Inference of Dynamic Boolean Networks
動的ネットワークの論理モデルに関する推論と学習

Katsumi Inoue
井上 克巳
National Institute of Informatics, Japan
国立情報学研究所

CSPSAT2-ERATO Minato Projects Joint Seminar on Theory, Implementation, and Applications of Logic and Inference

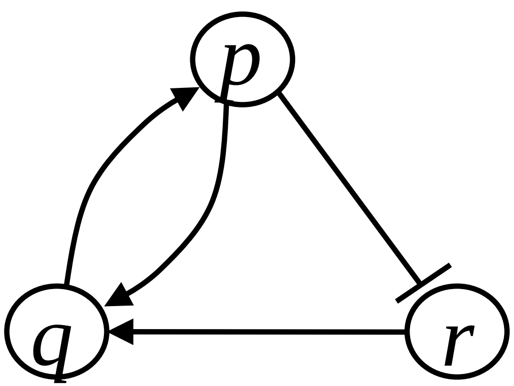
論理と推論の理論, 実装, 応用に関する合同セミナー

Hokkaido University, Sapporo, July 25th, 2013

Boolean Network

- Discrete model of genetic networks and adaptive systems
- $N = (V, F)$
  - $V = \{v_1, \ldots, v_n\}$: finite set of nodes ⇔ gene
  - $F = \{f_1, \ldots., f_n\}$: Boolean functions ⇔ gene regulation rule

Boolean network



$$p(t+1) = q(t)$$
$$q(t+1) = p(t) \wedge r(t)$$
$$r(t+1) = \overline{p(t)}$$

State transition table

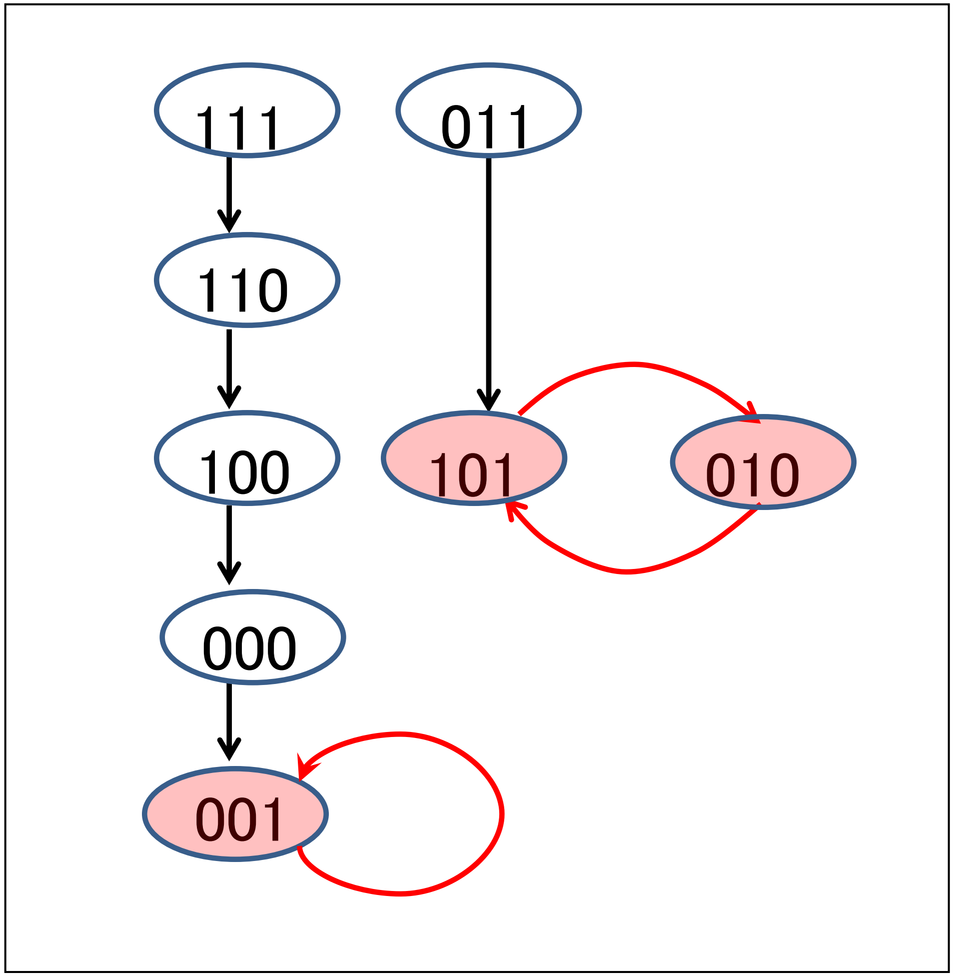
Input			Output		
Time t			Time $t+1$		
p	q	r	p	q	r
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	0

Attractors

$$p(t+1) = q(t)$$
$$q(t+1) = p(t) \wedge r(t)$$
$$r(t+1) = \overline{p(t)}$$

- Periodic sequence of states
  - 011 → 101 → 010 → 101 → 010 → …
  - 111 → 110 → 100 → 000 → 001 → 001 → …
- Different attractors ⇔ Different cell types
- In Synchronous BN, any node reaches one attractor.

State transition diagram



Normal Logic Programs

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

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

  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* $\mathbf{H}_P$ is the set of ground atoms from language(*P*).
- An (*Herbrand*) *interpretation* of an NLP *P* is a subset of $\mathbf{H}_P$.
- An interpretation *I satisfies* a ground rule of the form:

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

  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

- $\boxed{T_P(I) := \{ \mathrm{H} \mid \mathrm{H} \leftarrow \mathrm{L}_1 \wedge \ldots \wedge \mathrm{L}_n \in \mathit{ground}(P),\ I \models \mathrm{L}_1 \wedge \ldots \wedge \mathrm{L}_n \}.}$
- When P is a *definite* program, $I \models \mathrm{A}_1 \wedge \ldots \wedge \mathrm{A}_m$ iff $\forall i.\ \mathrm{A}_i \in I$.

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

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

  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,

  $I \models \mathrm{A}_1 \wedge \ldots \wedge \mathrm{A}_m \wedge \neg\mathrm{B}_1 \wedge \ldots \wedge \neg\mathrm{B}_n$ iff $\forall i.\ \mathrm{A}_i \in I$ and $\forall j.\ \mathrm{B}_j \notin I$.

  In this case, T_P is *nonmonotone* (Apt, Blair & Walker, 1988).
- The *orbit* of I wrt P (Blair *et al*., 1997) is $\langle T_P{}^k(I)\rangle_{k=0,1,2,\ldots}$,
  where $T_P{}^0(I) = I$, $T_P{}^{k+1}(I) = T_P(T_P{}^k(I))$ for $k = 0, 1, 2, \ldots$.

T_P operator, supportedness, completion

- An interpretation I is ***supported*** (Apt, Blair & Walker, 1988) if $\forall A \in I.\ \exists(A \leftarrow A_1 \wedge \ldots \wedge A_m \wedge \neg B_1 \wedge \ldots \wedge \neg B_n) \in \mathit{ground}(P)$ such that $\forall i.\ A_i \in I$ and $\forall j.\ B_j \notin I$.
- **<u>Prop.</u>** An interpretation I is a model of P iff $T_P(I) \subseteq I$.
- **<u>Prop.</u>** I is supported iff $I \subseteq T_P(I)$.
- **<u>Cor.</u>** I is a *supported model* of P iff $I = T_P(I)$.
- **<u>Prop.</u>** I is a model of *Comp*(P) iff $I = T_P(I)$, where *Comp*(P) is Clark's completion of P.
- **<u>Cor.</u>** I is a supported model of P iff I is a model of *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 *nonmonotone*.
- No fixpoint is reached in general.
- No supported model exists here.

Translating Synchronous BNs into NLPs (Inoue, *IJCAI* 2011)

Given a BN $N = (V, F)$, transform each $f_i \in F$ to DNF:

$$f_i(t) = \bigvee_{j=1}^{l_j} B_{i,j}(t),$$

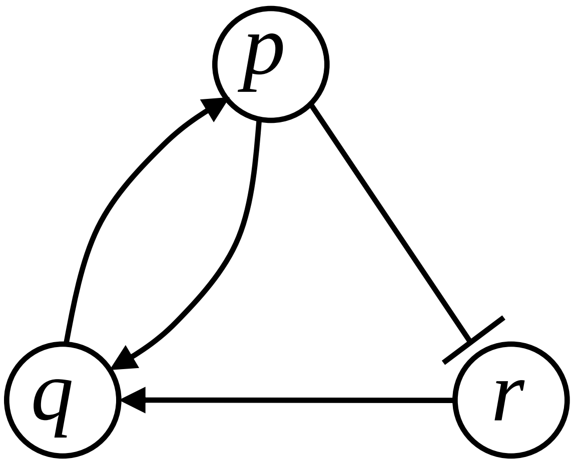
$$B_{i,j}(t) = v_{i,j,1}(t) \wedge \cdots \wedge v_{i,j,m_j}(t) \wedge \neg v_{i,j,m_j+1}(t) \wedge \cdots \wedge \neg v_{i,j,n_j}(t)$$

$V_C \subseteq V$: *constant nodes* (j=0)

$$\boxed{\begin{aligned}\pi(N) = \{ (v_i \leftarrow B_{i,j}) \mid v_i \in V \setminus V_C,\ 1 \le j \le l_i \} \\ \cup \{ (v_i \leftarrow v_i) \mid v_i \in V_C \}.\end{aligned}}$$

- For any state $\mathbf{v}(t) \in \{0,1\}^n$, put $I^t = \{ v_i \in V \mid v_i(t) = 1 \}$.
- $I^{t+1} = T_{\pi(N)}(I^t)$. The orbit of I^t wrt $T_{\pi(N)}$ is precisely the trajectory of N starting from $\mathbf{v}(t)$.

Boolean Network (Example)



$$p \leftarrow q.$$

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

$$r \leftarrow \neg p.$$

- Starting from $\mathbf{v}(0)=(0,1,1)$, the orbit of I_0 wrt $\pi(N)$ becomes:

1. $\{q, r\}$
2. $\{p, r\}$
3. $\{q\}$
4. $\{p, r\}$
5. repeat 3—2

- Starting from $\mathbf{v}(0)=(0,0,0)$, the orbit of I_0 wrt $\pi(N)$ becomes:

1. $\{\}$
2. $\{r\}$
3. $\{r\}$
4. fixpoint

Characterizing Point Attractors

- **<u>Theorem</u> (Inoue, 2011):** {*I*} is a point attractor of *N* iff *I* is a supported model of $\pi(N)$.
- The supported models of an NLP are exactly the models of its Clark's completion:

$$Comp(\pi(N)) = \bigwedge_{v_i \in V \setminus C} \left(v_i \leftrightarrow \bigvee_{j=1}^{l_i} B_{i,j} \right) \wedge \cancel{\bigwedge_{c_i \in C} (c_i \leftrightarrow c_i)}$$

c.f. (Tamura & Akutsu, 2009).

Supported Classes

(Inoue & Sakama, *Lifschitz Festschrift*, 2012)

- A **supported class** of a logic program P is defined as a nonempty set $\boldsymbol{S}$ of Herbrand interpretations satisfying the fixpoint equation:

$$\boldsymbol{S} = \{\ T_P(I) \mid I \in \boldsymbol{S}\ \}.$$

- A supported class $\boldsymbol{S}$ of P is **strict** if no proper subset of $\boldsymbol{S}$ is a supported class of P.
- **Theorem**: A non-empty set $\boldsymbol{S}$ of Herbrand interpretations is a strict supported class of P iff $\boldsymbol{S} = \{T_P{}^k(I) \mid k \in \omega\}$ for every $I \in \boldsymbol{S}$.
- **Theorem**: A finite set $\boldsymbol{S}$ of Herbrand interpretations of P is a strict supported class of P iff there is a directed cycle $I_1 \rightarrow I_2 \rightarrow \cdot\cdot\cdot \rightarrow I_k \rightarrow I_1$ ($k \geq 1$) in the state transition graph induced by T_P such that $\{I_1, I_2, \ldots, I_k\} = \boldsymbol{S}$.
- **Prop.**: Let $\boldsymbol{S}$ and $\boldsymbol{S'}$ be strict supported classes of a logic program P that has a finite Herbrand base. Then, $\boldsymbol{S} \neq \boldsymbol{S'}$ iff $\boldsymbol{S} \cap \boldsymbol{S'} = \{\}$.

Characterizing Attractors

- **<u>Theorem</u> (Inoue & Sakama, 2012):** ***S*** is an attractor of a Boolean network *N* iff ***S*** is a strict supported class of $\pi(N)$.

- **<u>Proposition</u>:** An interpretation *I is a supported model o*f a logic program *P* iff {*I*} is a supported class of *P.*
- **<u>Corollary</u>:** {*I*} is a point attractor of a Boolean network *N* iff *I* is a supported model of $\pi(N)$.

Supported Classes = Attractors

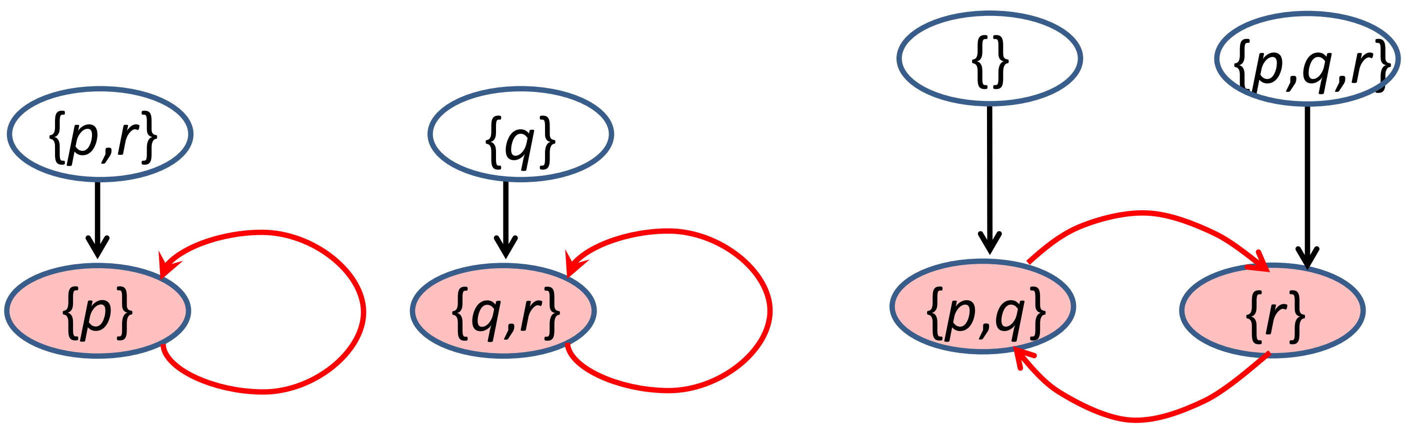
- P_1:

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

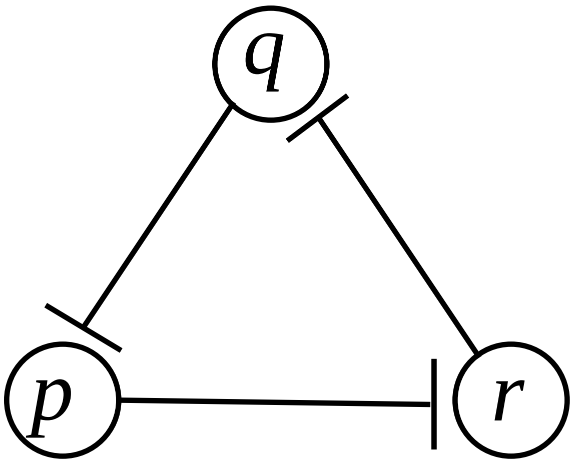
- There are 3 strict supported classes of P_1:

$$S_1 = \{\{p\}\},\ S_2 = \{\{q, r\}\},\ S_3 = \{\{p, q\}, \{r\}\}.$$

- S_1 and S_2 are the supported models of P_1 (*point attractors*).



Repressilator (Elowitz & Leibler, *Nature* **403**, 2000)

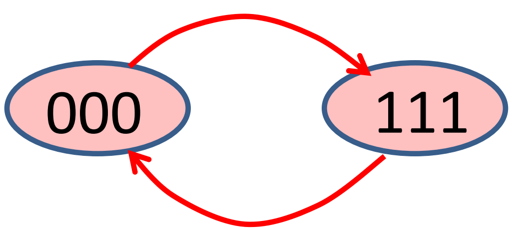


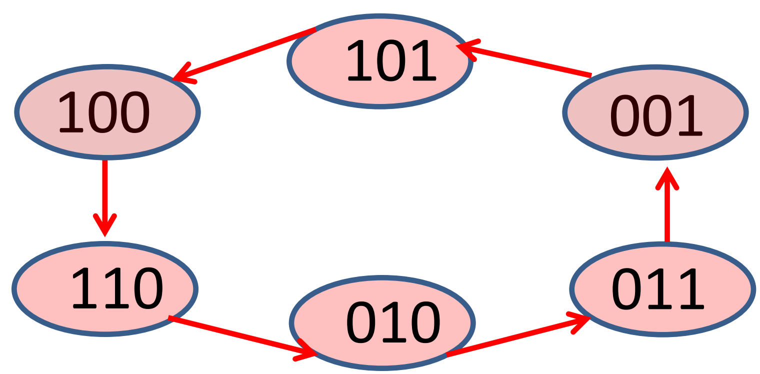
$$p(t+1) = \overline{q(t)}$$
$$q(t+1) = \overline{r(t)}$$
$$r(t+1) = \overline{p(t)}$$

$\pi(N)$:

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

- $\pi(N)$ has no supported model, but has 2 supported classes, which correspond to cycle attractors with period 2 and 6.





Learning 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)
- 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,* 2013)

- Herbrand interpretation I: a state of the world
- Logic program P: a *state transition system*, which maps an Herbrand interpretation into another interpretation (Blair *et al*., 1995—1997; Inoue, 2011; Inoue & Sakama, 2012)
- Next state $T_P(I)$: where T_P is the immediate consequence operator (T_P *operator*).
- We propose a new learning setting in ILP:
  - Given: a set of pairs of Herbrand interpretations (I,J) such that $J = T_P(I)$,
  - Induce a program P.
- C.f. learning from interpretations (LFI)
  - Given: a set S of Herbrand interpretations,
  - Induce a program P whose models are exactly S.

LFIT Applied to Dynamic Systems

- Learning rules of dynamic systems
  - Cellular Automata (CAs): mathematical model of complex adaptive systems (Conway, Wolfram)
  - Boolean Networks (BNs): logical model of gene regulation networks (Kauffman)
- CAs and BNs can be characterized as logic programs, and T_P operator captures their synchronous update (Inoue 2011).

- A learned program P is a *normal logic program* (NLP) in this case.
- Learning NLPs has been considered in ILP, but most approaches take the setting of *learning from entailment*.
- Learning NLPs under the *supported model semantics*.

LFIT Applied to Genetic Networks

- Given an Herbrand interpretation I, which corresponds to a *gene activity profile* (GAP) with *gene disruptions for* false atoms in I and *gene overexpressions* for true atoms in I, the interactions between genes are experimentally analyzed by observing a GAP J such that $J = T_P(I)$ holds after a time step has passed.
- LFIT of an NLP P corresponds to inferring a set of gene regulation rules for those experiments of 1-step GAP transitions in a BN.
- Any trajectory from a GAP in a BN reaches an *attractor,* which is either a *fixed point* or a *periodic oscillation.*
- Given a set of trajectories reaching to attractors of a BN, we can also infer an NLP that realizes these trajectories.

Subsumption, least generalization

- For two rules R_1, R_2 with the same head, R_1 *subsumes* R_2 if there is a substitution θ 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 $q(s_1,\dots, s_n)$ 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:

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

LF1T: **L**earning **f**rom **1**-Step **T**ransitions

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

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

$$R^I_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in \mathbf{B} ¥ I} \neg C ;$$

1. If R^I_A is not subsumed by any rule in P, *then* $P := P \cup \{R^I_A\}$ 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 $(b(R_2) ¥ \{\bar{l}\}) \subseteq (b(R_1) ¥ \{l\})$ then the *ground resolution* of R_1 and R_2 (upon l) is defined as

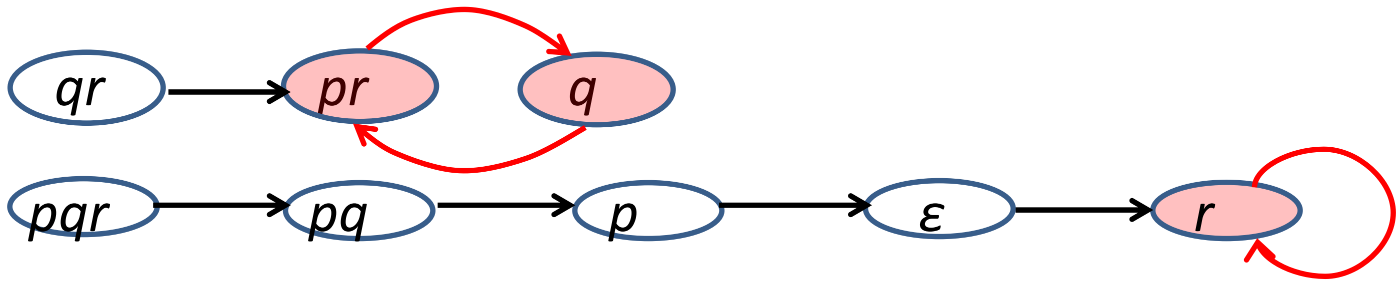
$$res(R_1,R_2) \;=\; h(R_1) \leftarrow \bigwedge_{K \in b(R_1)¥\{l\}} K$$

  In particular, if $(b(R_2) ¥ \{\bar{l}\}) = (b(R_1) ¥ \{l\})$ then the ground resolution is called the *naïve resolution* of R1 and R2 (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) [$R^I_A := A \leftarrow \bigwedge_{B\in I} B \wedge \bigwedge_{C\in \mathbf{B}\setminus I} \neg C$]



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) $[R^I{}_A := A \leftarrow \bigwedge_{B\in I} B \wedge \bigwedge_{C\in \mathbf{B}\text{¥}I} \neg C]$

Step	$I \rightarrow J$	Operation	Rule	ID	P	P_{old}
5	$pq \rightarrow p$	$R^{pq}{}_p$	$p \leftarrow p \wedge q \wedge \neg r$	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	$\varepsilon \rightarrow r$	$R^{\varepsilon}{}_r$	$r \leftarrow \neg p \wedge \neg q \wedge \neg r$	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

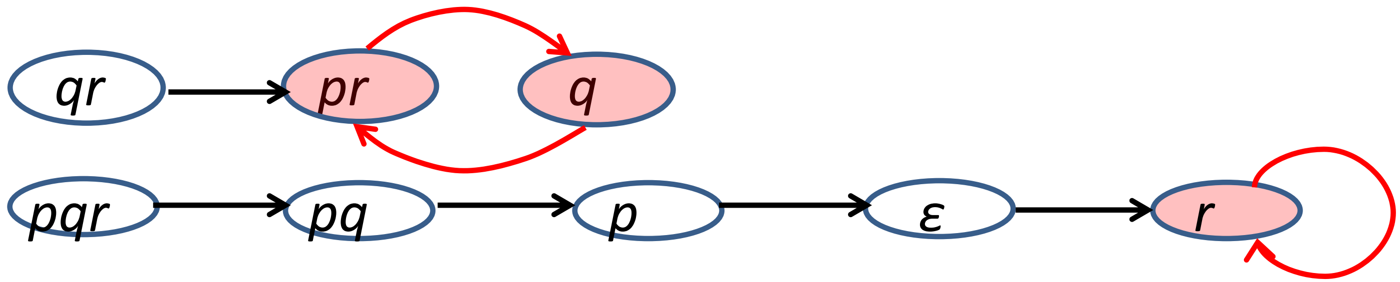
$$p \leftarrow q.$$
$$q \leftarrow p \wedge r.$$
$$r \leftarrow \neg p.$$

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) $[R^I_A := A \leftarrow \bigwedge_{B\in I} B \wedge \bigwedge_{C\in \mathbf{B}\setminus I} \neg C]$



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) $[R^I_A := A \leftarrow \bigwedge_{B \in I} B \wedge \bigwedge_{C \in \mathbf{B} \setminus I} \neg C]$

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

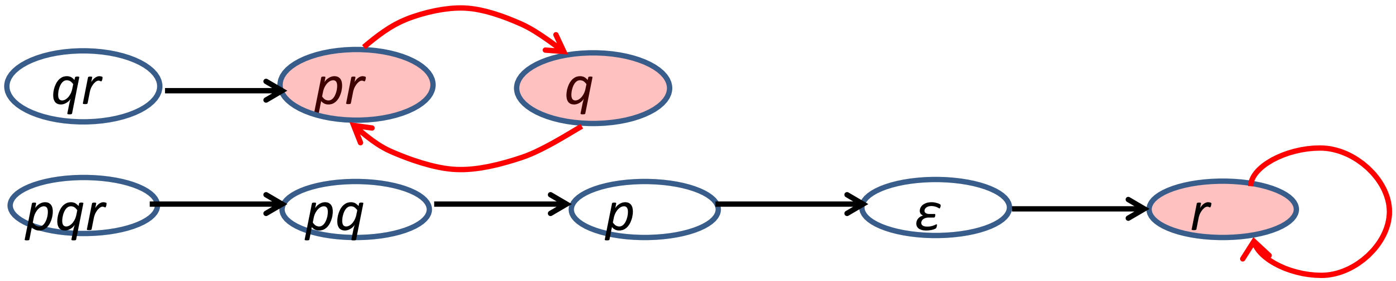
- **Theorem.** Using naïve resolution, the memory use of the LF1T algorithm is bounded by $O(n \cdot 3^n)$, and the time complexity of learning is bounded by $O(n^2 \cdot 9^n)$, where $n = |\boldsymbol{B}|$. On the other hand, with ground resolution, the memory use is bounded by $O(2^n)$, which is the 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 **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 $O(n^2 \cdot |E|^{4.5})$.

LFBA: **L**earning **f**rom **B**asins of **A**ttraction

- **Input:** $\mathcal{E} \subseteq 2^{2^{\mathbf{B}}}$: A set of orbits of interpretations (*)
- **Output:** NLP P s.t. for $\forall \mathcal{I} \in \mathcal{E}$, any $I \in \mathcal{I}$ belongs to the basin of attraction of some attractor of P contained in $\mathcal{I}$
- * **Assumption:** Each $\mathcal{I}$ contains the interpretations belonging to the orbit of some $I_0 \in \mathcal{I}$ wrt T_P, and that $\mathcal{I}$ constitutes a sequence $I_0 \rightarrow I_1 \rightarrow \ldots \rightarrow I_{k-1} \rightarrow J_0 \rightarrow \ldots \rightarrow J_{l-1} \rightarrow J_0 \rightarrow \ldots$, where $|\mathcal{I}| = k + l$ and $\{J_0, \ldots , J_{l-1}\}$ is an attractor.
- 2 orbits $\mathcal{I}, \mathcal{J} \in \mathcal{E}$ reach the same attractor iff $\mathcal{I} \cap \mathcal{J} = \emptyset$.

1. Put $P := \emptyset$;
2. If $\mathcal{E} = \emptyset$ then output P and stop;
3. Pick $\mathcal{I} \in \mathcal{E}$, and put $\mathcal{E} := \mathcal{E}$ ¥ $\{\mathcal{I}\}$;
4. Put $E := \{(I, J) \mid I, J \in \mathcal{I},\ J$ is the next state of $I\}$;
5. $P :=$ **LF1T**(E, P); Return to 2.

LFBA: Example



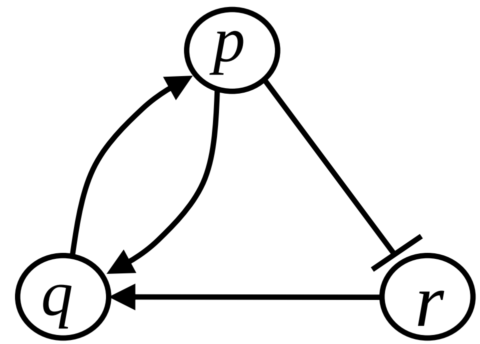
Input: $\mathcal{E} = \{I_1, I_2\}$

I_1 : $qr \rightarrow pr \rightarrow q \rightarrow pr \rightarrow q \rightarrow \ldots$

I_2 : $pqr \rightarrow pq \rightarrow p \rightarrow \varepsilon \rightarrow r \rightarrow r \rightarrow \ldots$

LF1T(E_1, ∅, ∅) = {3,5,7};

LF1T(E_2, {3,5,7}) = {11,14,19};



In general, identification of an exact NLP using **LF1T** may require $2^{|\mathbf{B}|}$ examples, while $|\mathcal{E}|$ in **LFBA** is bounded by $c\delta$, where δ is the number of attractors.

Learning Boolean Networks

- Benchmarks of Boolean networks are taken from (Dubrova and Teslenko, 2011).
- All possible 1-step state transitions of N from all $2^{|B|}$ 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	# × length of attractor	# rules (org./LFIT)	Naïve	Ground
Arabidopsis thalania	15	10 × 1	28 / 241	T.O.	13.825s
Budding yeast	12	7 × 1	54 / 54	6m01s	0.820s
Fission yeast	10	13 × 1	23 / 24	5.208s	0.068s
Mammalian cell	10	1 × 1, 1 × 7	22 / 22	5.756s	0.076s

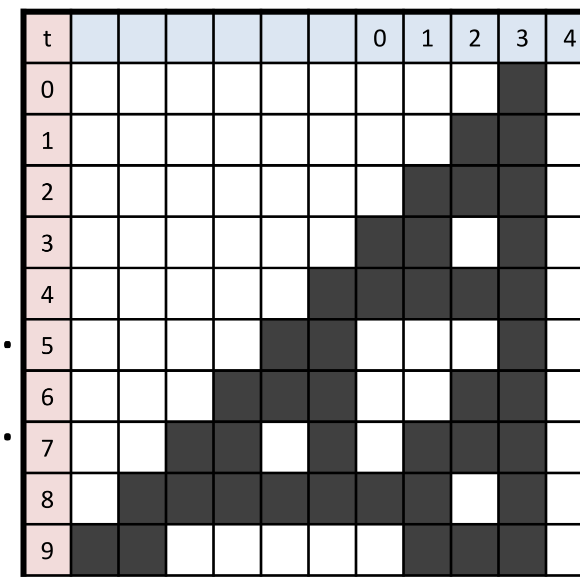
Cellular Automata (CA)

- 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 (the 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) ← c(x-1,t) ∧ c(x,t) ∧ ¬c(x+1,t).
- c(x,t+1) ← c(x-1,t) ∧ ¬c(x,t) ∧ c(x+1,t).
- c(x,t+1) ← ¬c(x-1,t) ∧ c(x,t) ∧ c(x+1,t).
- c(x,t+1) ← ¬c(x-1,t) ∧ c(x,t) ∧ ¬c(x+1,t).
- c(x,t+1) ← ¬c(x-1,t) ∧ ¬c(x,t) ∧ c(x+1,t).

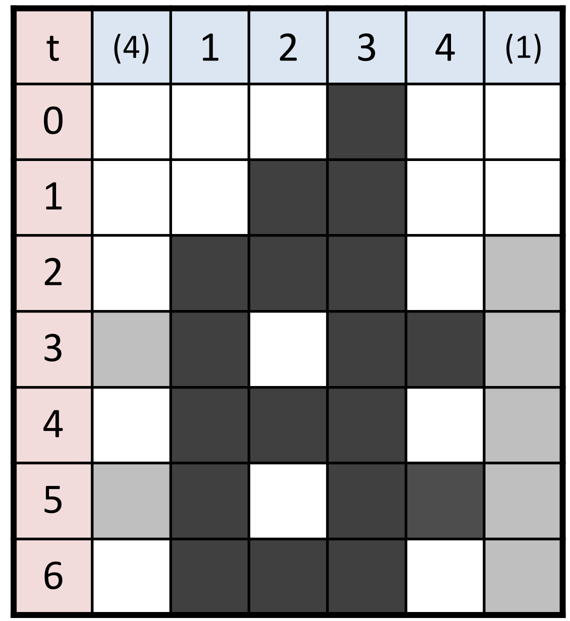


- Rule 110 is known to be Turing-complete.
- The logic program is *acyclic* (Apt & Bezem, 1990).

Incorporating Background Theories

- Torus world: length 4
- c(0, t) ← c(4, t).
- c(5, t) ← 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) → …



learning rules: 0→1 (4), 1→2 (2), 2→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)	$c(x) \leftarrow \neg c(x\text{-}1) \wedge \neg c(x) \wedge c(x+1)$	2	2
		$R^3{}_3$	$c(3) \leftarrow \neg c(2) \wedge c(3) \wedge \neg c(4)$	3	
		AI(3)	$c(x) \leftarrow \neg c(x\text{-}1) \wedge c(x) \wedge \neg c(x+1)$	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)	$c(x) \leftarrow \neg c(x\text{-}1) \wedge c(x) \wedge c(x+1)$	7	
		res(7,2)	$c(x) \leftarrow \neg c(x\text{-}1) \wedge c(x+1)$	8	4,8
		res(7,4)	$c(x) \leftarrow \neg c(x\text{-}1) \wedge c(x)$	9	8,9

Incorporating Inductive Bias (Cont.)

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

- $c(x,t+1) \leftarrow \neg c(x\text{-}1,t) \land c(x+1,t)$. (8)
- $c(x,t+1) \leftarrow \neg c(x\text{-}1,t) \land c(x,t)$. (9)
- $c(x,t+1) \leftarrow c(x,t) \land \neg c(x+1,t)$. (12)
- $c(x,t+1) \leftarrow \neg c(x,t) \land c(x+1,t)$. (16)

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

Conclusion & Ongoing Work

- Oscillating behavior can be observed in any deterministic operator on the Herbrand base. The attractors of synchronous Boolean networks are completely characterized by the supported class semantics of NLPs.
- Learning complex networks becomes more and more important. We tackled the induction problem of such dynamic systems in terms of NLP learning from synchronous state transitions.
  - Given any state transition diagram, which is either complete or partial, we can learn an NLP that exactly captures the system dynamics.
  - Learning is performed only from positive examples, and produces NLPs that consist only of rules to make literals true.
  - Generalization on state transition rules is done by resolution, in which each rule can be replaced by a general rule. An output NLP is as minimal as possible wrt the size of each rule, but may contain redundant rules.
- A more efficient construction in the bottom-up algorithm with BDD (ILP 2013).
- More complex schemes such as asynchronous and probabilistic updates do not obey transition by the T_P operator.

Reference

- Katsumi Inoue. Logic Programming for Boolean Networks. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (IJCAI-11), pp.924-930, AAAI Press, 2011.
- Katsumi Inoue and 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.
- Chiaki Sakama and Katsumi Inoue. Abduction, Unpredictability and Garden of Eden. *Logic Journal of the IGPL*, to appear, 2013.
- Katsumi Inoue, Tony Ribeiro and Chiaki Sakama. Learning from Interpretation Transition. *Machine Learning*, to appear, 2013.
- Tony Ribeiro, Katsumi Inoue and Chiaki Sakama. A BDD-Based Algorithm for Learning from Interpretation Transition. In: *Proceedings of ILP 2013, LNAI*, Springer, to appear, 2013.