Using Reachability Properties of Logic Program for Revising Biological Models

Xinwei Chai, Tony Ribeiro, Morgan Magnin, Olivier Roux, Katsumi Inoue

Laboratoire des Sciences du Numérique de Nantes, France
National Institute of Informatics, Tokyo

September 4, 2018
Process Scheme

- Biological *a priori* knowledge
- Real system
- Temporal properties
- Partial observation
- LFIT
- Model
  - Some reachability
  - Model Checking
Modelings

Boolean Network

\[
\begin{align*}
    f(a) &= \neg b \\
    f(b) &= a
\end{align*}
\]

Logic Program

\[
\begin{align*}
    a(t+1) &\leftarrow \neg b(t) \\
    b(t+1) &\leftarrow a(t)
\end{align*}
\]

State transition graph
Reachability problem

Given a BN, from initial state $\alpha$, does there exist a transition sequence that reaches the target state $\omega$?

Given a state transition graph, from initial state $\alpha$, does there exist a pathway towards the target state $\omega$?

Reachability of global states $\text{EF}(a_i, b_j, \ldots)$ → computationally difficult

$\implies$ Reachability of local states $\text{EF}a_i$
Difficulties and solution

- State space grows exponentially with the number of automata
- Traditional model checkers e.g. Mole\(^1\) and NuSMV\(^2\) fail global search → time out and/or out of memory
- **Static analysis**: avoid global search, at the cost of precision
  → A balance between time-space performance and conclusiveness

- Paulevé *et al.* introduced LCG (Local Causality Graph) [1, 2] for static analysis
- Implementation: Pint

- Efficient (beats many traditional model checkers) **but**
- Usually not conclusive when the density of the biological network increases

---

\(^1\)http://www.lsv.fr/~schwoon/tools/mole
\(^2\)http://nusmv.fbk.eu
Local Causality Graph (LCG)

Start with target state $\omega \rightarrow$ Find transitions reaching $\omega \rightarrow$ Find new target states to fire those transitions $\rightarrow \cdots$ Recursion $\cdots \rightarrow$ End with initial state $\alpha$

- Goal-oriented structure
- Formed by recursive updates
- Avoid global search in state transition graphs
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$,
$b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$, $b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$, $b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$,
$b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$,
$b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$,
$b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
$r'(a_1) = r'(e_1) \lor (r'(b_1) \land r'(c_1))$
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$, $b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
$r'(a_1) = r'(d_0) \land r'(c_1)$
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$, $b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes

$r'(a_1) = r'(d_0) \land r'(d_1)$
Example of LCG

Initial state \( \alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle \), target state \( \omega = a_1 \)

Rules: \( a_1 \leftarrow b_1 \land c_1 \), \( a_1 \leftarrow e_1 \),
\( b_1 \leftarrow d_0 \), \( c_1 \leftarrow d_1 \), \( d_1 \leftarrow b_1 \)

Small circles stand for transition nodes, squares for state nodes
\( r'(a_1) = r'(d_1) \)
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules:
- $a_1 \leftarrow b_1 \land c_1$
- $a_1 \leftarrow e_1$
- $b_1 \leftarrow d_0$
- $c_1 \leftarrow d_1$
- $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes

$r'(a_1) = r'(b_1) = r'(d_0) = 1$
Example of LCG

Initial state $\alpha = (a_0, b_1, c_0, d_0, e_0)$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$, $b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Example of LCG

Initial state $\alpha = \langle a_0, b_1, c_0, d_0, e_0 \rangle$, target state $\omega = a_1$

Rules: $a_1 \leftarrow b_1 \land c_1$, $a_1 \leftarrow e_1$, $b_1 \leftarrow d_0$, $c_1 \leftarrow d_1$, $d_1 \leftarrow b_1$

Small circles stand for transition nodes, squares for state nodes
Algorithm for Reachability

- **Input:** A logic program $P$, an initial state $\alpha$, a target state $\omega$ and a max number of iterations $k$
- **Output:** $reach(\omega) \in \{\text{False, True, Inconclusive}\}$

1. Construct the LCG $\ell = LCG(P, \alpha, \omega)$
2. Try to remove all cycles and prune useless edges from $\ell$
3. Try to prove unreachability of $\omega$ in $\ell$ using pseudo-reachability $reach'(\ell, \omega)$ and return **False** if $reach'(\ell, \omega) = \text{False}$
4. Try at most $k$ times
   - $\ell' \leftarrow \ell$
   - Simplify each **OR gate** such that $\ell'$ is a LCG with only **AND gates**
   - If there remain cycles:
     - Back to step (4)
   - Generate all trajectory that starts with $\alpha$ in $\ell'$ using ASP
     - If a trajectory $t$ ending with $\omega$ is found, return **True**
5. return **Inconclusive**
ASPReach

In an LCG, link $a_1 \rightarrow _\circ \rightarrow b_1$ can be translated as:
\[
\text{node('a','1',1). node('b','1',2). parent(1,2).}
\]

Core code:

\[
\begin{align*}
\text{prior(N1,N2) :- & parent(N2,N1). } \quad \text{Rule 1} \\
\text{prior(N1,N3) :- & prior(N1,N2), prior(N2,N3). } \quad \text{Rule 2} \\
\text{prior(N1,N2) :- & node(P1,S1,N1), node(P2,S2,N2),} \\
& \quad \text{node(P2,S3,N3), parent(N1,N3),} \\
& \quad \text{init(P2,S3), S2!=S3, P1!=P2. } \quad \text{Rule 3}
\end{align*}
\]

N for node, P for component, S for state
Rule 3: in the LCG, one branch contains $a_1 \rightarrow _\circ \rightarrow b_0$, another branch contains $b_1$, if $b_0 \in \alpha$, $a_1$ is to be reached before reaching $b_1$
Example

Initial state $\alpha = a_0, b_0, c_0$, target state $\omega = c_1$
Rules: $a_1 \leftarrow b_0, b_1 \leftarrow c_0, c_1 \leftarrow a_1 \land b_1$

$a \triangleright b$ means $a$ appears in the sequence before $b$

Rule 1 & 2 $\Rightarrow$ $b_0 \triangleright a_1 \triangleright c_1$, $c_0 \triangleright b_1 \triangleright c_1$
Rule 3 $\Rightarrow$ $a_1 \triangleright b_1$

The only admissible order is $a_1 \rightarrow b_1 \rightarrow c_1$
Benchmark

Traditional model checkers: Mole NuSMV → **memory-out**

Pure static analyzer: Pint [1]

Small example: λ-phage, 4 components

Big examples: TCR (T-Cell Receptor, 95 components) and EGFR (Epidermal Growth Factor Receptor, 106 components)

<table>
<thead>
<tr>
<th>Model</th>
<th>λ-phage</th>
<th>TCR</th>
<th>EGFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>4</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>Outputs</td>
<td>4</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>Total tests</td>
<td>$2^4 \times 4 = 64$</td>
<td>$2^3 \times 5 = 40$</td>
<td>$2^{13} \times 12 = 98,304$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyzer</th>
<th>Pint</th>
<th>PR</th>
<th>AR</th>
<th>Pint</th>
<th>PR</th>
<th>AR</th>
<th>Pint</th>
<th>PR</th>
<th>AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachable</td>
<td>36(56%)</td>
<td>38(59%)</td>
<td>38(59%)</td>
<td>16(40%)</td>
<td>64,282(65.4%)</td>
<td>74,268(75.5%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inconclusive</td>
<td><strong>2(3%)</strong></td>
<td>0(0%)</td>
<td>0(0%)</td>
<td>9,986(10.1%)</td>
<td>0(0%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unreachable</td>
<td>26(41%)</td>
<td>24(60%)</td>
<td>24,036(24.5%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total time</td>
<td>&lt; 1s</td>
<td>7s</td>
<td>0.85s</td>
<td>40s</td>
<td>9h50min</td>
<td>15min31s</td>
<td>3h46min</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PR=PermReach, AR=ASPReach
Collaboration with LFIT

- If the model is consistent with *a priori* knowledge
  - Do nothing
- If not consistent

<table>
<thead>
<tr>
<th>Knowledge</th>
<th>Reachable</th>
<th>Unreachable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge</td>
<td>$R_K$</td>
<td>$U_K$</td>
</tr>
<tr>
<td>Inferred model</td>
<td>$R_I$</td>
<td>$U_I$</td>
</tr>
<tr>
<td>Inconsistency (problem)</td>
<td>$R'_K = R_K \cap U_I$</td>
<td>$U'_K = R_I \cap U_K$</td>
</tr>
</tbody>
</table>

- Keep consistent with
  - Generalization (○)
  - Add transitions (×)
- Operation
  - Specialization (○)
  - Delete transitions (○)

where set $R$ and $U$ are consisting of pairs of form $(\alpha, \omega)$
Definitions

Specialization of a transition
By adding elements in the body of a transition, it is possible to change a reachable state to an unreachable one.

Generalization of a transition
By deleting elements in the body of a transition, it is possible to change an unreachable state to a reachable one.
Main Algorithm

- Input: an Automata Network $A$, reachable set $R_K$, unreachable set $U_K$
- Output: modified Automata Network $A$ or $\emptyset$ if not revisable

1. Construct the LCGs for the elements in $R_K$ and $U_K$, collect inconsistent instances in set $R'_K$ and $U'_K$
2. Specialize the transitions to make elements in $U'_K$ unreachable, if not possible, return $\emptyset$
3. Generalize the transitions to make elements in $R'_K$ reachable, if not possible, return $\emptyset$
4. Return $A$
Specialization

- Input: a logic program $P$, an unsatisfied element $(\alpha, \omega)$, a reachable set $Re$, an unreachable set $Un$
- Output: modified logic program $P$ or $\emptyset$ if not revisable

1. $Rev \leftarrow \{\omega\}$
2. For each $R$ s.t. $h(R) = Rev$, for each $R' \in \{R''| R'' \in Is(R) \land \exists (I, J) \in E, \ s.t. \ R''' \in P \cup \{R''\} \setminus \{R\}, h(R''') \in J, b(R''') \in I\}$
   - If $P' \leftarrow P \setminus \{R\} \cup \{R'\}$, $unreachable(P', \alpha, \omega)$ and $P'$ satisfies all previous properties, return $P'$
3. $Rev \leftarrow b(R)$ with $h(R) = Rev$ and back to step 2
4. There is no revision for $(\alpha, \omega)$, return $\emptyset$
Generalization

- Input: a logic program $P$, an unsatisfied element $(\alpha, \omega)$, a reachable set $Re$, an unreachable set $Un$
- Output: modified logic program $P$ or $\emptyset$ if not revisable

1. $Rev \leftarrow \{\omega\}$
2. For each $R$ s.t. $h(R) = Rev$, for each $R' \in lg(R)$
   - If $P' \leftarrow P \setminus \{R\} \cup \{R'\}$, reachable($P'$, $\alpha$, $\omega$) and $P'$ satisfies all previous properties, return $P'$
3. $Rev \leftarrow b(R)$ with $h(R) = Rev$ and back to step 2
4. There is no revision for $(\alpha, \omega)$, return $\emptyset$
Example

Rules: $a_1 \leftarrow b_1, a_1 \leftarrow d_1 \land c_0, b_1 \leftarrow c_0, c_1 \leftarrow b_0$
Initial state: $\alpha = \langle a_0, b_0, c_0, d_0 \rangle$
$U_K = \{(\alpha, b_1), (\alpha, d_1)\}, R_K = \{(\alpha, a_1)\}$

$L = \{\{(\alpha, a_1), (\alpha, b_1), (\alpha, d_1)\}, \{(\alpha, b_1)\}, \{(\alpha, d_1)\}\}$
Start from $\{(\alpha, b_1)\}$ and $\{(\alpha, d_1)\}$
$b_1 \leftarrow c_0$ can be specialized to $b_1 \leftarrow c_0 \land a_1$ to make $b_1$ unreachable
$a_1 \leftarrow d_1 \land c_0$ can only be generalized to $a_1 \leftarrow c_0$ as $d_1 \in U_K$
Check the reachability of $(\alpha, a_1)$: reachable, finish
Example

Rules: \( a_1 \leftarrow b_1, a_1 \leftarrow d_1 \land c_0, b_1 \leftarrow c_0, c_1 \leftarrow b_0 \)
Initial state: \( \alpha = \langle a_0, b_0, c_0, d_0 \rangle \)
\( U_K = \{ (\alpha, b_1), (\alpha, d_1) \}, R_K = \{ (\alpha, a_1) \} \)

\[ L = \{ \{ (\alpha, a_1), (\alpha, b_1), (\alpha, d_1) \}, \{ (\alpha, b_1) \}, \{ (\alpha, d_1) \} \} \]

- Start from \( \{ (\alpha, b_1) \} \) and \( \{ (\alpha, d_1) \} \)
- \( b_1 \leftarrow c_0 \) can be specialized to \( b_1 \leftarrow c_0 \land a_1 \) to make \( b_1 \) unreachable
- \( a_1 \leftarrow d_1 \land c_0 \) can only be generalized to \( a_1 \leftarrow c_0 \) as \( d_1 \in U_K \)
- Check the reachability of \( (\alpha, a_1) \): reachable, finish
Example

Rules: \(a_1 \leftarrow b_1, a_1 \leftarrow d_1 \land c_0, b_1 \leftarrow c_0, c_1 \leftarrow b_0\)

Initial state: \(\alpha = \langle a_0, b_0, c_0, d_0 \rangle\)

\(U_K = \{(\alpha, b_1), (\alpha, d_1)\}, R_K = \{(\alpha, a_1)\}\)

- \(L = \{\{(\alpha, a_1), (\alpha, b_1), (\alpha, d_1)\}, \{(\alpha, b_1)\}, \{(\alpha, d_1)\}\}\)
- Start from \{\((\alpha, b_1)\) and \{\((\alpha, d_1)\)\}
- \(b_1 \leftarrow c_0\) can be specialized to \(b_1 \leftarrow c_0 \land a_1\) to make \(b_1\) unreachable
- \(a_1 \leftarrow d_1 \land c_0\) can only be generalized to \(a_1 \leftarrow c_0\) as \(d_1 \in U_K\)
- Check the reachability of \((\alpha, a_1)\): reachable, finish
Example

Rules: $a_1 \leftarrow b_1, a_1 \leftarrow d_1 \land c_0, b_1 \leftarrow c_0, c_1 \leftarrow b_0$
Initial state: $\alpha = \langle a_0, b_0, c_0, d_0 \rangle$
$U_K = \{(\alpha, b_1), (\alpha, d_1)\}, \ R_K = \{(\alpha, a_1)\}$

$L = \{\{(\alpha, a_1), (\alpha, b_1), (\alpha, d_1)\}, \{(\alpha, b_1)\}, \{(\alpha, d_1)\}\}$

Start from $\{(\alpha, b_1)\}$ and $\{(\alpha, d_1)\}$

$b_1 \leftarrow c_0$ can be specialized to $b_1 \leftarrow c_0 \land a_1$ to make $b_1$ unreachable

$a_1 \leftarrow d_1 \land c_0$ can only be generalized to $a_1 \leftarrow c_0$ as $d_1 \in U_K$

Check the reachability of $(\alpha, a_1)$: reachable, finish
Conclusion

- Given background knowledge (reachability properties), the learned models are evaluated via LCG.
- Using classical specialization/generalization, the models learned by LF1T are revised while keeping consistent with the observation (time series data).

Ongoing work:
- Application in biological networks, e.g. mammalian circadian clock modeling
  - Exploit biologists knowledge to deal with few available data
References


Thank you!