Complexity

Juri Kolčák

Friday 30th January, 2026

Decision Problems

To formally reason about computational complexity of the dynamical analysis of Boolean networks, we phrase determining the most basic dynamical properties, e.g. fixed points, reachability, limit cycles, as **decision problems**: problems that answer yes or no.

Time (space) complexity of a decision problem for a given input is the number of computation steps (resp. bits of memory) a procedure computing it needs to reach and output an answer.

Instead of complexity for each individual input, we want an upper bound ("worst-case") for families of inputs of the same size. This upper bound is typically expressed **asymptotically**, as a function towards which it tends as the size of the input approaches infinity.

Complexity Classes

The exact asymptotic complexity is typically only considered in specific algorithm design (e.g. matrix multiplication).

For comparison between the complexity of different decision problems, broader **complexity classes** are used.

- **P**, class of all decision problems which can be solved by an algorithm running in polynomial time.
- **NP**, ... in polynomial time with non-deterministic choices.
- **PSPACE**, ... in polynomial space.

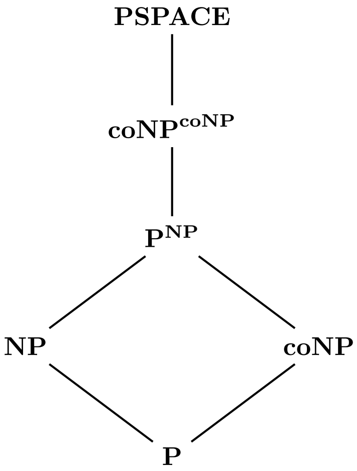
Complexity Classes

The exact asymptotic complexity is typically only considered in specific algorithm design (e.g. matrix multiplication).

For comparison between the complexity of different decision problems, broader **complexity classes** are used.

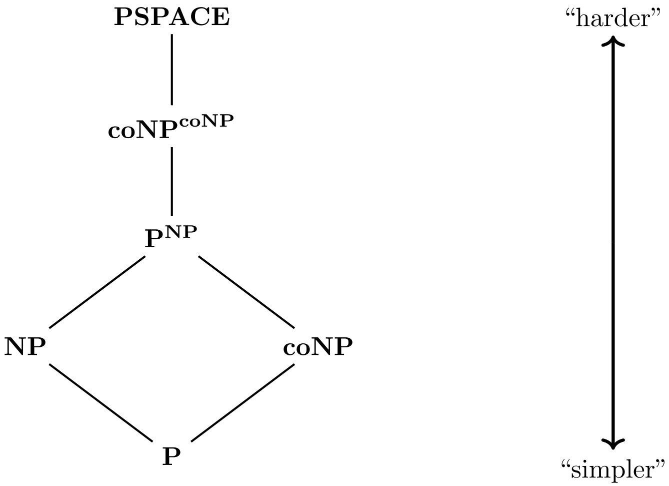
- **P**, class of all decision problems which can be solved by an algorithm running in polynomial time.
- **NP**, ... in polynomial time with non-deterministic choices.
- **PSPACE**, ... in polynomial space.
- **coNP**,... for which finding a counter-example is in **NP**.
- $\mathbf{P}^{\mathbf{NP}}$, ... in polynomial time with an **NP** oracle.
- $\mathbf{coNP}^{\mathbf{coNP}}$, ... counter-example is in **NP** with a **coNP** oracle.

Complexity Classes Inclusion



Hasse diagram of complexity classes ordered by set inclusion.

Complexity Classes Inclusion



Hasse diagram of complexity classes ordered by set inclusion.

Reduction

A **polynomial-time reduction** (Karp reduction) from a decision problem A to a decision problem B is a polynomial-time algorithm that transforms problem A inputs into problem B inputs, such that the answer is the same.

"Problem A is at most as complex as problem B."

"Problem B cannot be simpler than problem A."

Reduction

A **polynomial-time reduction** (Karp reduction) from a decision problem A to a decision problem B is a polynomial-time algorithm that transforms problem A inputs into problem B inputs, such that the answer is the same.

"Problem A is at most as complex as problem B."

"Problem B cannot be simpler than problem A."

A problem A is **hard** for a complexity class $\mathbf{C}$ if every problem in $\mathbf{C}$ can be reduced to A in polynomial time.

A problem A is **complete** for a complexity class $\mathbf{C}$ if $A \in \mathbf{C}$ and A is $\mathbf{C}$-hard.

The Boolean Network Case

The dynamical properties of interest (reachability, attractors) reduce to simple graph algorithms on the transition system.

The transitions system, however, is of size $\mathcal{O}(2^n)$ where n is the dimension of the Boolean network, both the number of nodes and the number of edges.

Graph algorithms on the transition system thus give as an **EXPTIME** upper bound.

The size of our input is determined by the size Boolean network f, which is dependant on the encoding used. We assume a symbolic encoding, e.g. propositional formulae or BDDs, as opposed to truth tables ($\mathcal{O}(2^n)$).

We further assume that for any configuration $\mathbf{x} \in \mathbb{B}^n$, $f(\mathbf{x})$ can be evaluated in linear time in the size of f.

Fixed Points

Given a Boolean network f of dimension n, deciding if there exists a configuration $\mathbf{x} \in \mathbb{B}^n$ such that $f(\mathbf{x}) = \mathbf{x}$ is **NP**-complete.

Fixed Points

Given a Boolean network f of dimension n, deciding if there exists a configuration $\mathbf{x} \in \mathbb{B}^n$ such that $f(\mathbf{x}) = \mathbf{x}$ is **NP**-complete.

The belonging to **NP** follows directly from evaluating $f(\mathbf{x})$ being linear. Given a witness $\mathbf{x}$, confirming it is a fixed point is polynomial.

Fixed Points

Given a Boolean network f of dimension n, deciding if there exists a configuration $\mathbf{x} \in \mathbb{B}^n$ such that $f(\mathbf{x}) = \mathbf{x}$ is **NP**-complete.

The belonging to **NP** follows directly from evaluating $f(\mathbf{x})$ being linear. Given a witness $\mathbf{x}$, confirming it is a fixed point is polynomial.

The hardness follows by a reduction from the `3SAT` problem.

Let Φ be an input formula for the `3SAT` problem, that is a propositional formula in CNF with exactly three literals per clause.
Let $V(\Phi)$ be the set of all variables and $C(\Phi)$ the set of all clauses of Φ.

For each variable $v \in V(\Phi)$, the BN contains a variable with $f_v(\mathbf{x}) = \mathbf{x}_v$.

For each clause $c \in C(\Phi)$, the BN contains a variable with $f_c(\mathbf{x}) = \neg \mathbf{x}_c \vee c$ where for each $v \in V(\Phi)$, all occurrences of v are replaced by $\mathbf{x}_v$ within c.

Fixed Points – Software

Pint

L. Paulevé. Pint: A static analyzer for transient dynamics of qualitative networks with ipython interface.

In J. Feret and H. Koeppl, editors, *Computational Methods in Systems Biology*, pages 309–316, Cham, Switzerland, September 2017. Springer International Publishing

Uses `SAT` solvers to enumerate fixed points.

GINsim

A. Naldi, D. Thieffry, and C. Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks.

In M. Calder and S. Gilmore, editors, *Computational Methods in Systems Biology*, pages 233–247, Berlin, Heidelberg, 2007. Springer

A. Naldi, C. Hernandez, W. Abou-Jaoudé, P. T. Monteiro, C. Chaouiya, and D. Thieffry. Logical modeling and analysis of cellular regulatory networks with ginsim 3.0.

Frontiers in Physiology, Volume 9 - 2018, 2018

Fixed points are expressed symbolically as paths within decision diagrams.

Reachability

Given a Boolean network f of dimension n, classical semantics $s \in \{sync, async, gen\}$, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{s}^* \mathbf{y}$ is **PSPACE**-complete.

Reachability

Given a Boolean network f of dimension n, classical semantics $s \in \{sync, async, gen\}$, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{s}^* \mathbf{y}$ is **PSPACE**-complete.

The belonging to **PSPACE** follows from the size of the transition system. If $\mathbf{y}$ is reachable from $\mathbf{x}$, then it is reachable in at most $2^n - 1$ steps. The algorithm therefore only needs to remember the current configuration and a step counter, which goes up to at most $2^n - 1$, and thus needs at most n bits.

Reachability

Given a Boolean network f of dimension n, classical semantics $s \in \{sync, async, gen\}$, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{s}^* \mathbf{y}$ is **PSPACE**-complete.

The belonging to **PSPACE** follows from the size of the transition system. If $\mathbf{y}$ is reachable from $\mathbf{x}$, then it is reachable in at most $2^n - 1$ steps. The algorithm therefore only needs to remember the current configuration and a step counter, which goes up to at most $2^n - 1$, and thus needs at most n bits.

The hardness for the synchronous semantics follows from a reduction of a **PSPACE**-complete problem "Does a deterministic Turing machine accept an input word without using more than $k \in \mathbb{N}$ tape cells?" into reachability in **reaction systems**, which are a special case of synchronous Boolean networks.

A. Dennunzio, E. Formenti, L. Manzoni, and A. E. Porreca. Complexity of the dynamics of reaction systems.
Information and Computation, 267:96–109, 2019

Asynchronous Reachability

Given a Boolean network f of dimension n, we build a Boolean network f' of dimension $3n+2$ as follows:

$$f'_i(\mathbf{x}') = ((\neg \mathbf{x}'_w \vee \mathbf{x}'_z)' \wedge \mathbf{x}'_i) \vee (\mathbf{x}'_w \wedge \neg \mathbf{x}'_z \wedge \mathbf{x}'_{ci})$$
$$f'_{ci}(\mathbf{x}') = \neg \mathbf{x}'_z \wedge ((\neg \mathbf{x}'_w \wedge f_i(\mathbf{x}'_{[1,\dots,n]}) \vee (\mathbf{x}'_w \wedge \mathbf{x}'_{ci}))$$
$$f'_{\overline{c}i}(\mathbf{x}') = \neg \mathbf{x}'_z \wedge ((\neg \mathbf{x}'_w \wedge \neg f_i(\mathbf{x}'_{[1,\dots,n]})) \vee (\mathbf{x}'_w \wedge \mathbf{x}'_{\overline{c}i}))$$
$$f'_w(\mathbf{x}') = \neg \mathbf{x}'_z \wedge \left(\mathbf{x}'_w \vee \bigwedge_{i \in \{1,\dots,n\}} (\mathbf{x}'_{ci} \vee \mathbf{x}'_{\overline{c}i}) \right)$$
$$f'_z(\mathbf{x}') = \left(\mathbf{x}'_w \wedge \bigwedge_{i \in \{1,\dots,n\}} (\mathbf{x}'_{ci} \Leftrightarrow \mathbf{x}'_i \wedge \mathbf{x}'_{\overline{c}i} \Leftrightarrow \neg \mathbf{x}'_i) \right) \vee$$
$$\left(\mathbf{x}'_z \wedge \left(\mathbf{x}'_w \vee \bigvee_{i \in \{1,\dots,n\}} (\mathbf{x}'_{ci} \vee \mathbf{x}'_{\overline{c}i}) \right) \right)$$

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{B}^n \colon \mathbf{x} \xrightarrow[f]{sync}{}^* \mathbf{y} \iff \mathbf{x}\mathbf{0}^{2n+2} \xrightarrow[f']{async}{}^* \mathbf{y}\mathbf{0}^{2n+2} \iff \mathbf{x}\mathbf{0}^{2n+2} \xrightarrow[f']{gen}{}^* \mathbf{y}\mathbf{0}^{2n+2}$$

Most Permissive Reachability

Given a Boolean network f of dimension n, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{mp}{}^{*} \mathbf{y}$ is in $\mathbf{P}^{\mathbf{NP}}$, and is in $\mathbf{P}$ in case f is locally monotone.

Most Permissive Reachability

Given a Boolean network f of dimension n, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{mp}{}^{*} \mathbf{y}$ is in $\mathbf{P}^{\mathbf{NP}}$, and is in $\mathbf{P}$ in case f is locally monotone.

MP reachability has a three-phase witness $\mathbf{x} \xrightarrow{(1)}{}^{*} \hat{\mathbf{x}} \xrightarrow{(2)}{}^{*} \hat{\mathbf{y}} \xrightarrow{(3)}{}^{*} \mathbf{y}$ of length at most $3n$ where variables only change value from Boolean to transient (1), between the transient values (2), from transient to Boolean (3).

Most Permissive Reachability

Given a Boolean network f of dimension n, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{mp}^* \mathbf{y}$ is in $\mathbf{P}^{\mathbf{NP}}$, and is in $\mathbf{P}$ in case f is locally monotone.

MP reachability has a three-phase witness $\mathbf{x} \xrightarrow{(1)}^* \hat{\mathbf{x}} \xrightarrow{(2)}^* \hat{\mathbf{y}} \xrightarrow{(3)}^* \mathbf{y}$ of length at most $3n$ where variables only change value from Boolean to transient (1), between the transient values (2), from transient to Boolean (3).

Consider the following algorithm:

```
1:  $W \leftarrow \{1, \dots, n\}$
2:  while true do
3:      $\mathbf{d} \leftarrow \mathbf{x}$
4:      for $\_i \in W$ do
5:          for $i \in \{j \in W \mid \mathbf{d}_j \in \mathbb{B}\}$ do
6:              if $\exists \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) \neq \mathbf{d}_i$ then
7:                  $\mathbf{d}_i \leftarrow *$
8:      $\mathbf{d}' \leftarrow \mathbf{d}; W' \leftarrow W$
9:      for $i \in \{j \in W \mid \mathbf{d}_j \notin \mathbb{B}\}$ do
10:         if $\forall \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) \neq \mathbf{x}_i$ then
11:             if $\mathbf{x}_i = \mathbf{y}_i$ then
12:                 $W' \leftarrow W \setminus \{i\}$; break
13:             $\mathbf{d}'_i \leftarrow 1 - \mathbf{x}_i$
14:     if $W = W'$ then return $\mathbf{y} \in \mathbf{d}'$
15:     else
16:         $W \leftarrow W'$
```

Most Permissive Reachability

Given a Boolean network f of dimension n, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{mp}{}^* \mathbf{y}$ is in $\mathbf{P}^{\mathbf{NP}}$, and is in $\mathbf{P}$ in case f is locally monotone.

MP reachability has a three-phase witness $\mathbf{x} \xrightarrow{(1)}{}^* \hat{\mathbf{x}} \xrightarrow{(2)}{}^* \hat{\mathbf{y}} \xrightarrow{(3)}{}^* \mathbf{y}$ of length at most $3n$ where variables only change value from Boolean to transient (1), between the transient values (2), from transient to Boolean (3).

Consider the following algorithm:

```
 1: $W \leftarrow \{1, \dots, n\}$
 2: while true do
 3:     $\mathbf{d} \leftarrow \mathbf{x}$
 4:     for $\_i \in W$ do
 5:         for $i \in \{j \in W \mid \mathbf{d}_j \in \mathbb{B}\}$ do
 6:             if $\exists \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) \neq \mathbf{d}_i$ then
 7:                 $\mathbf{d}_i \leftarrow *$
 8:     $\mathbf{d}' \leftarrow \mathbf{d}$; $W' \leftarrow W$
 9:     for $i \in \{j \in W \mid \mathbf{d}_j \notin \mathbb{B}\}$ do
10:         if $\forall \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) \neq \mathbf{x}_i$ then
11:             if $\mathbf{x}_i = \mathbf{y}_i$ then
12:                 $W' \leftarrow W \setminus \{i\}$; break
13:             $\mathbf{d}'_i \leftarrow 1 - \mathbf{x}_i$
14:     if $W = W'$ then return $\mathbf{y} \in \mathbf{d}'$
15:     else
16:         $W \leftarrow W'$
```

The outer while loop (2) is run at most n times as the size of W decreases in each iteration (16).

Each for loop $(4, 5, 9)$ is also iterated at most n times, giving us polynomial $(\mathcal{O}(n^3))$ number of calls of the type $\exists \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) = 1$, an instance of the SAT problem.

Most Permissive Reachability

Given a Boolean network f of dimension n, and two configurations $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, deciding if $\mathbf{x} \xrightarrow{mp}{}^* \mathbf{y}$ is in $\mathbf{P}^{\mathbf{NP}}$, and is in $\mathbf{P}$ in case f is locally monotone.

MP reachability has a three-phase witness $\mathbf{x} \xrightarrow{(1)}{}^* \hat{\mathbf{x}} \xrightarrow{(2)}{}^* \hat{\mathbf{y}} \xrightarrow{(3)}{}^* \mathbf{y}$ of length at most $3n$ where variables only change value from Boolean to transient (1), between the transient values (2), from transient to Boolean (3).

Consider the following algorithm:

```
1:  $W \leftarrow \{1, \dots, n\}$
2:  while true do
3:      $\mathbf{d} \leftarrow \mathbf{x}$
4:      for $\_i \in W$ do
5:          for $i \in \{j \in W \mid \mathbf{d}_j \in \mathbb{B}\}$ do
6:              if $\exists \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) \neq \mathbf{d}_i$ then
7:                  $\mathbf{d}_i \leftarrow *$
8:      $\mathbf{d}' \leftarrow \mathbf{d}$; $W' \leftarrow W$
9:      for $i \in \{j \in W \mid \mathbf{d}_j \notin \mathbb{B}\}$ do
10:         if $\forall \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) \neq \mathbf{x}_i$ then
11:             if $\mathbf{x}_i = \mathbf{y}_i$ then
12:                 $W' \leftarrow W \setminus \{i\}$; break
13:             $\mathbf{d}'_i \leftarrow 1 - \mathbf{x}_i$
14:     if $W = W'$ then return $\mathbf{y} \in \mathbf{d}'$
15:     else
16:         $W \leftarrow W'$
```

The outer while loop (2) is run at most n times as the size of W decreases in each iteration (16).

Each for loop $(4, 5, 9)$ is also iterated at most n times, giving us polynomial $(\mathcal{O}(n^3))$ number of calls of the type $\exists \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) = 1$, an instance of the SAT problem.

For locally monotone f, $\exists \mathbf{z} \in \mathbf{d}, f_i(\mathbf{z}) = 1$ has a polynomial solution, the "maximal parameter".

Reachability – Software

NuSMV

A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking.

In E. Brinksma and K. G. Larsen, editors, *Computer Aided Verification*, pages 359–364, Berlin, Heidelberg, 2002. Springer

Model checker for discrete finite dynamical systems.

Boolean networks can be exported in the correct format from GINsim.

Pint offers static analysis tools that can reduce the number of transitions.

mpbn

L. Paulevé, J. Kolčák, T. Chatain, and S. Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks.

Nature Communications, 11(1):4256, 08 2020

Tool for most permissive reachability and attractor analysis.

Cyclic Attractors

Given a Boolean network f of dimension n, classical semantics $s \in \{sync, async, gen\}$, and a configuration $\mathbf{x} \in \mathbb{B}^n$, deciding if $\mathbf{x}$ belongs to a cyclic attractor with the semantics s is **PSPACE**-complete.

Cyclic Attractors

Given a Boolean network f of dimension n, classical semantics $s \in \{sync, async, gen\}$, and a configuration $\mathbf{x} \in \mathbb{B}^n$, deciding if $\mathbf{x}$ belongs to a cyclic attractor with the semantics s is **PSPACE**-complete.

Follows directly from the complexity of the reachability problem.

In particular, $\mathbf{x}$ is in an attractor if it is reachable from every $\mathbf{y}$ reachable from $\mathbf{x}$.

Inversely, $\mathbf{x}$ is not in an attractor if there exist a configuration $\mathbf{y}$ reachable from $\mathbf{x}$, from which $\mathbf{x}$ isn't reachable.

To verify the inverse, we non-deterministically choose $\mathbf{y}$ and verify the non-reachability of $\mathbf{x}$, giving us **NPSPACE**, thus the original problem is in **coNPSPACE** = **PSPACE**.

Cyclic Attractors

Given a Boolean network f of dimension n, classical semantics $s \in \{sync, async, gen\}$, and a configuration $\mathbf{x} \in \mathbb{B}^n$, deciding if $\mathbf{x}$ belongs to a cyclic attractor with the semantics s is **PSPACE**-complete.

Follows directly from the complexity of the reachability problem.

In particular, $\mathbf{x}$ is in an attractor if it is reachable from every $\mathbf{y}$ reachable from $\mathbf{x}$.

Inversely, $\mathbf{x}$ is not in an attractor if there exist a configuration $\mathbf{y}$ reachable from $\mathbf{x}$, from which $\mathbf{x}$ isn't reachable.

To verify the inverse, we non-deterministically choose $\mathbf{y}$ and verify the non-reachability of $\mathbf{x}$, giving us **NPSPACE**, thus the original problem is in **coNPSPACE** = **PSPACE**.

The hardness is shown in the same way as for reachability, starting with the results for reaction systems.

Most Permissive Cyclic Attractors

Given a Boolean network f of dimension n, and a configuration $\mathbf{x} \in \mathbb{B}^n$, deciding if $\mathbf{x}$ belongs to a cyclic attractor with most permissive semantics is in $\mathbf{coNP}^{\mathbf{coNP}}$, and is in **coNP** in case f is locally monotone.

Most Permissive Cyclic Attractors

Given a Boolean network f of dimension n, and a configuration $\mathbf{x} \in \mathbb{B}^n$, deciding if $\mathbf{x}$ belongs to a cyclic attractor with most permissive semantics is in $\mathbf{coNP}^{\mathbf{coNP}}$, and is in **coNP** in case f is locally monotone.

Recall that attractors of most permissive Boolean networks are exactly the minimal trap spaces. Given a subspace $\mathbf{d} \in \{0, 1, *\}^n$, one therefore has to verify two properties:

- whether $\mathbf{d}$ is closed by f, $\forall \mathbf{x} \in \mathbf{d}, f(\mathbf{x}) \in \mathbf{d}$;
- and whether $\mathbf{d}$ is minimal, $\forall \mathbf{d}' \subseteq \mathbf{d}$, $\mathbf{d}'$ is not closed by f;

Let us consider the inverse problems for a given Boolean network f of dimension n and subspace $\mathbf{d} \in \{0, 1, *\}^n$, `IsNotClosed` and `IsNotMinimal`.

Closed and Minimal Subspaces

`IsNotClosed` is equivalent to determining if there exists a variable $i \in \{1, \ldots, n\}$ with $\mathbf{d}_i \neq *$ and a configuration $\mathbf{x} \in \mathbf{d}$ such that $f_i(\mathbf{x}) \neq \mathbf{d}_i$.

This is an instance of the `SAT` problem and is thus in **NP**, and in **P** in case f is locally monotone.
The inverse problem `IsClosed` is then in **coNP**, respectively in **coP** $=$ **P**.

Closed and Minimal Subspaces

`IsNotClosed` is equivalent to determining if there exists a variable $i \in \{1, \ldots, n\}$ with $\mathbf{d}_i \neq *$ and a configuration $\mathbf{x} \in \mathbf{d}$ such that $f_i(\mathbf{x}) \neq \mathbf{d}_i$.

This is an instance of the `SAT` problem and is thus in **NP**, and in **P** in case f is locally monotone.
The inverse problem `IsClosed` is then in **coNP**, respectively in $\mathbf{coP} = \mathbf{P}$.

`IsNotMinimal` consists of finding a subspace $\mathbf{d}' \subseteq \mathbf{d}$ which is closed by f. `IsNotMinimal` thus uses `IsClosed` as a subroutine, or an "oracle", and is in $\mathbf{NP}^{\texttt{IsClosed}}$.

The inverse problem `IsMinimal` is then in $\mathbf{coNP}^{\texttt{IsClosed}}$, which is $\mathbf{coNP}^{\mathbf{coNP}}$, or $\mathbf{coNP}^{\mathbf{P}} = \mathbf{coNP}$ in case f is locally monotone.

Cyclic Attractors – Software

BNS

E. Dubrova and M. Teslenko. A sat-based algorithm for finding attractors in synchronous boolean networks.

IEEE/ACM Transactions on Computational Biology and Bioinformatics, 8(5):1393–1399, Jul 2011

Uses `SAT` solvers to enumerate attractors of synchronous Boolean networks.

BoolSim

A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks.

Bioinformatics, 24(17):1917–1925, Jul 2008

Cabean

C. Su and J. Pang. Cabean: a software for the control of asynchronous boolean networks.

Bioinformatics, 37(6):879–881, Aug 2020

Uses symbolic representation of the reachable configurations by decision diagrams to enumerate attractors. BoolSim handles both synchronous and fully asynchronous cases. Cabean is fully asynchronous specific.

mpbn for most permissive, same as reachability.