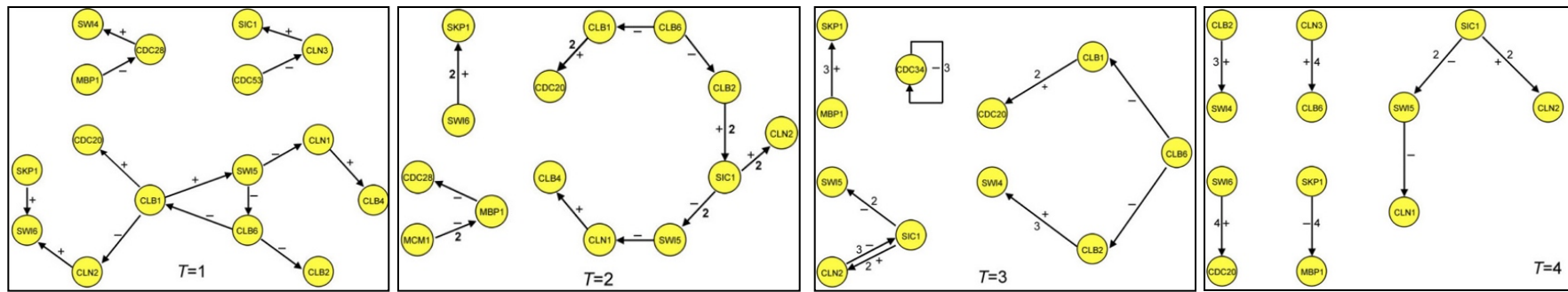
**Step 5:** Merge all logic programs into one by avoiding rule subsumption.

**Step 6:** Remove all rules that are not necessary to explain  $O$ .

**Output:** A set of rules with the optimum delay which realizes  $O$ .

# Experiments of a Markov(5) system

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



# Traces	Run time (# of seconds)				
	k=1	k=2	k=3	k=4	k=5
10	0.23s	0.27s	<b>0.16s</b>	0.28s	0.31s
100	1.87s	2.49s	<b>1.63s</b>	2.37s	2.84s
1 000	15.3s	18.5s	<b>13.3s</b>	20.1s	23.8s
10 000	<b>146s</b>	218s	147s	201s	234s
100 000	2 177s	2 577s	<b>1 643s</b>	1 764s	2243s
1 000 000	27 768s	22 517	12 384	15 670	20 413

- LfKT input goes from 10 to  $10^6$  traces.

# Multi-Valued LFIT

(Ribeiro, Magnin, Inoue and Sakama, ICMLA 2015)

- Atoms of a logic program to the form  $var^{val}$ . We consider a multi-valued logic program as a set of rules of the form:

$$v^{val} \leftarrow v_1^{val_1} \wedge \dots \wedge v_n^{val_n}$$
$$(v=val \leftarrow v_1=val_1 \wedge \dots \wedge v_n=val_n)$$

where  $v^{val}$  and  $v_i^{val_i}$  are atoms ( $n \geq 0$ ).

- **Example:**  $P = \{a^1 \leftarrow a^0 \wedge b^2; b^1 \leftarrow a^1 \wedge b^0\}$ .
- A *multi-valued interpretation*  $I$  is a set of assignments such that  $I$  contains one and only one ground atom  $v^{val}$  for any  $v \in \mathbf{V}$ .
- Note: **no negation:** definite programs (or equational theory)
- Generalization and specialization can be redefined accordingly.

# Other extensions

- From synchronous to asynchronous updates
- From deterministic to probabilistic and nondeterministic transitions (Martínez *et al.*, ICLP 2015; ICAPS 2016)
- And their delayed and/or multi-valued versions
  
- Open Question: *When there are incompatible transitions, which models should be selected?*
  - delay, synchronicity, noise, nondeterminism, uncertainty, etc.

# Contents

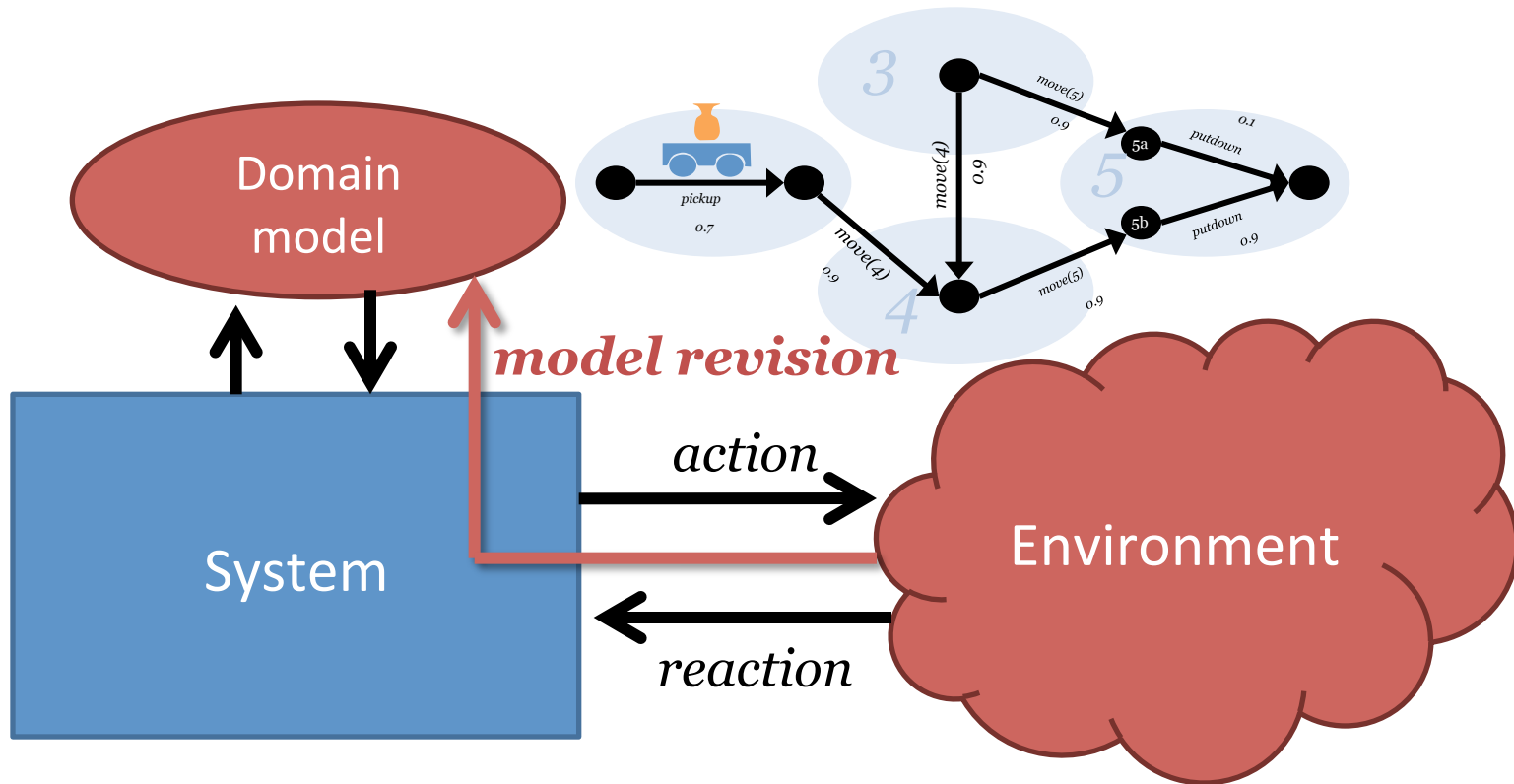
- **Motivation**
- **Principles**
- **Extensions**
- **Applications:**
  - Biology
  - Cellular Automata
  - Robotics (learning action rules)
  - AGI (learning agents' logics)
- **Ongoing Work**

# Two types of changes

- Endogenous change:
  - natural growth
  - physical dynamics
  - biological development
  - internal change in the closed system
- Exogenous change:
  - action
  - external event
  - enzymatic reaction
  - decision/choice with free will



# Revising Plans in Adaptive Systems

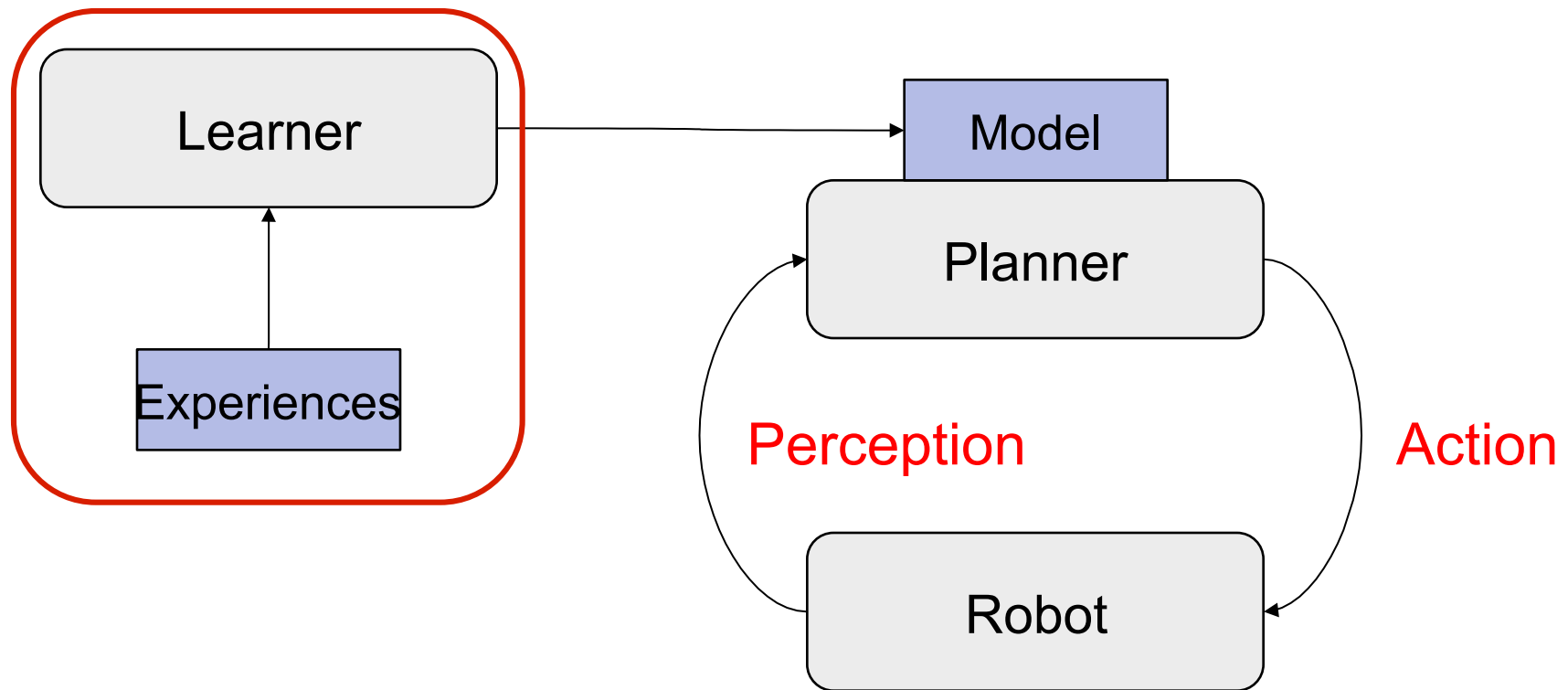


Behavioural model revision through **probabilistic rule learning**  
– action rules are learned from logs of trails using ASP

- Sykes, D., Corapi, D., Magee, J., Kramer, J., Russo, A., Inoue, K.: “Learning revised models for planning in adaptive systems”, *ICSE 2013*: 63-71.

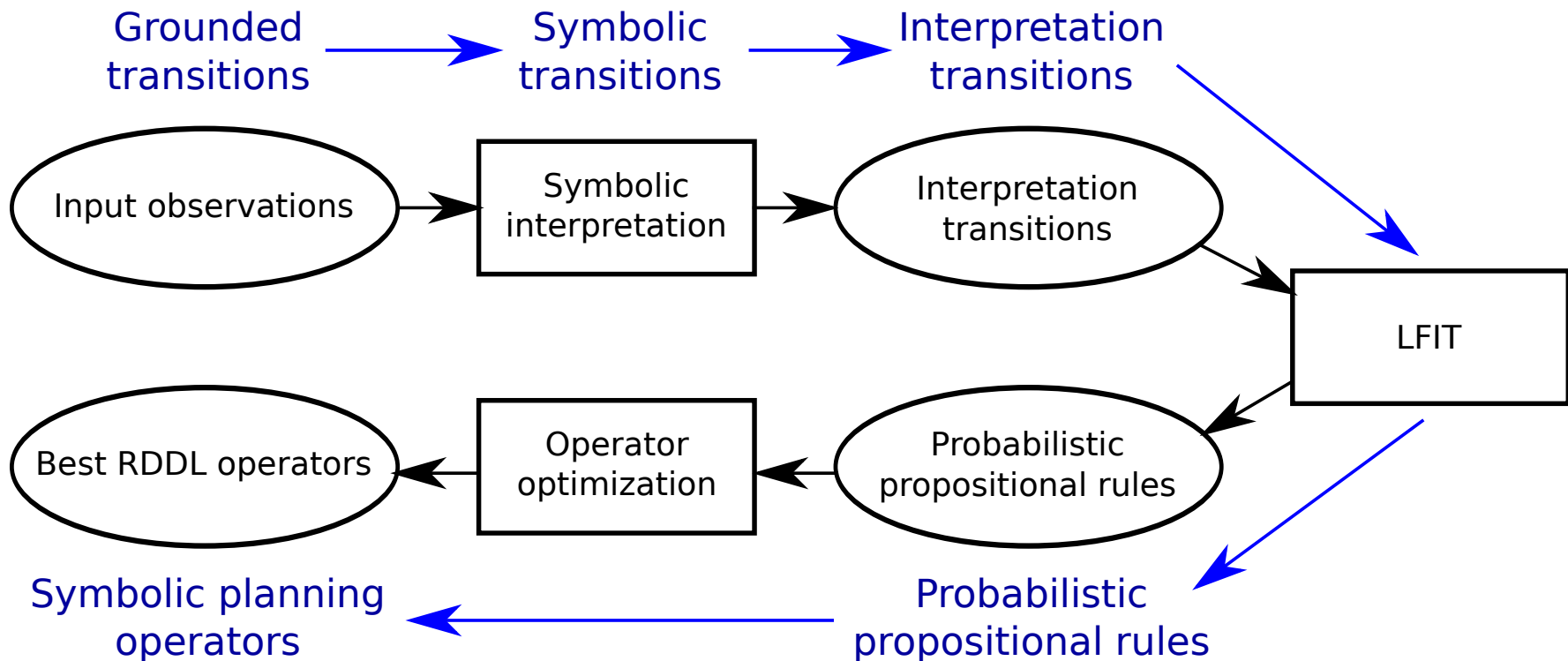
# Learning Uncertain Actions in Stochastic Domains for Robot Planning

(joint work with David Martínez and his colleagues in CSIC-UPC)



# Learning Uncertain Actions through LFIT

(Martínez *et al.*, ICLP 2015; ICAPS 2016)



# Action Learning with LFIT

- Learn from transitions ( $state_t$ ,  $action_t$ ,  $state_{t+1}$ )
- Learned rule:

$p = 0.3$  :  $robot-at(X,Z)$   $\leftarrow$   $robot-at(X,Y)$   $\wedge$   $\neg$   $obstacle-at(X,Z)$   $\wedge$   $move-north$

Probability      Head = Effect      Body = Preconditions      +      Action



# IPPC 2014 Domains (Martínez *et al.*, ICAPS 2016)

- **Triangle Tireworld.** (Easiest) Uncertain effects, no exogenous effects, 5 predicates, 3 actions, 7 operators with max 2 terms.
- **Crossing Traffic.** (Intermediate) Uncertain effects, exogenous effects, 8 predicates, 4 actions, 6 operators with max 3 terms.
- **Elevators.** (Hard) Uncertain effects, exogenous effects, 10 predicates, 4 actions, 17 operators with max 3 terms.

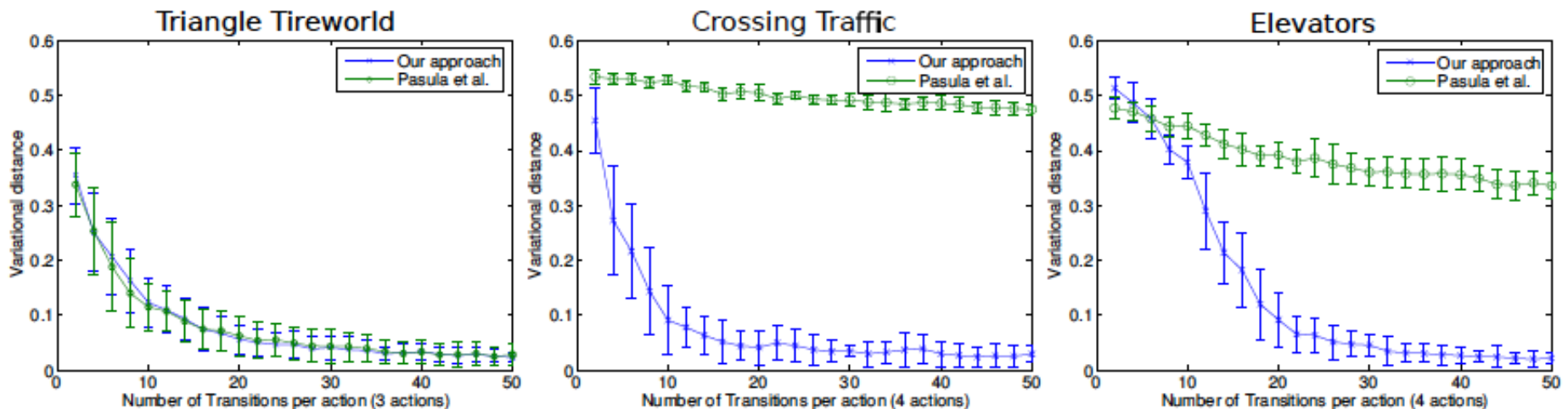
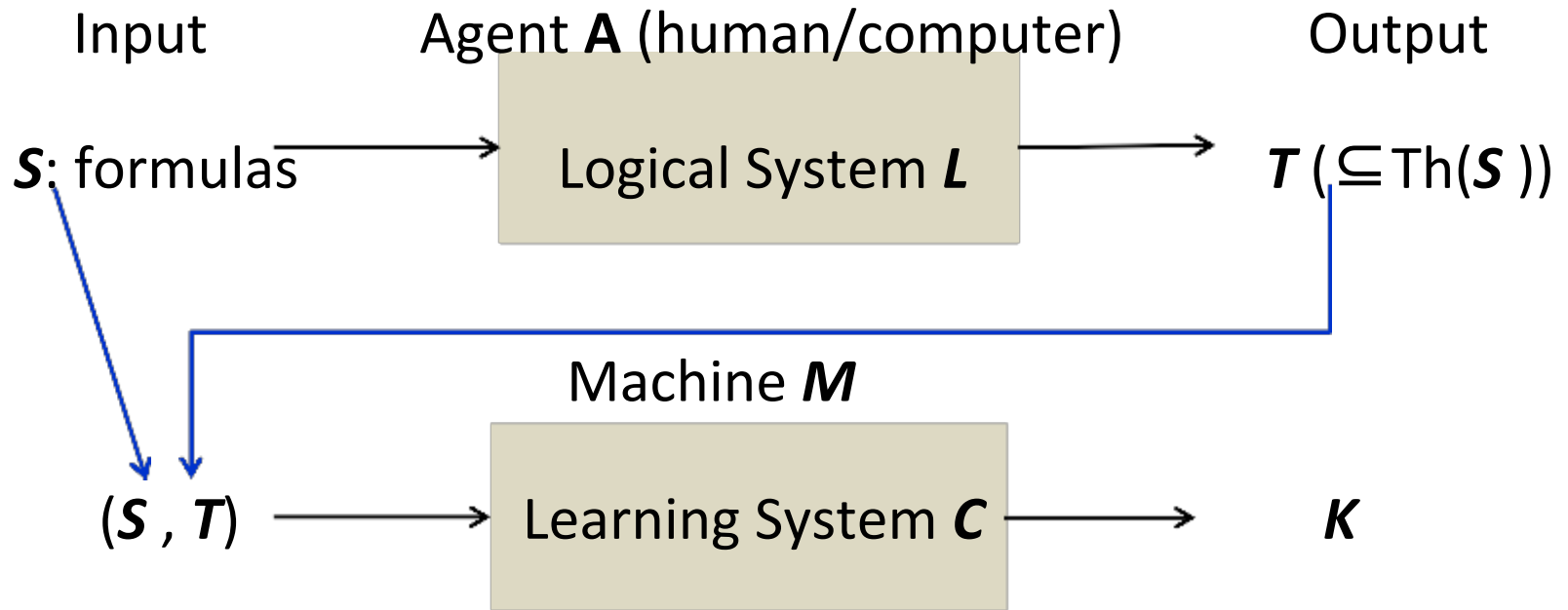


Figure 5: Comparison with Pasula et al. The results shown are the means and standard deviations obtained from 50 runs. The evaluation was done with 5000 random transitions. **Left:** Triangle Tireworld ( $\alpha = 0.02$ ,  $\epsilon = 0.1$ ,  $\omega = 2$ ,  $\delta = 0$ ,  $\kappa = \infty$ , no tree). **Center:** Crossing Traffic ( $\alpha = 0.025$ ,  $\epsilon = 0.1$ ,  $\omega = 3$ ,  $\delta = 0.05$ ,  $\kappa = 500$ , tree). **Right:** Elevators ( $\alpha = 0.015$ ,  $\epsilon = 0.1$ ,  $\omega = 3$ ,  $\delta = 0.05$ ,  $\kappa = 500$ , tree).

# “Can Machines Learn Logics?”



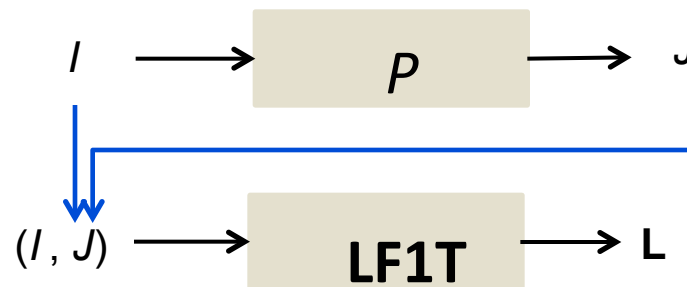
- Given input  $(\mathbf{S}, \mathbf{T})$ , a machine **M** produces an axiomatic system **K**.
- It is possible to learn meta-level one-step deduction rules, e.g., MP.
- Would be applied to learning abduction and other non-standard logics.

➤ Sakama, C., Inoue, K.: “Can Machines Learn Logics?”, *8<sup>th</sup> Int’l Conf. Artificial General Intelligence (AGI 2015)*.

# Learning Deduction Rules by LF1T

(Sakama, Ribeiro and Inoue, ILP 2015)

- We assume a deduction system  $L$  represented by a **metalogic program**  $P$  that provides transitions  $(I, J)$  satisfying  $J = T_P(I)$ .
- Given  $(I, J)$  as an input, our goal is to examine whether **LF1T** can reproduce correct inference rules of  $L$  represented by meta-rules in  $P$ .



# Example

- Given the Herbrand base:

$\mathbf{B} = \{ \text{hold}(p), \text{hold}(q), \text{hold}(r), \text{hold}(p \rightarrow r) \}$ ,

a rule with  $\text{hold}(r)$  in the head is constructed as follows.

- Step 0:** **LF1T** starts with the most general rule:

$$\text{hold}(r) \leftarrow \quad (1)$$

- Step 1:** The transition  $(\{\}, \{\})$  is given. (1) is inconsistent with this (namely,  $\{\}$  should produce  $\{\text{hold}(r)\}$  under (1)), so (1) is minimally specialized by introducing an atom from **B**:

$$\text{hold}(r) \leftarrow \text{hold}(p) \quad (2)$$

$$\text{hold}(r) \leftarrow \text{hold}(q) \quad (3)$$

$$\text{hold}(r) \leftarrow \text{hold}(r) \quad (4)$$

$$\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r) \quad (5)$$



# Example

- **Step 2:** The transition  $(\{\text{hold}(p)\}, \{\text{hold}(p)\})$  is given.

$$\text{hold}(r) \leftarrow \text{hold}(p) \quad (2)$$

is inconsistent with this, so (2) is specialized into

$$\text{hold}(r) \leftarrow \text{hold}(p), \text{hold}(q)$$

$$\text{hold}(r) \leftarrow \text{hold}(p), \text{hold}(r)$$

$$\text{hold}(r) \leftarrow \text{hold}(p), \text{hold}(p \rightarrow r)$$

These rules are respectively subsumed by

$$\text{hold}(r) \leftarrow \text{hold}(q) \quad (3)$$

$$\text{hold}(r) \leftarrow \text{hold}(r) \quad (4)$$

$$\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r) \quad (5)$$

hence removed. As a result, (3), (4) and (5) remain.

# Example

- **Step 3:** The transition  $(\{\text{hold}(q)\}, \{\text{hold}(q)\})$  is given.

$$\text{hold}(r) \leftarrow \text{hold}(q) \quad (3)$$

is inconsistent with this, so (3) is specialized into

$$\text{hold}(r) \leftarrow \text{hold}(q), \text{hold}(p) \quad (6)$$

$$\text{hold}(r) \leftarrow \text{hold}(q), \text{hold}(r)$$

$$\text{hold}(r) \leftarrow \text{hold}(q), \text{hold}(p \rightarrow r)$$

The last two rules are respectively subsumed by

$$\text{hold}(r) \leftarrow \text{hold}(r) \quad (4)$$

$$\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r) \quad (5)$$

and removed. As a result, (4), (5) and (6) remain.

input	output
$(\{\}, \{\})$	$\text{hold}(r) \leftarrow \text{hold}(p)$ $\text{hold}(r) \leftarrow \text{hold}(q)$ $\text{hold}(r) \leftarrow \text{hold}(r)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r)$
$(\{\text{hold}(p)\}, \{\text{hold}(p)\})$	<del><math>\text{hold}(r) \leftarrow \text{hold}(p)</math></del> $\text{hold}(r) \leftarrow \text{hold}(q)$ $\text{hold}(r) \leftarrow \text{hold}(r)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r)$
$(\{\text{hold}(q)\}, \{\text{hold}(q)\})$	<del><math>\text{hold}(r) \leftarrow \text{hold}(q)</math></del> $\text{hold}(r) \leftarrow \text{hold}(r)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r)$ $\text{hold}(r) \leftarrow \text{hold}(p), \text{hold}(q)$
$(\{\text{hold}(p \rightarrow r)\}, \{\text{hold}(p \rightarrow r)\})$	$\text{hold}(r) \leftarrow \text{hold}(r)$ <del><math>\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r)</math></del> $\text{hold}(r) \leftarrow \text{hold}(p), \text{hold}(q)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(p)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(q)$
$(\{\text{hold}(p), \text{hold}(q)\}, \{\text{hold}(p), \text{hold}(q)\})$	$\text{hold}(r) \leftarrow \text{hold}(r)$ <del><math>\text{hold}(r) \leftarrow \text{hold}(p), \text{hold}(q)</math></del> $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(p)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(q)$
$(\{\text{hold}(p \rightarrow r), \text{hold}(q)\}, \{\text{hold}(p \rightarrow r), \text{hold}(q)\})$	$\text{hold}(r) \leftarrow \text{hold}(r)$ $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(p)$ <del><math>\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(q)</math></del>
$(\{\text{hold}(p \rightarrow r), \text{hold}(p)\}, \{\text{hold}(p), \text{hold}(r)\})$	$\text{hold}(r) \leftarrow \text{hold}(r)$ :Repetition $\text{hold}(r) \leftarrow \text{hold}(p \rightarrow r), \text{hold}(p)$ :Modus Ponens

# Experimental Results

- Given  $\mathbf{B} = \{ \text{hold}(p), \text{hold}(\neg p), \text{hold}(q), \text{hold}(\neg q), \text{hold}(p \rightarrow q), \text{hold}(q \rightarrow r), \text{hold}(p \rightarrow r) \}$ , **LF1T** produces:
  - $\text{hold}(\neg p) \leftarrow \text{hold}(\neg q) \wedge \text{hold}(p \rightarrow q)$  : **Modus Tollens**
  - $\text{hold}(p \rightarrow r) \leftarrow \text{hold}(p \rightarrow q) \wedge \text{hold}(q \rightarrow r)$  : **Hypothetical Syllogism**
- Given  $\mathbf{B} = \{ \text{hold}(p), \text{hold}(\neg p), \text{hold}(q), \text{hold}(\neg q), \text{hold}(p \vee q), \text{hold}(\neg p \vee \neg q), \text{hold}(r \vee s), \text{hold}(\neg r \vee \neg s), \text{hold}(p \rightarrow r), \text{hold}(q \rightarrow s) \}$ , **LF1T** produces:
  - $\text{hold}(p) \leftarrow \text{hold}(p \vee q) \wedge \text{hold}(\neg q)$  : **Disjunctive Syllogism**
  - $\text{hold}(r \vee s) \leftarrow \text{hold}(p \vee q) \wedge \text{hold}(p \rightarrow r) \wedge \text{hold}(q \rightarrow s)$   
: **Constructive Dilemma**
  - $\text{hold}(\neg p \vee \neg q) \leftarrow \text{hold}(\neg r \vee \neg s) \wedge \text{hold}(p \rightarrow r) \wedge \text{hold}(q \rightarrow s)$   
: **Destructive Dilemma**
- **Given a transition  $(I, J) = (\{\text{hold}(p \rightarrow q), \text{hold}(q)\}, \{\text{hold}(p)\})$ , **LF1T** produces**
- $\text{hold}(p) \leftarrow \text{hold}(q) \wedge \text{hold}(p \rightarrow q)$   
: **Fallacy of Affirming the Consequence** (a rule for **Abduction**)

# Contents

- **Motivation**
- **Principles**
- **Extensions**
- **Applications**
- **Ongoing Work:**
  - DREAM Challenges
  - NN-LFIT

# Scientific Challenge

(Ongoing work with Morgan Magnin, Tony Ribeiro, Olivier Roux)

## Goals:

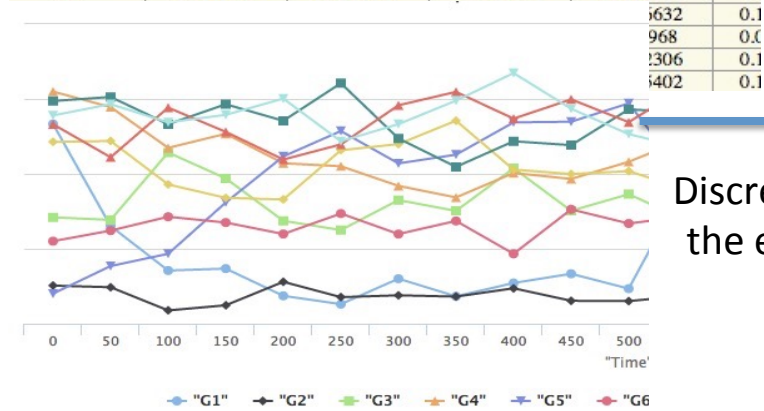
- Build **predictive** dynamic **models** from **time series data**
- Assess the **efficiency** of the approach by **taking part to an open Machine Learning challenge**.
- Motto: **Logical modeling** has its own merits compared with continuous approaches (e.g., ODE)

## Underlying questions:

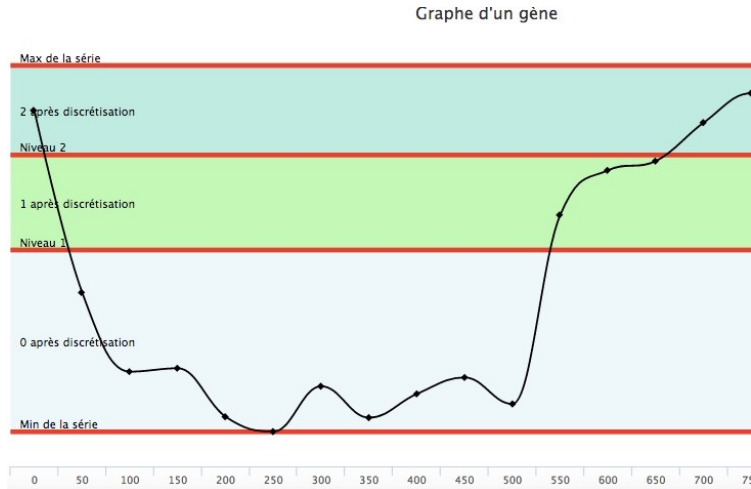
- What would be **efficient algorithms** for **discretization** and **learning**?
- What is a *good* predictive model?
- How to validate the **benefits** of the algorithms on "**real-life**" data?

# Abstraction of systems to build qualitative dynamic models

0	0.6665114	0.1272186	0.3550646	0.7745716	0.1
50	0.3257748	0.1218223	0.3464115	0.7229108	0.1
100	0.1775012	0.0443587	0.5712888	0.586828	0.2
150	0.1838851	0.0615354	0.4849771	0.6338205	0
200	0.0930693	0.1398431	0.3435551	0.5354375	0.4
250	0.0653015	0.0886361	0.3124305	0.5255509	0.6
300	0.1502136	0.094808	0.4126412	0.4599706	0.4
350	0.0913323	0.090451	0.3764574	0.4214017	0.4
400	0.1358562	0.1182462	0.5188737	0.5030134	0.6
450	0.1667014	0.0765614	0.3768678	0.4831358	0.6
500	0.1170103	0.075504	0.4322347	0.5403027	0.7
550	0.4708819	0.093721	0.3453156	0.618386	0.1
600	0.5545089	0.1443166	0.4763379	0.6233838	0.3
650	0.5717737	0.1580246	0.352766	0.6429242	0.2
700	0.6437458	0.0906398	0.3817925	0.7448212	0.1
750	0.6988616	0.0959292	0.4025808	0.6477586	0.1
800	0.6002293	0.1251139	0.3520098	0.7884273	0.1



Discretization of the expression



LFIT

```
ARNTE(0,T) += AMRE(0,T-1);
ARNTE(0,T) += CLOCK(0,T-1);
ARNTE(0,T) += CRF1(0,T-1); CRF2(0,T-1);
ARNTE(0,T) += CRF1(0,T-1); NRDI1(1,T-1);
ARNTE(0,T) += CRF1(0,T-1); NRDI1(0,T-1);
ARNTE(0,T) += CRF1(0,T-1); PER1(1,T-1);
ARNTE(0,T) += CRF1(0,T-1); PER1(2,T-1);
ARNTE(0,T) += CRF1(0,T-1); PER2(2,T-1);
ARNTE(0,T) += CRF1(0,T-1); PER2(3,T-1);
ARNTE(0,T) += CRF1(0,T-1); PPARA(1,T-1);
ARNTE(0,T) += CRF1(0,T-1); PPARA(0,T-1);
ARNTE(0,T) += CRF1(1,T-1); CRF2(0,T-1);
ARNTE(0,T) += CRF1(1,T-1); NRDI1(2,T-1);
ARNTE(0,T) += CRF1(1,T-1); NRDI1(0,T-1);
ARNTE(0,T) += CRF1(1,T-1); PER1(0,T-1);
ARNTE(0,T) += CRF1(1,T-1); PER1(2,T-1);
ARNTE(0,T) += CRF1(1,T-1); PER2(2,T-1);
ARNTE(0,T) += CRF1(1,T-1); PER2(3,T-1);
ARNTE(0,T) += CRF1(1,T-1); PPARA(2,T-1);
ARNTE(0,T) += CRF1(2,T-1); CRF2(0,T-1);
ARNTE(0,T) += CRF1(2,T-1); NRDI1(1,T-1);
ARNTE(0,T) += CRF1(2,T-1); NRDI1(0,T-1);
ARNTE(0,T) += CRF1(2,T-1); PER1(0,T-1);
ARNTE(0,T) += CRF1(2,T-1); PER1(2,T-1);
ARNTE(0,T) += CRF1(2,T-1); PER2(2,T-1);
ARNTE(0,T) += CRF1(2,T-1); PER2(3,T-1);
ARNTE(0,T) += CRF1(2,T-1); PPARA(2,T-1);
ARNTE(0,T) += CRF1(3,T-1); CRF2(2,T-1); PER1(1,T-1); PER2(0,T-1);
ARNTE(0,T) += CRF1(3,T-1); CRF2(2,T-1); PER1(1,T-1); PPARA(1,T-1);
ARNTE(0,T) += CRF1(3,T-1); NRDI1(1,T-1);
ARNTE(0,T) += CRF1(3,T-1); NRDI1(2,T-1);
ARNTE(0,T) += CRF1(3,T-1); PER1(0,T-1);
ARNTE(0,T) += CRF1(3,T-1); PER2(0,T-1); PPARA(2,T-1);
ARNTE(0,T) += CRF2(1,T-1);
ARNTE(0,T) += NRDI1(3,T-1);
ARNTE(0,T) += PER1(3,T-1);
ARNTE(0,T) += PER2(1,T-1);
ARNTE(0,T) += PPARA(3,T-1);
ARNTE(1,T) += AMRE(0,T-1);
ARNTE(1,T) += CLOCK(0,T-1);
ARNTE(1,T) += CRF1(0,T-1); CRF2(0,T-1);
ARNTE(1,T) += CRF1(0,T-1); NRDI1(1,T-1);
ARNTE(1,T) += CRF1(0,T-1); NRDI1(0,T-1);
ARNTE(1,T) += CRF1(0,T-1); PER1(1,T-1);
ARNTE(1,T) += CRF1(0,T-1); PER1(2,T-1);
ARNTE(1,T) += CRF1(0,T-1); PER2(2,T-1);
ARNTE(1,T) += CRF1(0,T-1); PER2(3,T-1);
ARNTE(1,T) += CRF1(0,T-1); PPARA(1,T-1);
ARNTE(1,T) += CRF1(0,T-1); PPARA(0,T-1);
ARNTE(1,T) += CRF1(1,T-1); CRF2(0,T-1);
ARNTE(1,T) += CRF1(1,T-1); NRDI1(2,T-1);
ARNTE(1,T) += CRF1(1,T-1); NRDI1(0,T-1);
ARNTE(1,T) += CRF1(1,T-1); PER1(0,T-1);
ARNTE(1,T) += CRF1(1,T-1); PER1(2,T-1);
ARNTE(1,T) += CRF1(1,T-1); PER2(2,T-1);
ARNTE(1,T) += CRF1(1,T-1); PER2(3,T-1);
ARNTE(1,T) += CRF1(1,T-1); PPARA(2,T-1);
```

Logic program

**Time series data**  
Regulatory networks: up to 45.000 genes over 48 time points for every gene

# A wide range of possible models and inference algorithms

- **Expression discretization**
  - Boolean
  - Multi-valued
- **Time discretization**
  - Event-driven model
  - Discrete-time model
  - Dense-time model
- **Semantics of discrete transitions**
  - Synchronous
  - Partially synchronous (a set of transitions can be fired altogether)
  - Asynchronous (one at a time)
- **Determinism of discrete transitions** : yes or no
- **Markov property**: yes or no



# Two main targets

## DREAM Challenges data:

- Predict **steady states** and associated model (without signs):
  - DREAM4 (2009): **100 genes** over 21 time points [*synthetic* data]
  - DREAM5 (2010): **1000 genes** over 17 time points [*real* data]
- Predict **trajectories** and associated model:
  - DREAM8 (2013): 40 genes over 8 times points [*real* data]

**Circadian rhythm** (French ANR-funded project): predict **trajectories** and **associated model** (with nature of interactions)

**Input:** expression data (Delaunay et al.) [*real* data]

- **45.000 genes**
- 4 datasets (1 normal + 3 perturbations)
- 48 time points for every gene



*powered by Sage Bionetworks*

- Start: **2006**
- Periodicity: **Annual**
- **DREAM** = "Dialogue for Reverse Engineering Assessments and Methods": Reverse engineering for regulatory, signaling, and metabolic networks
- **Aim:** encourage the design of new efficient computational **models** and **methods** to analyze systems from biology
- URL: <http://dreamchallenges.org>
- A wide range of partners (IBM Research, Sage Bionetworks, ...)
- Main competitor: kaggle
- **Reward:** no financial prize, but publications in journals
- Main issue: how to rank methods?
- **Scoring committee** for each challenge
- **Evaluation criteria:** Predictive power with specific metrics: AUPvR, AUROC

# Prediction problems issued in DREAM

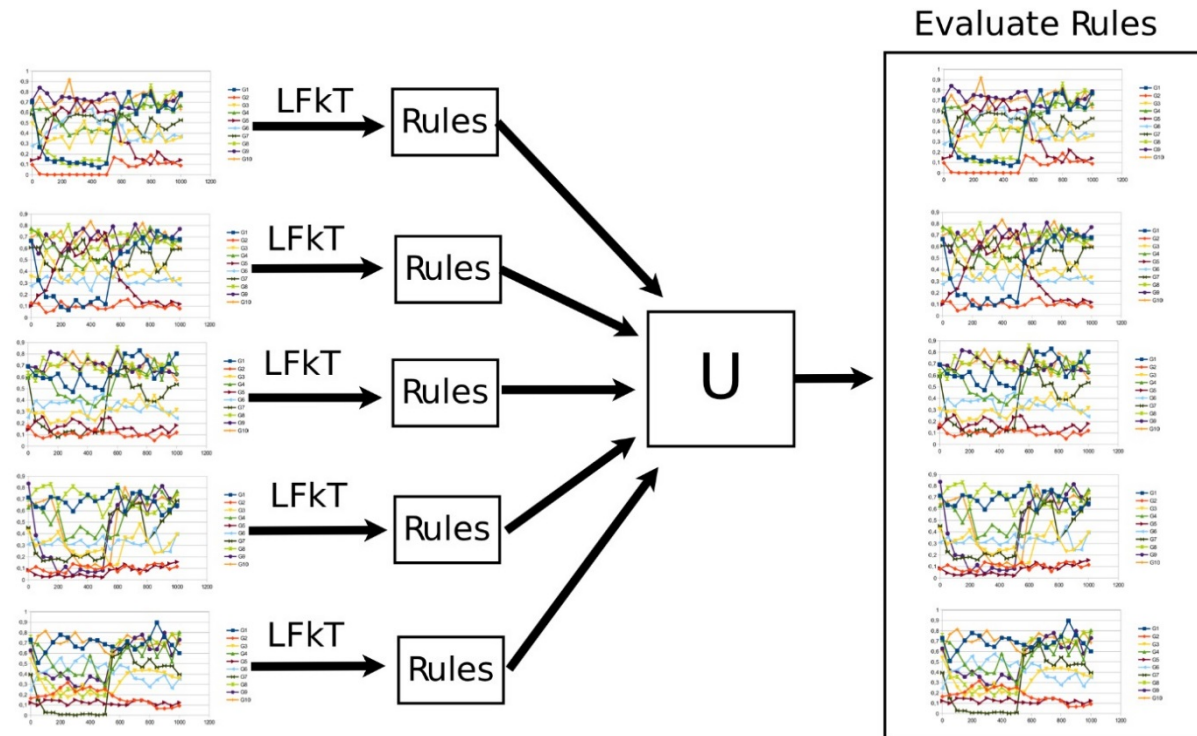
- **Structure** of the network
- **Behavioral prediction** with regard to specific conditions (e.g., knockouts) or initial states
  - Steady states
  - Trajectories
- Judges are **blind** to the approach that is used, i.e., only the prediction results counts... (at least at short-term perspective)
- **Computational time is not a criterion** that is assessed in this kind of challenge.
  - Most methods tend to lie in the range of **minutes** to **hours**.
  - But some others could take **up to 48h per gene**.

# About DREAM4 data

- Targeted systems (adapted from E. coli and yeast network):
  - 5 different systems each composed of 10 genes
  - 5 different systems composed of 100 genes
- Datasets available for **each system of 10 genes (resp. 100)**:
  - **5 (resp. 10) time series data with 21 time points corresponding to different perturbations**
  - **Steady state at wild type, i.e. 1 steady state**
  - Steady state after knocking out each gene, i.e. 10 steady states (resp. 100)
  - Steady state after knocking down each gene (transcription rate at 50%), i.e. 10 steady states (resp. 100)
  - Steady states after some random multifactorial perturbations, i.e. 10 steady states

# Learning experiments with LFkT

- Learn **independently** each series with LFkT
- **Evaluate rules on all series** (full cross-validation)



# Evaluation on DREAM4 data

- Goal: Given an **initial state** and {5 for networks of 10 genes; 20 for networks with 100 genes} different **conditions of dual genes to be knockout** simultaneously, predict **point attractors**
- **Evaluation of precision: mean square error** of the difference between predicted/expected values

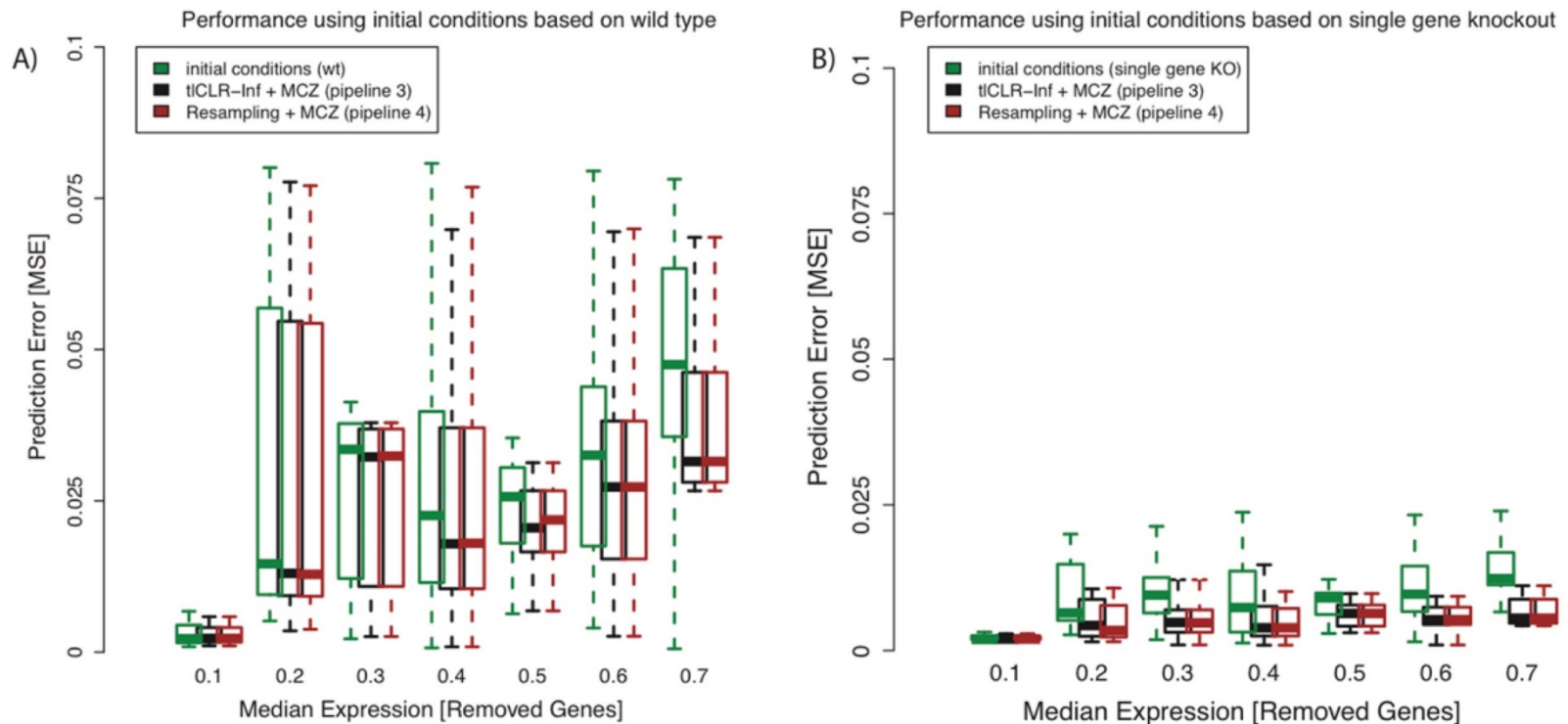
Benchmark	Number of genes	Run time	MSE
insilico_size100_1	100	51h	0.068
insilico_size100_2	100	47h	0.044
insilico_size100_3	100	59h	0.052
insilico_size100_4	100	50h	0.053
insilico_size100_5	100	48h	0.071

Experiments run on a processor Intel Xeon (X5650, 2.67GHz) with 12GB of RAM

# Comparison with other methods

DREAM4: Combining Genetic and Dynamic Information to Identify Biological Networks and Dynamical Models  
[Greenfield et al., 2010]

## Double Knockout Predictions



Precision of prediction on the networks of size 100.

# Artificial Neural Networks: learning transition

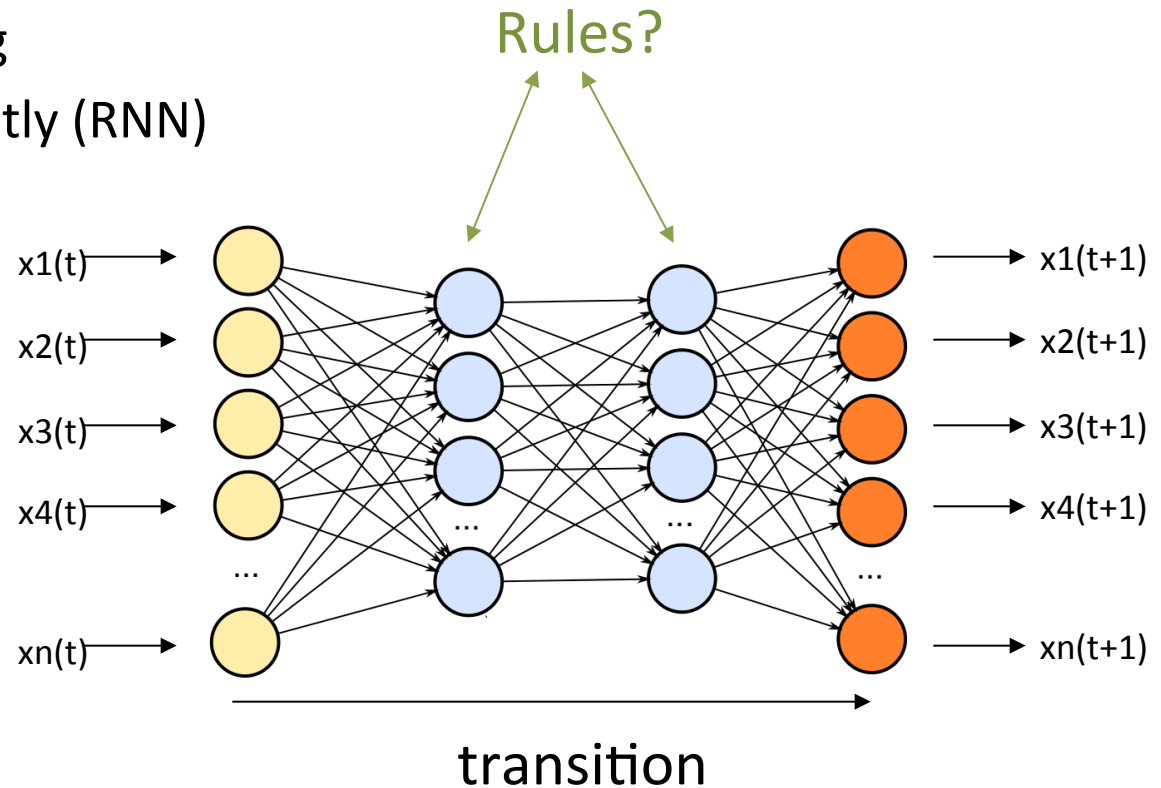
(Ongoing work with Enguerrand Gentet )

- **Advantages:**

- Continuous variables
- Good generalization
- Inter-variables learning
- Manage delays efficiently (RNN)

- **Challenges:**

- **Architecture choice**
- **Rule extraction**





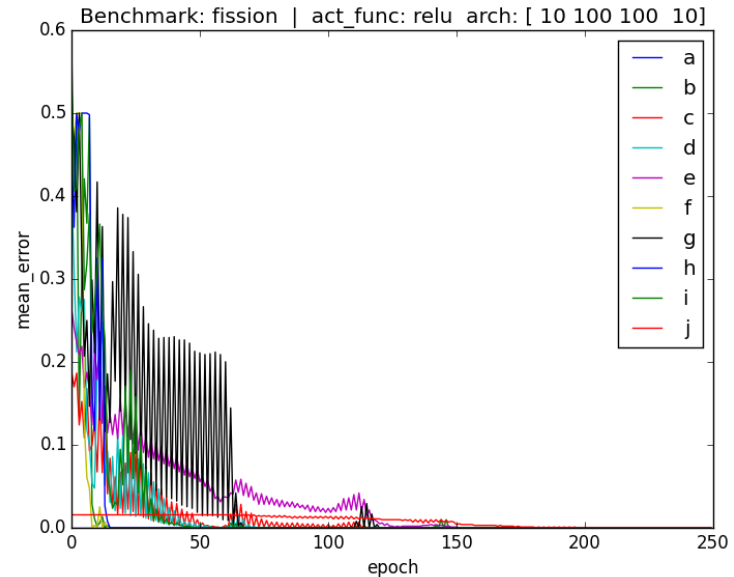
# Artificial Neural Networks: ongoing work

## Ongoing: NN-LFIT

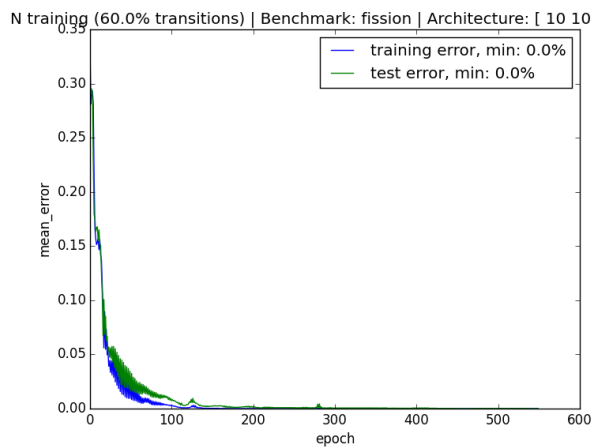
- Learning Boolean networks
- 4-layer NN (2 hidden layers)
- No delays

## Results:

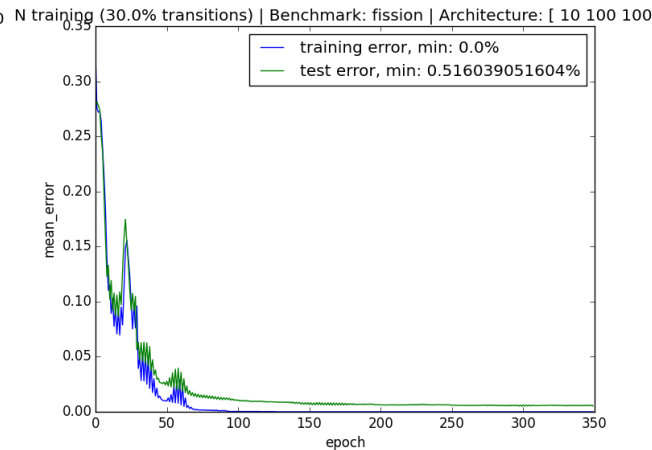
- Successful learning
- Good generalization



Training on the whole set of transition:  
all the transitions are successfully learned



Training on 60% of transitions:  
0% error on the remaining 40%



Training on 30% of transitions:  
0.5% error on the remaining 70%

## Plans:

- continuous values
- rule extraction
- delays

# LFIT: Summary

- **Motivation:** Modeling in Dynamic Environments
- **Principles:** LF1T (Learning from 1-Step Transitions)
  - Bottom-Up Algorithm (generalization)
  - BDD Optimization
  - Top-Down Algorithm (specialization)
- **Extensions:** LF $k$ T (Learning Markov( $k$ ) Systems), Multi-Valued/Asynchronous/Probabilistic Extensions
- **Applications:** Biology, Robotics, AGI, etc.
- **Ongoing Work:** DREAM Challenges, NN-LFIT

# Publication (1/2)

- Katsumi Inoue: [Logic Programming for Boolean Networks](#), in: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pp.924-930, AAAI (2011)
- Katsumi Inoue, Chiaki Sakama: [Oscillating Behavior of Logic Programs](#), in: *Correct Reasoning—Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, LNAI, Vol.7625, pp. 345-362, Springer (2012)
- Daniel Sykes, Domenico Corapi, Jeff Magee, Jeff Kramer, Alessandra Russo, Katsumi Inoue: [Learning Revised Models for Planning in Adaptive Systems](#), in: *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*, pp.63-71, IEEE/ACM (2013)
- Chiaki Sakama, Katsumi Inoue: [Abduction, Unpredictability and Garden of Eden](#), *Logic Journal of the IGPL*, 21(6):980-998 (2013)
- Katsumi Inoue, Tony Ribeiro, Chiaki Sakama: [Learning from Interpretation Transition](#), *Machine Learning*, 94(1):51-79 (2014)
- Tony Ribeiro, Katsumi Inoue, Chiaki Sakama: [A BDD-Based Algorithm for Learning from Interpretation Transition](#), in: *Post-Proceedings of ILP 2013*, LNAI, Vol.8812, pp.47-63, Springer (2014)
- Tony Ribeiro, Katsumi Inoue: [Learning Prime Implicant Conditions from Interpretation Transition](#), in *Post-Proceedings of ILP 2014*, LNAI, Vol.9046, pp.108-125, Springer (2015)
- Tony Ribeiro, Morgan Magnin, Katsumi Inoue, Chiaki Sakama: [Learning Delayed Influences of Biological Systems](#), *Frontiers in Bioengineering and Biotechnology*, 2:81 (2015)

# Publication (2/2)

- Tony Ribeiro, Morgan Magnin, Katsumi Inoue, Chiaki Sakama: [Learning Multi-Valued Biological Models with Delayed Influence from Time-Series Observations](#), in: *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA 2015)*, pp.25-31, IEEE (2015)
- Marcus Völker, Katsumi Inoue: [Logic Programming for Cellular Automata](#), in: *Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015)*, *CEUR Workshop Proceedings*, Vol.1433 (2015)
- David Martínez, Tony Ribeiro, Katsumi Inoue, Guillem Alenyà, Carme Torras: [Learning Probabilistic Action Models from Interpretation Transitions](#), in: *Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015)*, *CEUR Workshop Proceedings*, Vol.1433 (2015)
- David Martínez, Tony Ribeiro, Katsumi Inoue, Guillem Alenyà, Carme Torras: Learning Relational Dynamics of Stochastic Domains for Planning, in: *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS '16)*, to appear, AAAI (2016)
- Chiaki Sakama, Katsumi Inoue: [Can Machines Learn Logics?](#) in: *Proceedings of the 8th International Conference on Artificial General Intelligence (AGI 2015)*, *LNAI*, Vol.9205, pp. 341-351, Springer (2015)
- Chiaki Sakama, Tony Ribeiro, Katsumi Inoue: Learning Inference by Induction, in: *Post-Proceedings of ILP 2015*, *LNAI*, Vol.9575, to appear, Springer (2016)