Minimal Intervention Strategies in Logical Signaling Networks with Answer Set Programming

Roland Kaminski    Torsten Schaub    Anne Siegel
Santiago Videla
Contents

1 Introduction

2 Answer Set Programming

3 Benchmarks

4 Conclusions
Systems biology: a whole-istic approach

- aims at understanding the systems rather than isolated parts
- large amount of heterogeneous and noisy experimental *omics* data
- incomplete, contradictory, incorrect prior biological knowledge

Biologists rely on...

- abstract *representations* of complex biological systems
- computational tools for *reasoning* about such systems
Logical networks

\[ \phi = \left\{ \begin{array}{ll}
  i_1 & \mapsto i_1 \\
  i_2 & \mapsto i_2 \\
  a & \mapsto \neg d \\
  b & \mapsto a \land i_1 \\
  c & \mapsto b \lor e \\
  d & \mapsto c \\
  e & \mapsto \neg i_1 \land i_2 \\
  f & \mapsto e \lor g \\
  o_1 & \mapsto c \\
  o_2 & \mapsto g \\
\end{array} \right\} \]

Problems around logical networks:
- learning/completion/repairing
- finding steady states
- discriminate in-out behaviors
- identifying key-players

All of them are...
highly combinatorial search and optimization problems
Motivation: identifying key-players in signaling networks

How to systematically control the state of the cell?

Applications in...
cancer research, drug discovery, diagnosis, experimental design
Control of logical signaling networks


Intuitive notion of an “intervention strategy”

Given a logical network and side constraints, find Boolean interventions forcing a set of target species into a goal state.

Other approaches to control logical signaling networks:

- Based on elementary flux modes from metabolic networks. (Wang, R.-S. and Albert, R. 2011). BMC Systems Biology
Toy example

\[ \Phi = \{ \begin{align*}
    i_1 & \mapsto i_1 \\
    i_2 & \mapsto i_2 \\
    a & \mapsto \neg d \\
    b & \mapsto a \land i_1 \\
    c & \mapsto b \lor e \\
    d & \mapsto c \\
    e & \mapsto \neg i_1 \land i_2 \\
    f & \mapsto e \lor g \\
    g & \mapsto f \\
    o_1 & \mapsto c \\
    o_2 & \mapsto g
\end{align*} \} \]

- scenarios (goals, side constraints):
  1. \( \{ o_1 \mapsto f, o_2 \mapsto t \}, \{ i_1 \mapsto t \} \)
  2. \( \{ a \mapsto t \}, \emptyset \)
- intervention strategies?
Toy example

\[ \phi = \{ \]
\[ i_1 \mapsto i_1 \]
\[ i_2 \mapsto i_2 \]
\[ a \mapsto \neg d \]
\[ b \mapsto a \land i_1 \]
\[ c \mapsto b \lor e \]
\[ d \mapsto c \]
\[ e \mapsto \neg i_1 \land i_2 \]
\[ f \mapsto e \lor g \]
\[ g \mapsto f \]
\[ o_1 \mapsto c \]
\[ o_2 \mapsto g \]
\[ \} \]

scenarios (goals, side constraints):
1. \{\( o_1 \mapsto f \), \( o_2 \mapsto t \}\}, \{\( i_1 \mapsto t \}\}
2. \{\( a \mapsto t \}\}, ()

intervention strategies:
1. \{\( b \mapsto f \), \( d \mapsto f \), \( f \mapsto t \)\}

Logical steady state under 3-valued logic

1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint
Toy example

\[
\phi' = \begin{cases} 
  i_1 & \mapsto \top \\
i_2 & \mapsto i_2 \\
a & \mapsto \neg d \\
b & \mapsto \bot \\
c & \mapsto b \lor e \\
d & \mapsto \bot \\
e & \mapsto \neg i_1 \land i_2 \\
f & \mapsto \top \\
o_1 & \mapsto c \\
o_2 & \mapsto g \\
\end{cases}
\]

scenarios (goals, side constraints):
1. \(\{o_1 \mapsto f, o_2 \mapsto t\}\), \(\{i_1 \mapsto t\}\)
2. \(\{a \mapsto t\}, \emptyset\)

intervention strategies:
1. \(\{b \mapsto f, d \mapsto f, f \mapsto t\}\)

Logical steady state under 3-valued logic
1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint

true, false, undefined
Toy example

\[ \phi' = \begin{cases} 
  i_1 & \mapsto \top \\
  i_2 & \mapsto i_2 \\
  a & \mapsto \neg d \\
  b & \mapsto \bot \\
  c & \mapsto b \lor e \\
  d & \mapsto \bot \\
  e & \mapsto \neg i_1 \land i_2 \\
  f & \mapsto \top \\
  g & \mapsto f \\
  o_1 & \mapsto c \\
  o_2 & \mapsto g \\
\end{cases} \]

- scenarios (goals, side constraints):
  1. \{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\}
  2. \{a \mapsto t\}, \{\}

- intervention strategies:
  1. \{b \mapsto f, d \mapsto f, f \mapsto t\}

Logical steady state under 3-valued logic

1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint

true, false, undefined
Toy example

\[ \phi' = \left\{ \begin{align*}
    i_1 & \mapsto \top \\
    i_2 & \mapsto i_2 \\
    a & \mapsto \neg d \\
    b & \mapsto \bot \\
    c & \mapsto b \lor e \\
    d & \mapsto \bot \\
    e & \mapsto \neg i_1 \lor i_2 \\
    f & \mapsto \top \\
    g & \mapsto f
\end{align*} \right\} \]

- scenarios (goals, side constraints):
  1. \{ \( o_1 \mapsto f \), \( o_2 \mapsto t \) \}, \{ \( i_1 \mapsto t \) \}
  2. \{ \( a \mapsto t \) \}, \emptyset

- intervention strategies:
  1. \{ \( b \mapsto f \), \( d \mapsto f \), \( f \mapsto t \) \}

Logical steady state under 3-valued logic

1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint true, false, undefined
Toy example

\[ \phi' = \left\{ \begin{array}{c}
    i_1 \mapsto \top \\
    i_2 \mapsto i_2 \\
    a \mapsto \neg d \\
    b \mapsto \bot \\
    c \mapsto b \lor e \\
    d \mapsto \bot \\
    e \mapsto \neg i_1 \land i_2 \\
    f \mapsto \top \\
    g \mapsto f \\
    o_1 \mapsto c \\
    o_2 \mapsto g \\
\end{array} \right\} \]

- scenarios (goals, side constraints):
  1. \{\( o_1 \mapsto f \), \( o_2 \mapsto t \}\}, \{\( i_1 \mapsto t \)\}
  2. \{\( a \mapsto t \)\}, \emptyset

- intervention strategies:
  1. \{\( b \mapsto f \), \( d \mapsto f \), \( f \mapsto t \)\}

Logical steady state under 3-valued logic

1. “clamped” variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint

true, false, undefined
Toy example

\[ \phi' = \begin{cases} 
  i_1 & \mapsto & \top \\
  i_2 & \mapsto & i_2 \\
  a & \mapsto & \neg d \\
  b & \mapsto & \perp \\
  c & \mapsto & b \lor e \\
  d & \mapsto & \perp \\
  e & \mapsto & \neg i_1 \land i_2 \\
  f & \mapsto & \top \\
  o_1 & \mapsto & c \\
  o_2 & \mapsto & g \\
\end{cases} \]

- scenarios (goals, side constraints):
  1. \{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\}
  2. \{a \mapsto t\}, \emptyset

- intervention strategies:
  1. \{b \mapsto f, d \mapsto f, f \mapsto t\}

Logical steady state under 3-valued logic

1. "clamped" variables
2. set the remaining to \textit{undefined}
3. propagate until reaching the fixpoint
Toy example

\[ \phi'' = \left\{ \begin{array}{c}
i_1 & \mapsto & i_1 \\
i_2 & \mapsto & i_2 \\
a & \mapsto & \neg d \\
b & \mapsto & \bot \\
c & \mapsto & b \lor e \\
d & \mapsto & \bot \\
e & \mapsto & \neg i_1 \land i_2 \\
f & \mapsto & \top \\
o_1 & \mapsto & c \\
o_2 & \mapsto & g \\
\end{array} \right\} \]

- scenarios (goals, side constraints):
  1. \{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\}
  2. \{a \mapsto t\}, \emptyset

- intervention strategies:
  1. \{b \mapsto f, d \mapsto f, f \mapsto t\}

Logical steady state under 3-valued logic:

1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint

true, false, undefined
Toy example

\[ \phi'' = \left\{ \begin{array}{c}
i_1 \mapsto i_1 \\
i_2 \mapsto i_2 \\
a \mapsto \neg d \\
b \mapsto \bot \\
c \mapsto b \lor e \\
d \mapsto \bot \\
e \mapsto \neg i_1 \land i_2 \\
f \mapsto \top \\
g \mapsto f \\
o_1 \mapsto c \\
o_2 \mapsto g \\
\end{array} \right\} \]

- scenarios (goals, side constraints):
  1. \( \{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\} \)
  2. \( \{a \mapsto t\}, \emptyset \)

- intervention strategies:
  1. \( \{b \mapsto f, d \mapsto f, f \mapsto t\} \)

Logical steady state under 3-valued logic:
1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint

true, false, undefined
Toy example

\[ \phi'' = \left\{ \begin{array}{l}
i_1 \mapsto i_1 \\
i_2 \mapsto i_2 \\
a \mapsto \neg d \\
b \mapsto \bot \\
c \mapsto b \lor e \\
d \mapsto \bot \\
e \mapsto \neg i_1 \land i_2 \\
f \mapsto \top \\
o_1 \mapsto c \\
o_2 \mapsto g \\
\end{array} \right\} \]

- scenarios (goals, side constraints):
  1. \{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\}
  2. \{a \mapsto t\}, \emptyset

- intervention strategies:
  1. \{b \mapsto f, d \mapsto f, f \mapsto t\}

Logical steady state under 3-valued logic:

1. “clamped” variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint
Toy example

\[
\phi'' = \left\{ \begin{array}{ll}
  i_1 & \mapsto i_1 \\
  i_2 & \mapsto i_2 \\
  a & \mapsto \neg d \\
  b & \mapsto \bot \\
  c & \mapsto b \lor e \\
  d & \mapsto \bot \\
  e & \mapsto \neg i_1 \land i_2 \\
  f & \mapsto \top \\
  g & \mapsto f \\
  o_1 & \mapsto c \\
  o_2 & \mapsto g
\end{array} \right. 
\]

- scenarios (goals, side constraints):
  1. \{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\}
  2. \{a \mapsto t\}, \emptyset

- intervention strategies:
  1. \{b \mapsto f, d \mapsto f, f \mapsto t\}

Logical steady state under 3-valued logic

1. "clamped" variables
2. set the remaining to undefined
3. propagate until reaching the fixpoint
**Toy example**

\[ \phi = \begin{cases} 
  i_1 &\mapsto & i_1 \\
  i_2 &\mapsto & i_2 \\
  a &\mapsto & \neg d \\
  b &\mapsto & a \land i_1 \\
  c &\mapsto & b \lor e \\
  d &\mapsto & c \\
  e &\mapsto & \neg i_1 \land i_2 \\
  f &\mapsto & e \lor g \\
  g &\mapsto & f \\
  o_1 &\mapsto & c \\
  o_2 &\mapsto & g 
\end{cases} \]

- scenarios (*goals, side constraints*):
  1. \{o_1 \mapsto f , o_2 \mapsto t\}, \{i_1 \mapsto t\}
  2. \{a \mapsto t\}, \emptyset

- intervention strategies:
  1. \{b \mapsto f , d \mapsto f , f \mapsto t\}
  2. \{b \mapsto f , d \mapsto f , g \mapsto t\}
  3. \{b \mapsto f , e \mapsto f , f \mapsto t\}
  4. \{b \mapsto f , e \mapsto f , g \mapsto t\}
  5. \{b \mapsto f , i_2 \mapsto f , f \mapsto t\}
  6. \{b \mapsto f , i_2 \mapsto f , g \mapsto t\}
  7. \{c \mapsto f , f \mapsto t\}
  8. \{c \mapsto f , g \mapsto t\}
  9. \{c \mapsto f , e \mapsto t\}
Intervention strategies

Input:
- logical network
- side constraints
- goals

Output:
- all \( \subseteq \)-minimal intervention strategies

Samaga, R. et al. 2010

Approach:
- exhaustive breadth-first search
- search space reduction methods
- available as a MATLAB plugin

Limitations:
- discards admissible solutions
- poor scalability
- hard to maintain and extend

⇒ can we address this problem with Answer Set Programming?
Contents

1 Introduction

2 Answer Set Programming

3 Benchmarks

4 Conclusions
Instance representation

→ **Logical network** \((V, \phi)\) and candidate interventions \(X \subseteq V\)

variable(i1). variable(i2). variable(a). ...
candidate(i2). candidate(b). candidate(c). ...

→ **Mappings in** \(\phi\), e.g., \(a \mapsto \neg d, b \mapsto a \land i_1, f \mapsto e \lor g\)

formula(a,1). dnf(1,1). clause(1,d,-1).
formula(b,2). dnf(2,2). clause(2,a,1). clause(2,i1,1).
formula(f,3). dnf(3,3). clause(3,e,1).
    dnf(3,4). clause(4,g,1).

→ **Intervention scenarios** \(\{o_1 \mapsto f, o_2 \mapsto t\}, \{i_1 \mapsto t\}\); \(\{a \mapsto t\}, \emptyset\)

scenario(1).
constrained(1,i1,1). goal(1,o1,-1). goal(1,o2,1).
scenario(2).
goal(2,a,1).
Logic program

→ **Guess interventions (naive guessing)**

\[
\begin{align*}
0 \{ \text{intervention}(V,1;-1) \} & 1 : - \text{candidate}(V).
\end{align*}
\]

→ **Clamped and free variables**

\[
\begin{align*}
\text{eval}(Z,V,S) & : - \text{scenario}(Z), \text{intervention}(V,S). \\
\text{eval}(Z,V,S) & : - \text{constrained}(Z,V,S), \text{not intervention}(V). \\
\text{free}(Z,V,D) & : - \text{formula}(V,D), \text{scenario}(Z), \\
& \text{not constrained}(Z,V), \text{not intervention}(V).
\end{align*}
\]

→ **Propagate truth values according to Kleene’s three-valued logic**

\[
\begin{align*}
\text{eval}_\text{clause}(Z,C,-1) & : - \text{clause}(C,V,S), \text{eval}(Z,V,-S). \\
\text{eval}(Z,V,1) & : - \text{free}(Z,V,D), \text{dnf}(D,C), \\
& \text{eval}(Z,W,T) : \text{clause}(C,W,T). \\
\text{eval}(Z,V,-1) & : - \text{free}(Z,V,D), \text{eval}_\text{clause}(Z,C,-1) : \text{dnf}(D,C).
\end{align*}
\]

→ **Eliminate answer sets not being an intervention strategy**

\[
\begin{align*}
: - \text{goal}(Z,T,S), \text{not eval}(Z,T,S).
\end{align*}
\]
Potassco: Potsdam Answer Set Solving Collection

- gringo
- clasp
- claspD
- hclasp
- unclasp
- clingo
- iclingo
- oclingo

gringo + clasp: not enough to compute $\subseteq$-minimal answer sets.

Two extensions of clasp to address $\subseteq$-minimality

⇒ claspD: allows for solving disjunctive logic programs
⇒ hclasp: incorporates domain-specific heuristics into the language
Computing all \( \subseteq \)-minimal answer sets

\[ \texttt{gringo + claspD + meta-encodings from metasp}^1 \]

\#minimize \{ intervention(\_\_) \}.

\$ \texttt{gringo --reify encoding.lp instance.lp |\]
\texttt{gringo - meta.lp metaD.lp metaO.lp |}
\texttt{<(echo "optimize(1,1,incl).") | claspD 0}

\[ \texttt{gringo + hclasp + (heuristic-driven) solution recording}^2 \]

\texttt{_heuristic(intervention(V),false,1) :- candidate(V).}

\$ \texttt{gringo encoding.lp instance.lp | hclasp -e record 0}

\(^1\)http://www.cs.uni-potsdam.de/wv/metasp/
\(^2\)http://www.cs.uni-potsdam.de/hclasp/
Real-world problem instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Variables</th>
<th>Scenarios</th>
<th>Side Constraints</th>
<th>Goals</th>
<th>Candidate MISs</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGFR</td>
<td>103</td>
<td>1</td>
<td>28</td>
<td>2</td>
<td>$8.36 \times 10^{11}$</td>
</tr>
<tr>
<td>EGFR multiple</td>
<td>103</td>
<td>34</td>
<td>974</td>
<td>408</td>
<td>$4.63 \times 10^{32}$</td>
</tr>
<tr>
<td>TCR</td>
<td>94</td>
<td>1</td>
<td>17</td>
<td>12</td>
<td>$1.34 \times 10^{25}$</td>
</tr>
<tr>
<td>TBH6b</td>
<td>203</td>
<td>1</td>
<td>17</td>
<td>3</td>
<td>$6.03 \times 10^{47}$</td>
</tr>
</tbody>
</table>

3 instances from (Samaga, R. et al. 2010)

- limited to intervention strategies of up to size 3
- good performance ($\leq 2$ min) but poor scalability

plus 1 larger unpublished instance (TBH6b).

Using ASP could we...

- ... find larger intervention strategies?
- ... solve the unbounded problem?
- ... scale to larger logical networks?
## Performance

<table>
<thead>
<tr>
<th>k</th>
<th>EGFR</th>
<th>EGFR multiple</th>
<th>TCR</th>
<th>TBH6b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$l_{\text{claspD}}$</td>
<td>$h_{\text{clasp}}$</td>
<td>$l_{\text{claspD}}$</td>
<td>$h_{\text{clasp}}$</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0.02</td>
<td>0</td>
<td>0.86</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>0.11</td>
<td>0</td>
<td>0.93</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>0.13</td>
<td>8</td>
<td>1.15</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>0.14</td>
<td>38</td>
<td>11.16</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>0.17</td>
<td>74</td>
<td>41.57</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>0.16</td>
<td>83</td>
<td>73.14</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>0.16</td>
<td>83</td>
<td>90.66</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>0.16</td>
<td>83</td>
<td>90.56</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>0.18</td>
<td>83</td>
<td>88.95</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
<td>0.19</td>
<td>83</td>
<td>84.60</td>
</tr>
<tr>
<td>∞</td>
<td>21</td>
<td>0.13</td>
<td>83</td>
<td>91.83</td>
</tr>
</tbody>
</table>

Using ASP we can...

- ... find larger intervention strategies ✓
- ... solve the unbounded problem ✓
- ... scale to larger logical networks ✓
Contents

1. Introduction
2. Answer Set Programming
3. Benchmarks
4. Conclusions
Contributions

- Characterization of the *Minimal Intervention Strategies* problem relying on Kleene’s three-valued logic and fixpoint semantics
- ASP encoding: http://potassco.sf.net/apps.html#interventions
- ⊆-minimality addressed with *claspD* or *hclasp*
- Benchmarking using real-world instances
- ASP greatly outperforms the existing dedicated algorithm
- *hclasp* + solution recording outperforms *claspD* + metasp
Future work

- Computational complexity
- Performance subject to:
  - # of side constraints
  - # of goals and their location in the network
  - # of intervention scenarios
  - topological properties of the network
- Improvements to the ASP encoding
- How to turn this deluge of MIS into insights for biologists?
  - Exploit solvers’ features like cautious and brave reasoning
  - Extend the problem to a family of logical networks
  - Incorporate more biological knowledge constraining the search
Acknowledgements

Roland Kaminski
Steffen Klamt

Torsten Schaub
Axel Vom Kamp

Anne Siegel
Regina Samaga
Thank you
Improved guessing

satisfy(V,W,S) :- formula(W,D), dnf(D,C), clause(C,V,S).
closure(V,T)  :- goal(V,T).
closure(V,S*T) :- closure(W,T), satisfy(V,W,S),
    not goal(V,-S*T).

{ intervention(V,S) : closure(V,S) : candidate(V) }.
:- intervention(V,1), intervention(V,-1).
Grounding the naive guessing

0#count{intervention(g,-1),intervention(g,1)}1.
0#count{intervention(f,-1),intervention(f,1)}1.
0#count{intervention(e,-1),intervention(e,1)}1.
0#count{intervention(d,-1),intervention(d,1)}1.
0#count{intervention(c,-1),intervention(c,1)}1.
0#count{intervention(b,-1),intervention(b,1)}1.
0#count{intervention(i2,-1),intervention(i2,1)}1.

⇒ $3^7 = 2187$ candidate intervention strategies
Grounding the improved guessing

closure(o1,-1). closure(o2,1). closure(i1,-1).
closure(d,-1). closure(i2,-1). closure(c,-1).
closure(g,1). closure(b,-1). closure(f,1). closure(i2,1).
closure(e,1). closure(e,-1). closure(a,1). closure(i1,1).

#count{intervention(i2,1),intervention(i2,-1),
  intervention(e,1), intervention(f,1),
  intervention(e,-1),intervention(b,-1),
  intervention(c,-1),intervention(g,1),
  intervention(d,-1)}.

:-intervention(e,1),intervention(e,-1).
:-intervention(i2,1),intervention(i2,-1).

⇒ $3^2 \times 2^5 = 288$ candidate intervention strategies