If you see a formula in the Physical Review that extends over a quarter of a page, forget it. It's wrong. Nature isn't that complicated.
— Bernd T Matthias.
Influence Diagrams

What do they mean?
Usually NOTHING.

Is EGFR regulated, shut down, or oscillating, by the negative feedback loop?

Is this an AND or an OR?

How can this positive feedback loop ever reset once started?
Nonetheless, the basic idea of influence diagrams can be cast as an alternative notation for automata.
Influence Transitions

5 basic influence transitions

- Influence between stems
- Excitation between pole and stem
- Inhibition between pole and stem
- Accretion on a stem
- Degradation on a stem

Plus 2 abbreviations for self-loops

- Node
- Pole (self-loop)
- Stem (unique arc between two nodes or poles)
- Current node or pole (unique in each automaton)
Influence diagrams where the only two-ended influences are excitation and inhibition between poles and stems, are called **POLIN DIAGRAMS**.

It is convenient to name poles: those names correspond to the channel names in automata. (It does not seem critical to name other nodes.)

By convention, then, equally named poles are always equally connected (otherwise they would not correspond to channel names).

By definition, each two-ended influence connects separate automata.

A population of 3 automata

But we often represent population schemas by two-ended influences within the same diagram:

A population schema for a population of size n of such automata

Still, a two-ended influence is always intended between two separate automata.
- Each node in a polin becomes a node in an automaton.

- Each pole becomes an output self-transition in the automaton, with the same name.

- Each pole-to-stem connection becomes an input transition in the automaton between the stem nodes (reverse transition if inhibition). The name of the transition comes from the name of the source pole.

- Accretion/degradation arcs, become delays in the appropriate direction.

- Multiple connections on a single stem become multiple transitions between nodes.
Monopolins
A monopolin may have one or more poles, but all such poles are named with a single name. Other nodes are unnamed.

Nodes are connected by oriented stems.

Activation and inhibition arcs connect poles to stems.

The orientation of a stem can be omitted when clear by convention (activation is then always towards a pole, and inhibition away from it).

One node can be marked as current (red) to indicate the current state of a specific polin instance.

Names that may appear on arcs do not belong to the arcs: they simply indicate that the arc comes from some pole with that name.
More Monopolins

“and-up/and-dn”

“and-up/or-dn”

“and-up/or-dn”

two copies of the “same” pole (poles with the same name must have exactly the same outgoing arcs)

(each of the !hi outputs obviously connects to all the ?hi transitions anywhere in the network)
This simple increment/decrement idea can actually give good results, if done carefully:

But **CAVEAT EMPTOR:** influence diagrams in biology are not meant to convey semantics!

---

**Faithful Modeling of Transient Behavior in Developmental Pathways**

Amir Rubinstein\(^1\), Vyacheslav Gurevich\(^2\), Yona Kassir\(^2\) and Ron Y. Pinter\(^1\)

\(^1\)Dept. of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel
\(^2\)Dept. of Biology, Technion – Israel Institute of Technology, Haifa 32000, Israel

---

Fig. 2. The working hypothesis network describing the relationship between the expression of IME1 and IME2.
Amplifiers
Amplifiers

100*b with n*a vs 10*deg

100*b with n*a vs 10*deg

linear

hysteresis
Basic Excitation Cascade

Abstracting a little library of composable monopolin components

\[
\text{Amp}_{hi}(hi,up,dn) = \\
\quad !hi; \quad \text{Amp}_{hi}(hi,up,dn) \\
\quad + ?dn; \quad \text{Amp}_{lo}(hi,up,dn)
\]

\[
\text{Amp}_{lo}(hi,up,dn) = \\
\quad ?up; \quad \text{Amp}_{hi}(hi,up,dn)
\]

c raises much faster than b 
C.f. MAPK cascade

directive sample 3.0 10000
directive plot a@1.0:chan new b@1.0:chan new c@1.0:chan

let A() = !a; A()
let B1() = ?a; B2() and B2() = !b; B2()
let C1() = ?b; C2() and C2() = !c; C2()
run 1 of A() run 100 of B1() run 100 of C1()
When competing with degradation, the a signal (very weak) is not able to fully raise b. However, c is still raised.

Double excitation seems to make the off response a bit quicker.

Triple excitation
Double Excitation and Hysteresis

![Graphs showing double excitation and hysteresis with linear, sublinear, more sublinear, and superlinear behaviors.](image)

**Linear**
- Fast Change: more hysteresis
- t=10

**Sublinear**
- Slow Change: less hysteresis
- t=100

**More Sublinear**
- Fast Change: more hysteresis
- t=10

**Superlinear**
- Slow Change: less hysteresis
- t=100

*Note: The graphs illustrate the relationship between excitation and hysteresis, with different behaviors depending on the time constant (t) and the type of excitation (linear, sublinear, more sublinear, superlinear).*
Excitation Cascade with Degradation

100*b vs 10*deg

100*b,c vs 20*deg

100*a,b,c vs 30*deg

100*b,c vs 10*deg

100*a,b,c vs 10-aeg

no "sigma" response
Double Excitation Cascade with Degradation

100*b vs 10*deg

TIME VS D

100*b,c vs 20*deg

"sigma" response due to Erlang process
Multistables and Oscillators
Each stimulates self and inhibits others

```plaintext
directive sample 5.0 10000
directive plot !a; !b
new a@1.0:chan new b@1.0:chan
let A_hi() = do !a; A_hi() or ?b; A_lo() and A_lo() = ?a; A_hi()
let B_hi() = do !b; B_hi() or ?a; B_lo() and B_lo() = ?b; B_hi()
run 100 of (A_hi() | B_hi())
```

```plaintext
directive sample 5.0 10000
directive plot !a; !b; !c
new a@1.0:chan new b@1.0:chan new c@1.0:chan
let A_hi() = do !a; A_hi() or ?b; A_lo() or ?c; A_lo() and A_lo() = ?a; A_hi()
let B_hi() = do !b; B_hi() or ?c; B_lo() or ?a; B_lo() and B_lo() = ?b; B_hi()
let C_hi() = do !c; C_hi() or ?a; C_lo() or ?b; C_lo() and C_lo() = ?c; C_hi()
run 50 of (A_hi() | B_hi() | C_hi())
```
Mulstistables with Noise

let A_hi() = do !a; A_hi() or ?b; A_lo() or ?c; A_lo() or delay@noise; A_lo()
and A_lo() = do ?a; A_hi() or delay@noise; A_hi()

let B_hi() = do !b; B_hi() or ?c; B_lo() or ?a; B_lo() or delay@noise; B_lo()
and B_lo() = do ?b; B_hi() or delay@noise; B_hi()

let C_hi() = do !c; C_hi() or ?a; C_lo() or ?b; C_lo() or delay@noise; C_lo()
and C_lo() = do ?c; C_hi() or delay@noise; C_hi()

run 100 of (A_hi() | B_hi() | C_hi())
Each stimulates the next and inhibits the previous.
Inverters
Good logic needs a good inverter

...that alternates in cascades

...that oscillates in odd cycles

Pushup Inverter

fiddling with rates does not seem to change the picture

no oscillation

poor alternation

no hysteresis
Pullup Inverter (deadlocking low)
Pushup/Pullup Inverter

great alternation

great rectification

poor alternation

no rectification

no oscillation

no hysteresis
Boolean Gates
A "monopolin signal" consists of either the presence of a certain pole (designated "hi") in state current, or the absence of such a pole.
Boolean Gates: The Automata View

\[ c = a \text{ or } b \]
\[ c = a \text{ and } b \]
\[ c = a \text{ imply } b \]
\[ c = a \text{ unless } b \]
\[ c = a \text{ xor } b \]

Inputs:
10 \( l_a \) for \( 4t \)
2t; 10 \( l_b \) for \( 4t \)
Boolean Gates: The Automata View

\[ c = a \text{ nor } b \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ c = a \text{ nand } b \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ c = b \text{ unless } a \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ c = b \text{ imply } a \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ c = a \text{ iff } b \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ !c \]

\[ ?a \]

\[ ?b \]

\[ !c \]

\[ ?a \]

\[ ?b \]
Xor and OpAmp
Xor in Detail

```
directive sample 10.0 1000
let clock(t:float, tick:chan) =      (* sends a tick every t time *)
  (val ti = t/200.0 val d = 1.0/ti
   let step(n:int) = if n<=0 then !tick; clock(t, tick) else delay@d; step(n-1)
   run step(200))
```

```
let S_a(tick:chan) = do !a; S_a(tick) or ?tick; () let S_b(tick:chan) = ?tick; S_b1(tick) and S_b1(tick:chan) = do !b; S_b1(tick) or ?tick; S_b2(tick) and S_b2(tick:chan) = do !b; S_b2(tick) or ?tick; () run 10 of (new tick:chan run (clock(8.0,tick) | S_a(tick)))
run 10 of (new tick:chan run (clock(4.0,tick) | S_b(tick)))
```

```c
Xor_hi_a(a:chan, b:chan, c:chan) =
do !c; Xor_hi_a(a,b,c) or ?b; Xor_lo_ab(a,b,c) or delay@1.0; Xor_lo_a(a,b,c)
and Xor_hi_b(a:chan, b:chan, c:chan) =
do !c; Xor_hi_b(a,b,c) or ?a; Xor_lo_ab(a,b,c) or delay@1.0; Xor_lo_b(a,b,c)
and Xor_lo_a(a:chan, b:chan, c:chan) =
do ?a; Xor_hi_a(a,b,c) or ?b; Xor_lo_ab(a,b,c)
and Xor_lo_b(a:chan, b:chan, c:chan) =
do ?b; Xor_hi_b(a,b,c) or ?a; Xor_lo_ab(a,b,c)
and Xor_lo_ab(a:chan, b:chan, c:chan) =
do delay@1.0; Xor_lo_a(a,b,c) or delay@1.0; Xor_lo_b(a,b,c)
run 50 of (Xor_lo_a(a,b,c) | Xor_lo_b(a,b,c))
```

```
let c = a xor b
```
Xor as an Op Amp

\[
c = A^*(a - b) \\
d = A^*(b - a)
\]

**(Follower (a standard OpAmp trick))**

- \(a=0\) \(b=0\) \(\Rightarrow\) \(d=b-a=0\)
- \(a=0\) \(b=1\) \(\Rightarrow\) \(d=b-a=1\)
- \(a=1\) \(b=0\) \(\Rightarrow\) \(d=b-a=0\)
- \(a=1\) \(b=1\) \(\Rightarrow\) \(d=b-a=0\)

**hence \(d=1\) at next step**

**hence \(d=b\)**

**“Noninverting Configuration”**

\[
d=b \text{ analog response!!}
\]
Changing the OpAmp Gain

An OpAmp provides “infinite” differential amplification, but a stable finite amplification can be obtained by a feedback loop with a load splitter (the follower is a special case of that, which gives gain 1). The equivalent here is simply changing the rate on the feedback link.

Empirical law:
\[ [d] = \frac{[b]}{\text{rate}(a)} \]
Op Amp Inverting Configuration

"Inverting Configuration"

c level depends on a and rate(a)
i.a. a signal is amplified according to rate(a)

c = not b
a zero (ideally, if rate(a) fast enough)
rate(a) has no effect on c
An Xor but Not an Op Amp

$c = A^*(a - b)$
$d = A^*(b - a)$

Not a Follower

$d \neq b$!
Exercise (Open)

- Find the ODEs of some Xor or OpAmp configuration (e.g. Follower), and possibly derive some laws from them.
Summary

- **Influence Diagrams**
  - Don’t trust them

- **Polin Diagrams**
  - An alternate influence-like notation for interacting automata

- **Monopolin Circuits**
  - Amplifiers
  - Inverters
  - Boolean Gates
  - OpAmp