Cells are plausible targets for high-level spatial languages

Jacob Beal, Jonathan Bachrach

Spatial Computing Workshop
@ IEEE SASO 2008

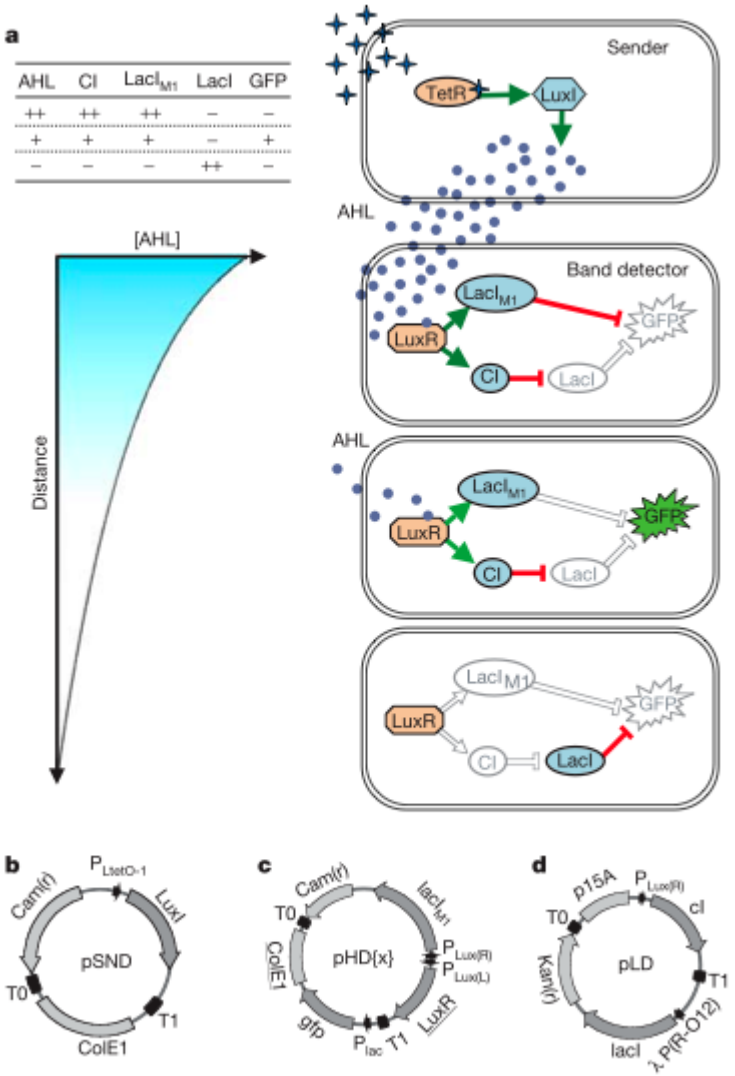
BBN TECHNOLOGIES

Compiling “band-detect”

Proto

```
(def band-detector (signal lo hi)
  (and (> signal lo)
       (< signal hi)))

(let
  ((v (diffuse (aTc) 0.8 0.05)))
 (green (band-detect v 0.2 1)))
```

Weiss bacteria



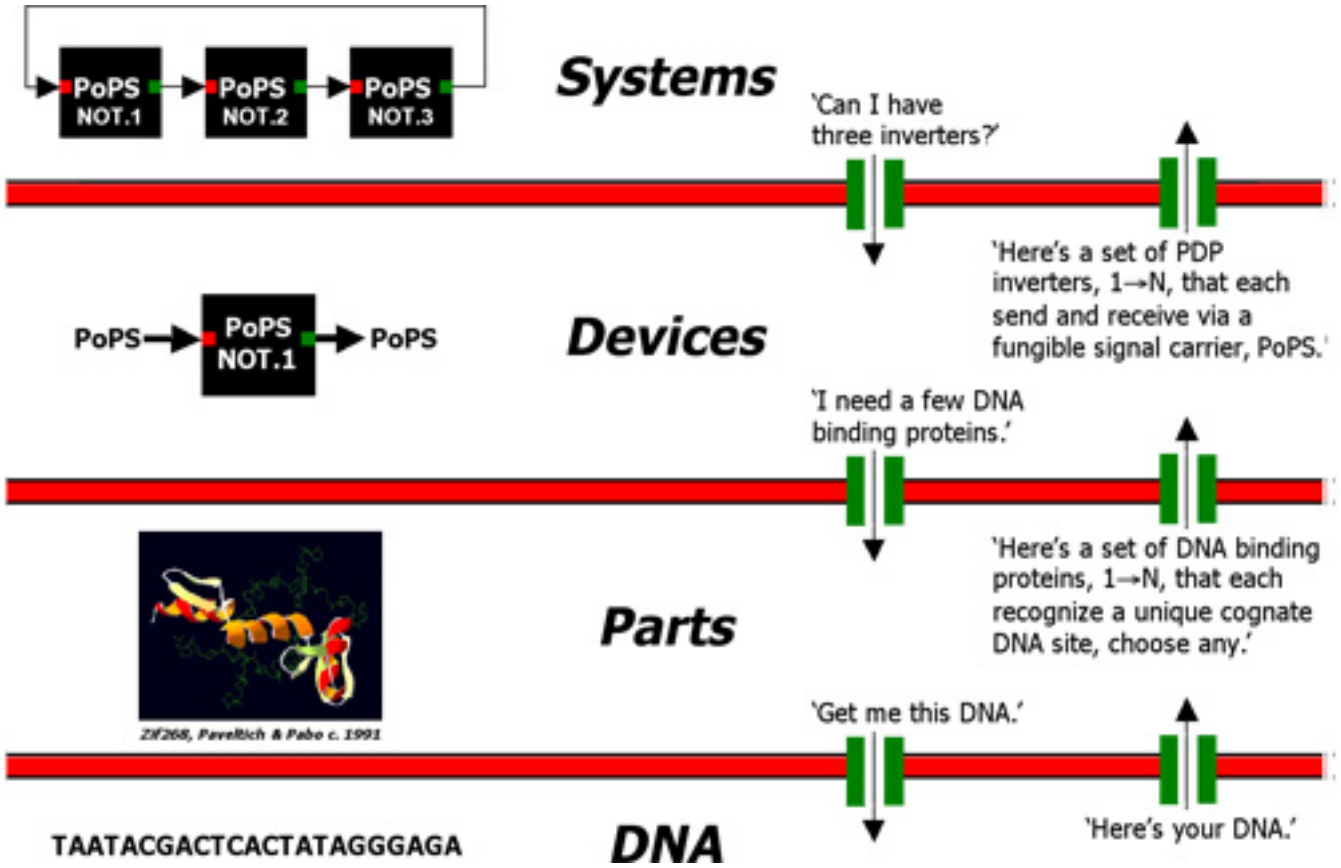
Outline

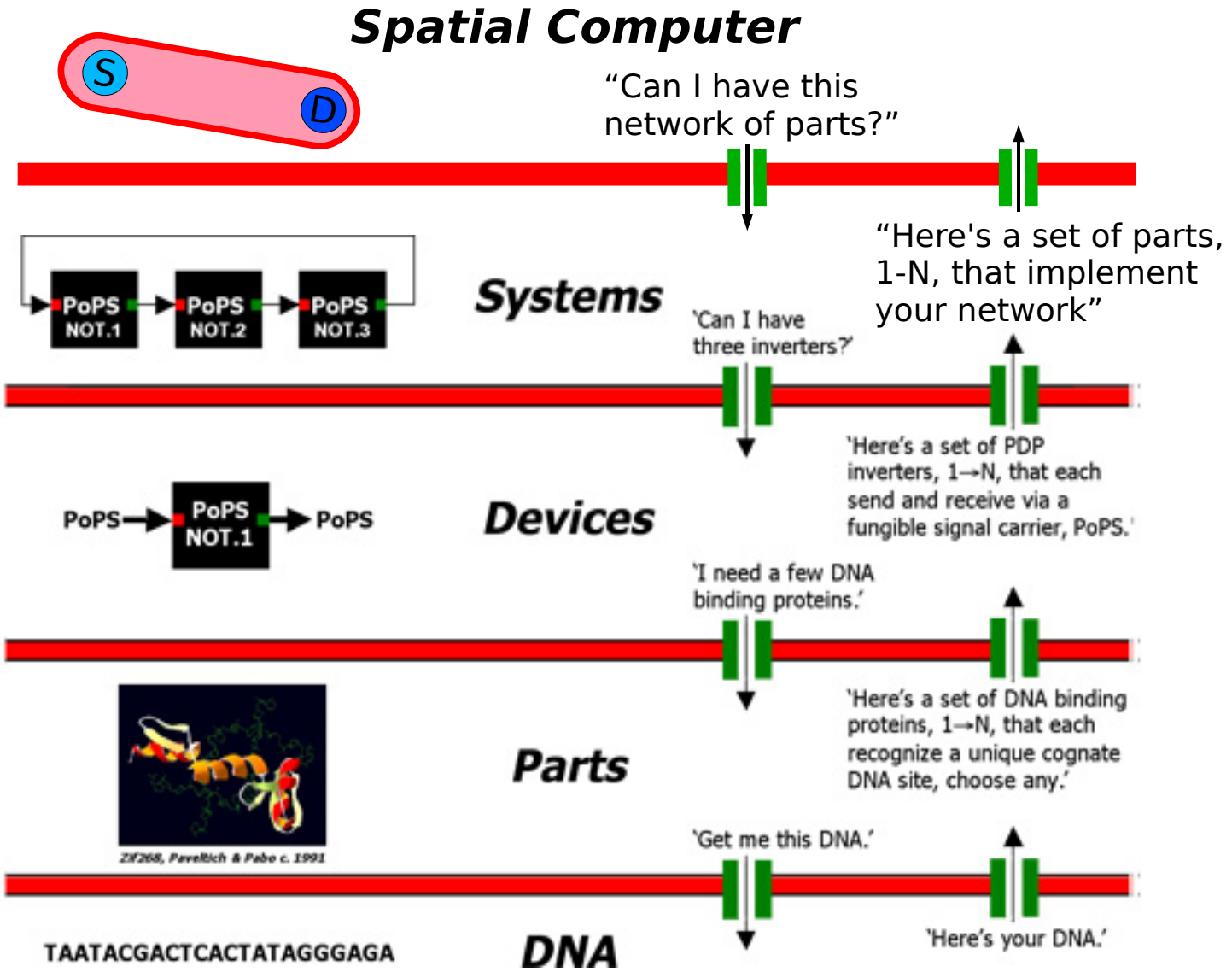
- Background:
  - Programming bacteria
  - Proto
- Compilation
- Optimization

HLLs & Bacteria

- High-level languages:
  - Shorter programs mean less efficient code
  - Optimizing compilers can help
- Bacteria
  - Extremely tight resource constraints
  - Inherently parallel chemical execution

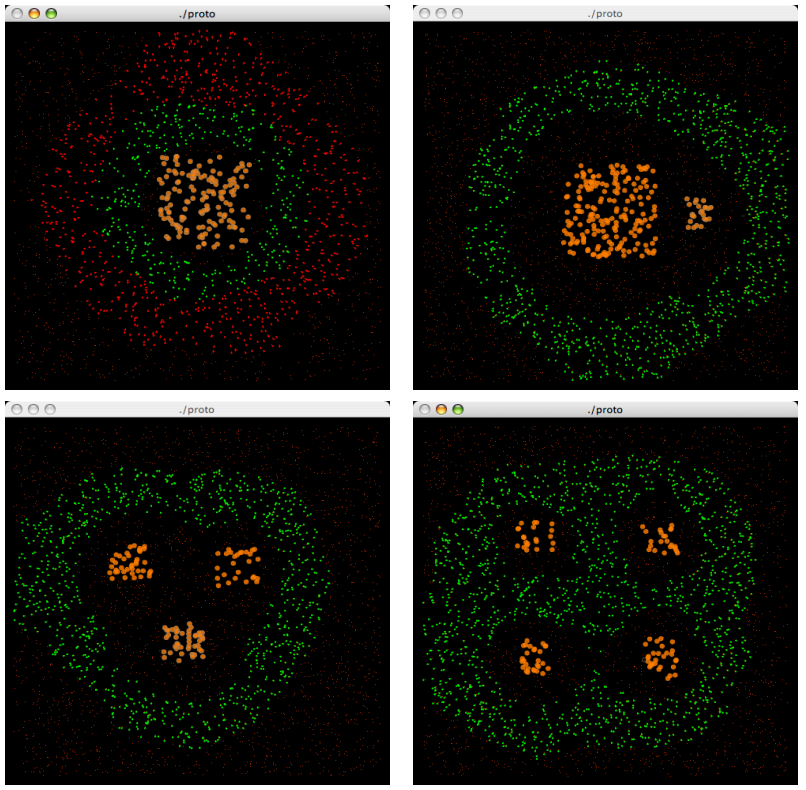
Synthetic Biology Vision



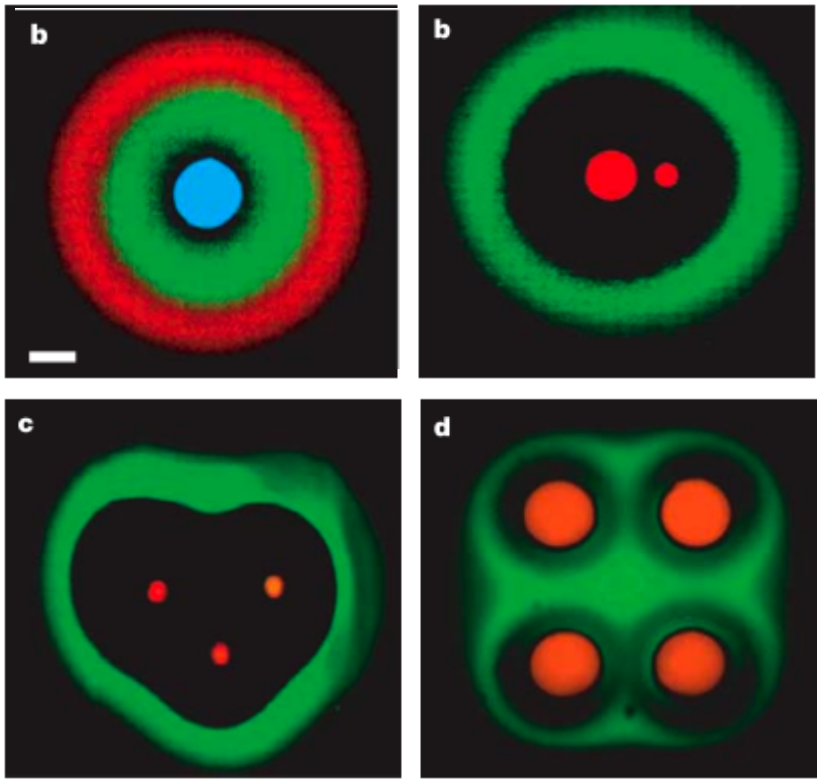

Spatial Computer
S
D
"Can I have this network of parts?"
"Here's a set of parts, 1-N, that implement your network"
PoPS NOT.1
PoPS NOT.2
PoPS NOT.3
Systems
'Can I have three inverters?'
'Here's a set of PDP inverters, 1→N, that each send and receive via a fungible signal carrier, PoPS.'
PoPS
PoPS NOT.1
PoPS
Devices
'I need a few DNA binding proteins.'
'Here's a set of DNA binding proteins, 1→N, that each recognize a unique cognate DNA site, choose any.'
ZIF268, Pavletich & Pabo c. 1991
Parts
'Get me this DNA.'
'Here's your DNA.'
TAATACGACTCACTATAGGGAGA
DNA

Band detect: behavior

Proto



Weiss bacteria



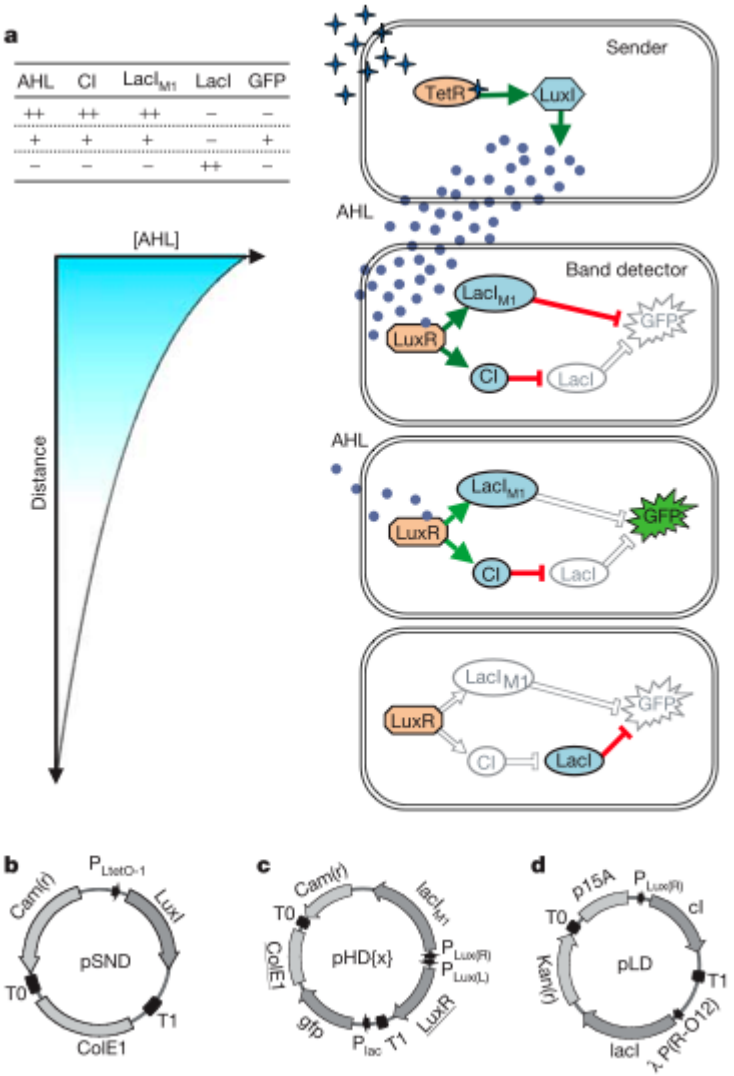
Band detect: code

Proto

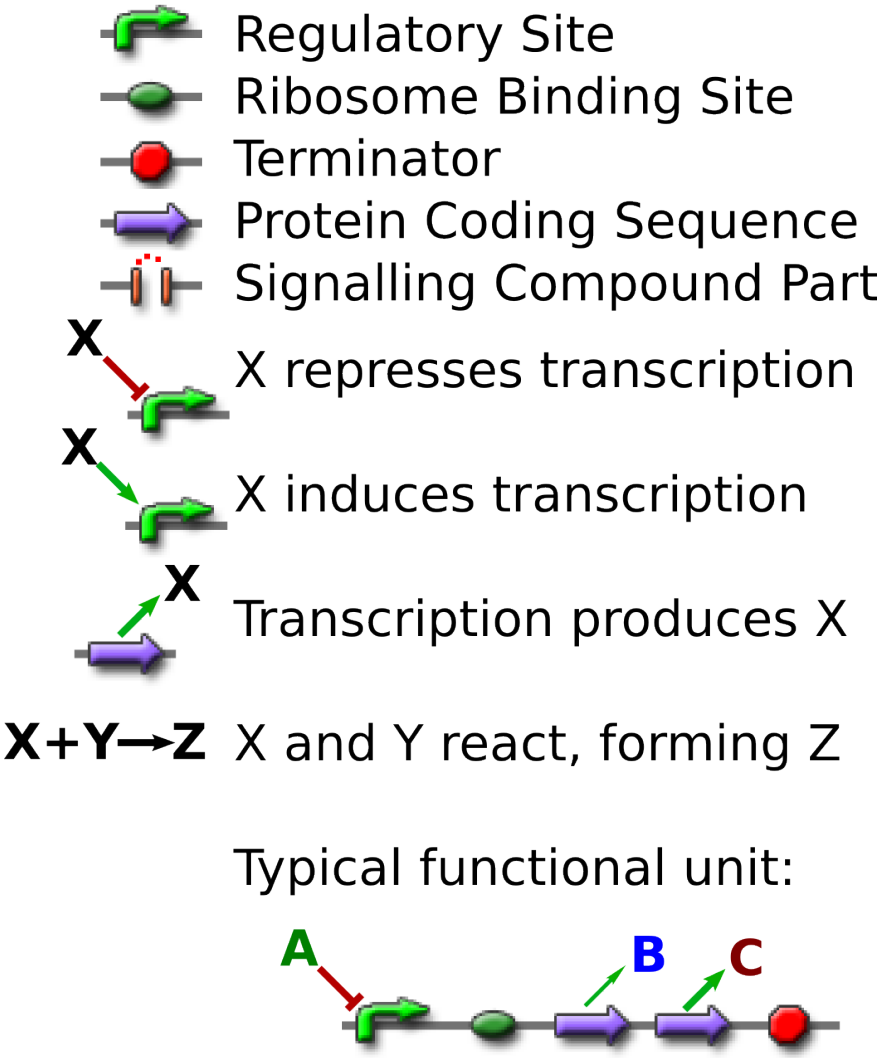
```
(def band-detector (signal lo hi)
  (and (> signal lo)
       (< signal hi)))

(let
  ((v (diffuse (aTc) 0.8 0.05)))
 (green (band-detect v 0.2 1)))
```

simpler, more reusable

Weiss bacteria

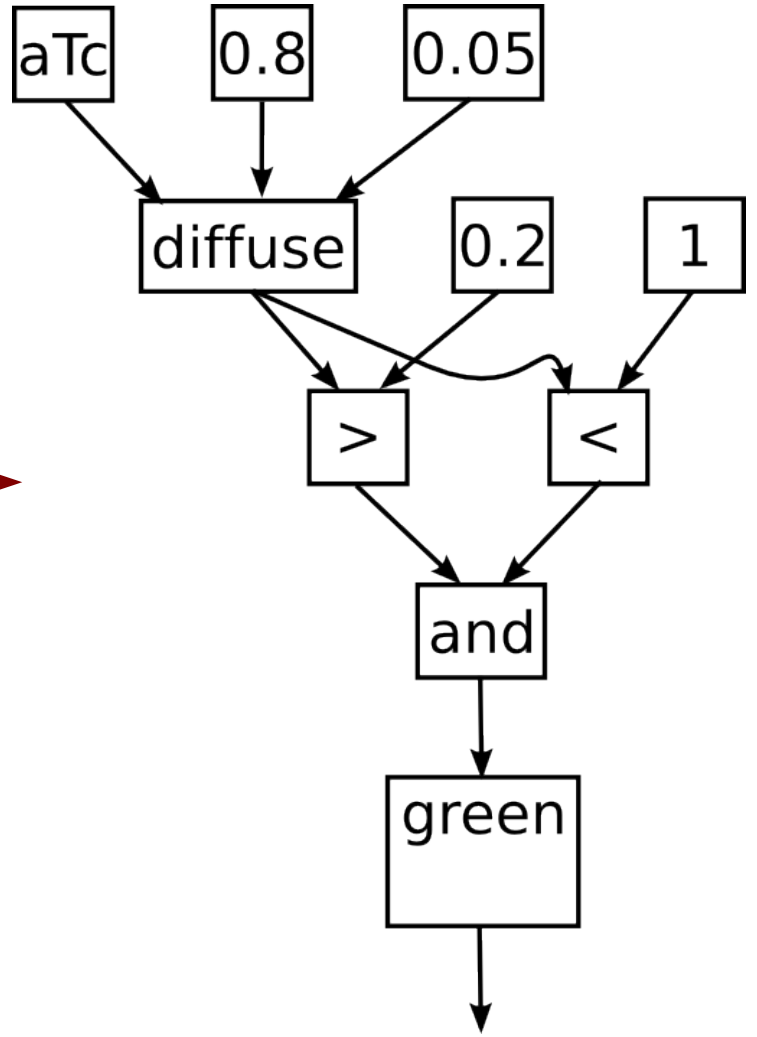


BioBrick Primitives



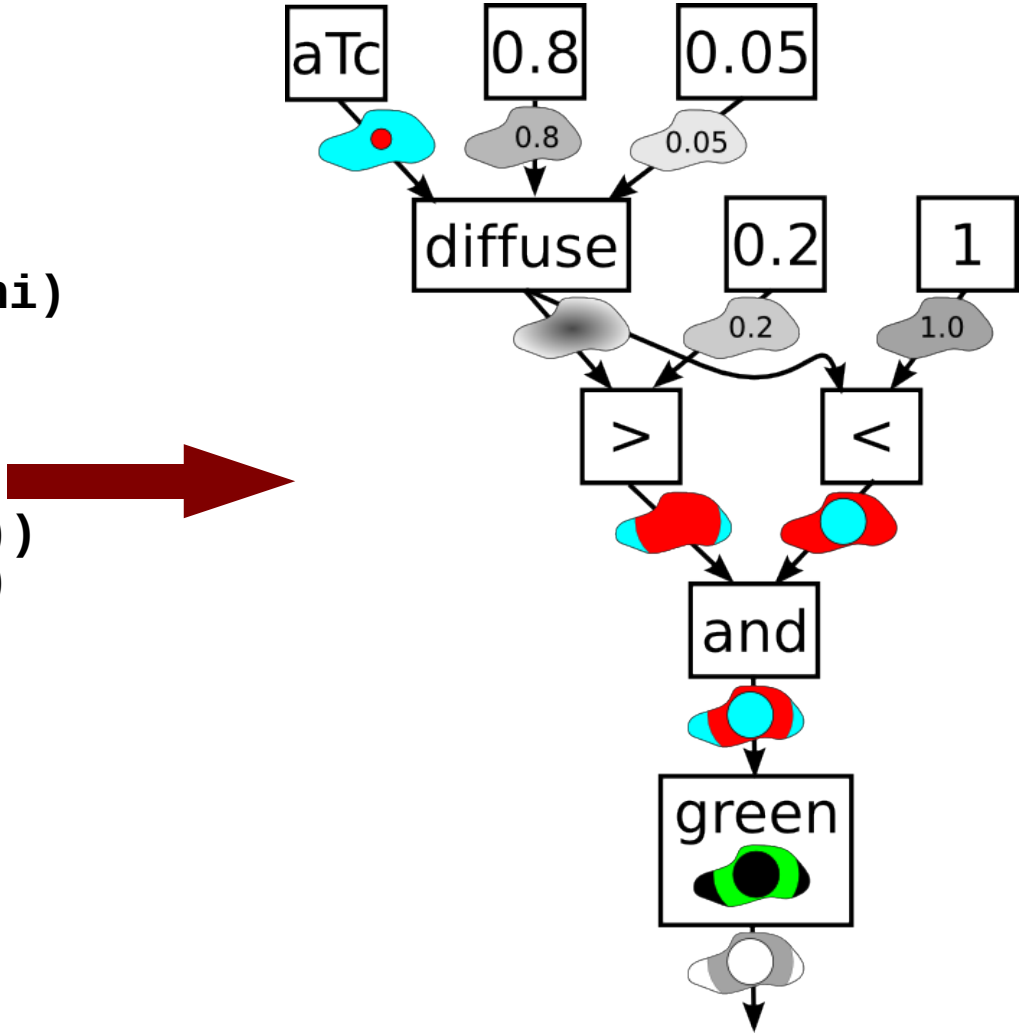
Proto: a light-weight spatial computing language

```
(def band-detector (signal lo hi)
  (and (> signal lo)
       (< signal hi)))

(let
  ((v (diffuse (aTc) 0.8 0.05)))
 (green (band-detect v 0.2 1)))
```



Proto: a light-weight spatial computing language

```
(def band-detector (signal lo hi)
  (and (> signal lo)
       (< signal hi)))

(let
  ((v (diffuse (aTc) 0.8 0.05)))
 (green (band-detect v 0.2 1)))
```

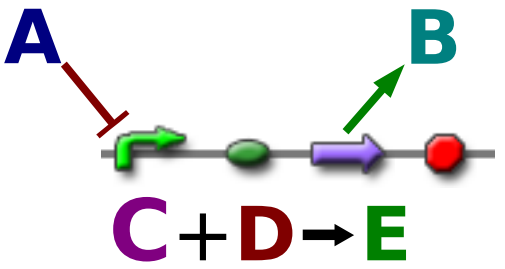


Proto to GRNs: First Steps

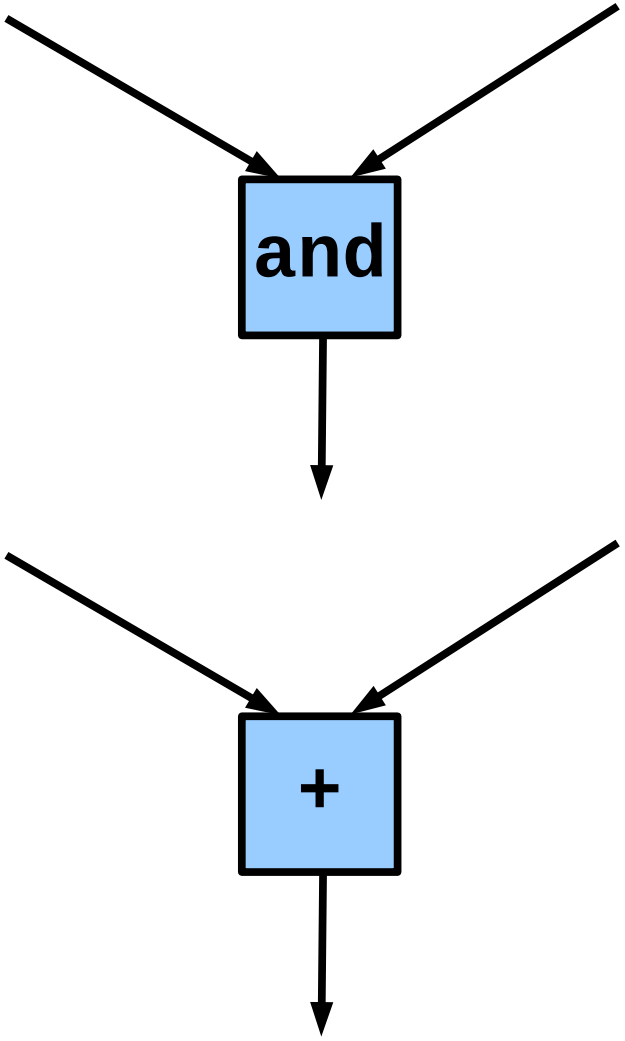
- Logical: **`AND`**, **`OR`**, **`NOT`** ✓
- Flow control: **`IF`**, **`MUX`** ✓
- Arithmetic: **`+`**, **`-`**, **`*`**, **`/`**, **`log`**, **`exp`**, ...
- Relational: **`>`**, **`<`**, **`=`**

Two possible implementations:

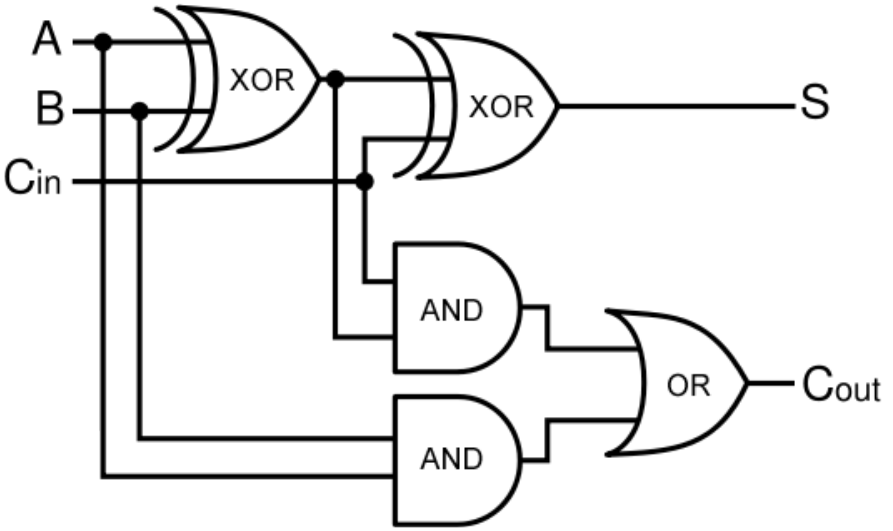
- Regulation
- Reaction



Digital Arithmetic is Expensive



- 1 operation



- 5 operations/bit

Use digital for booleans, analog for numbers

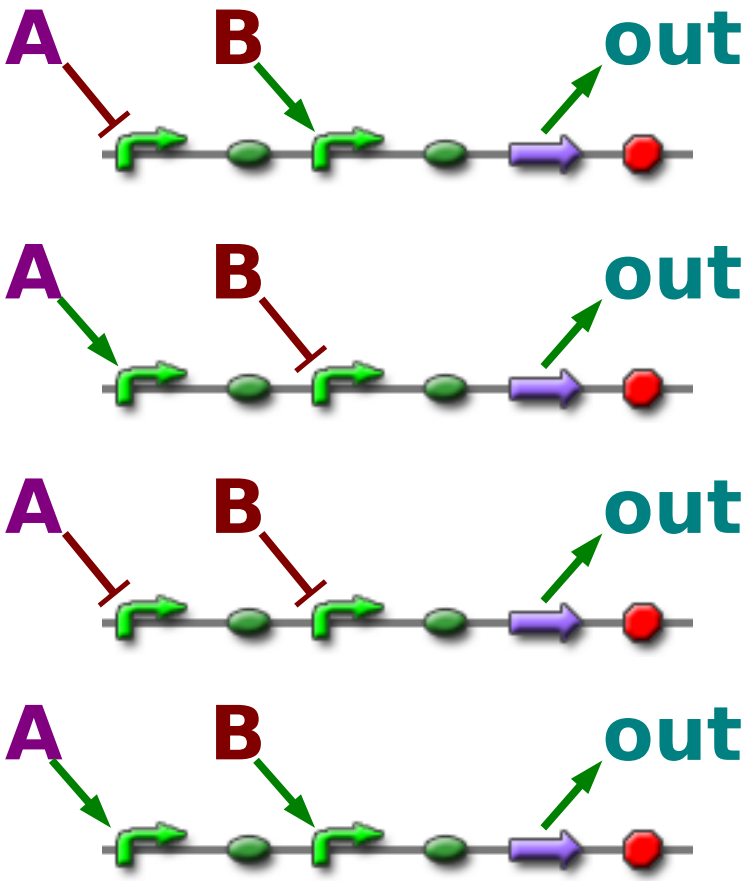
Arithmetic

- *c*: constitutive expression
- **(+ A B)**: same chemical represents both
- **(- A B)**: **A**+**B**→**C**
- **(log A)**, **(exp A)**: lookup tables
  - approximate w. summary of > tests?
- **(* A B)**, **(/ A B)**: log add, subtract

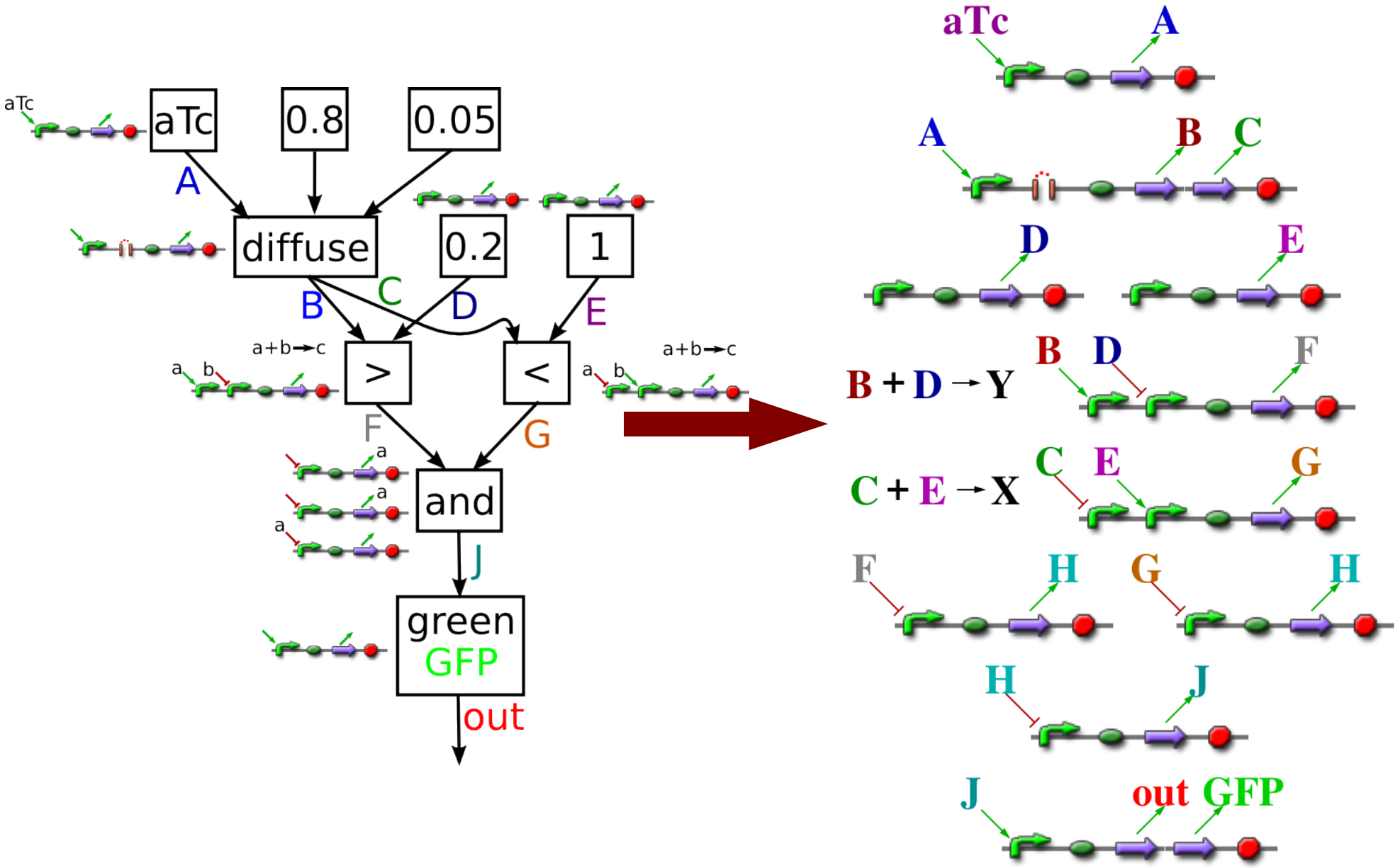
Range? How many bits?

Relational: A→D conversion

- **A**+**B**→**C**
  - `(< A B)`
  - `(> A B)`
  - `(= A B)`
  - `(!= A B)`



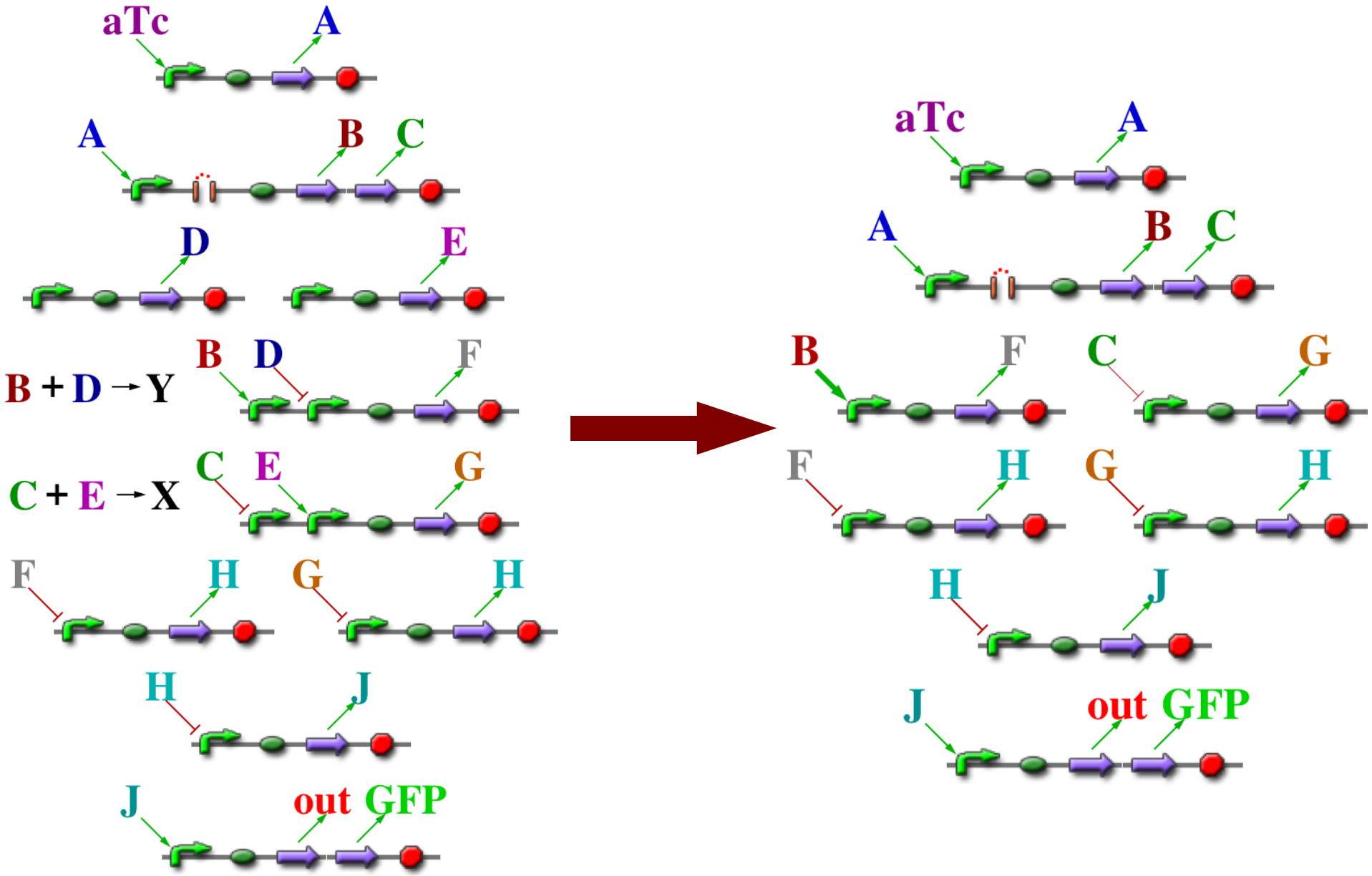
Naïve Implementation



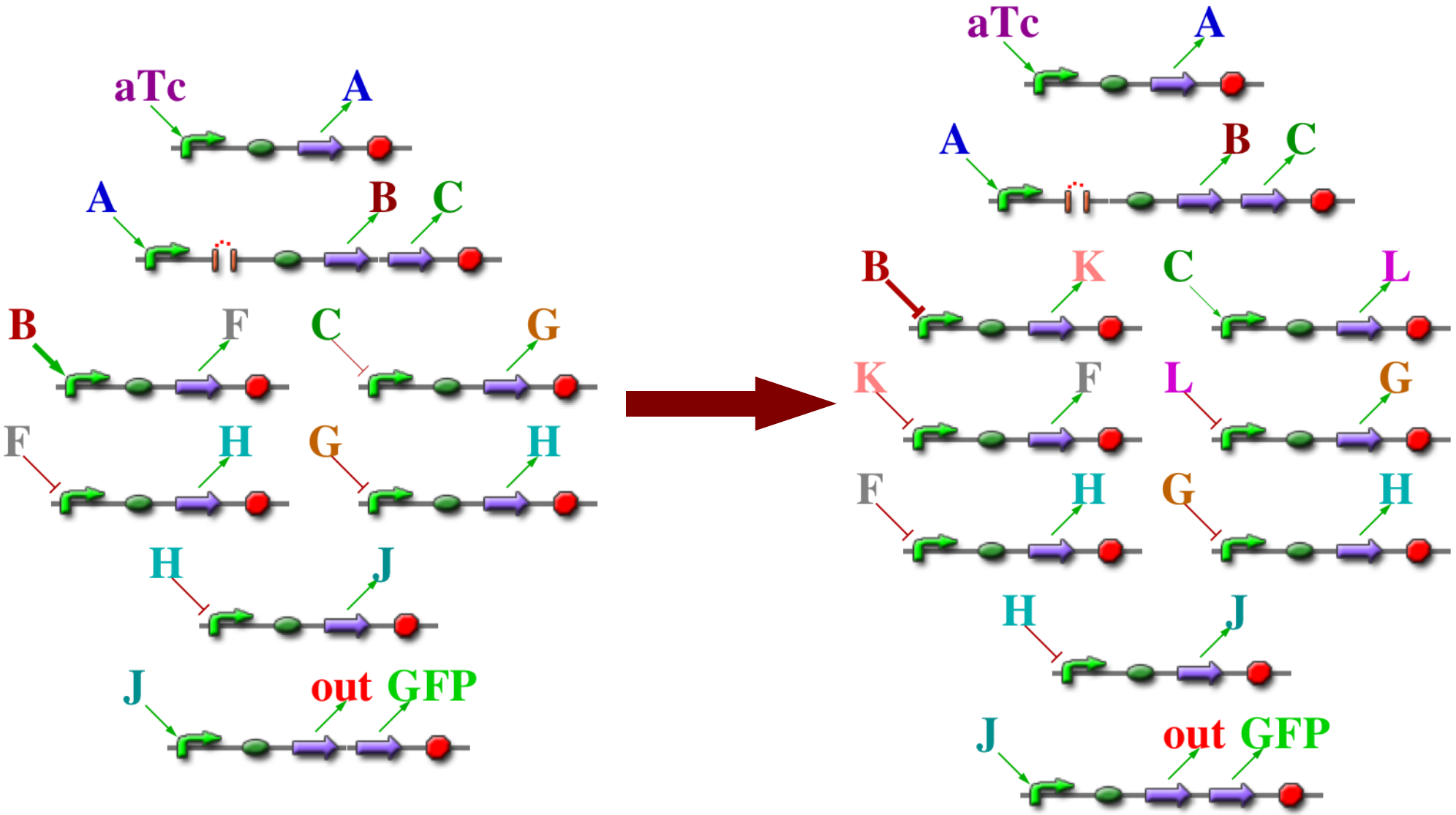
Resources Required

Resource	Hand Tuned	Naive
Signal-carrying chemical	3	11
Protein coding sequence	6	14
Promoters	5	14
Intercellular messengers	2	2
Chemical reactions	0	2

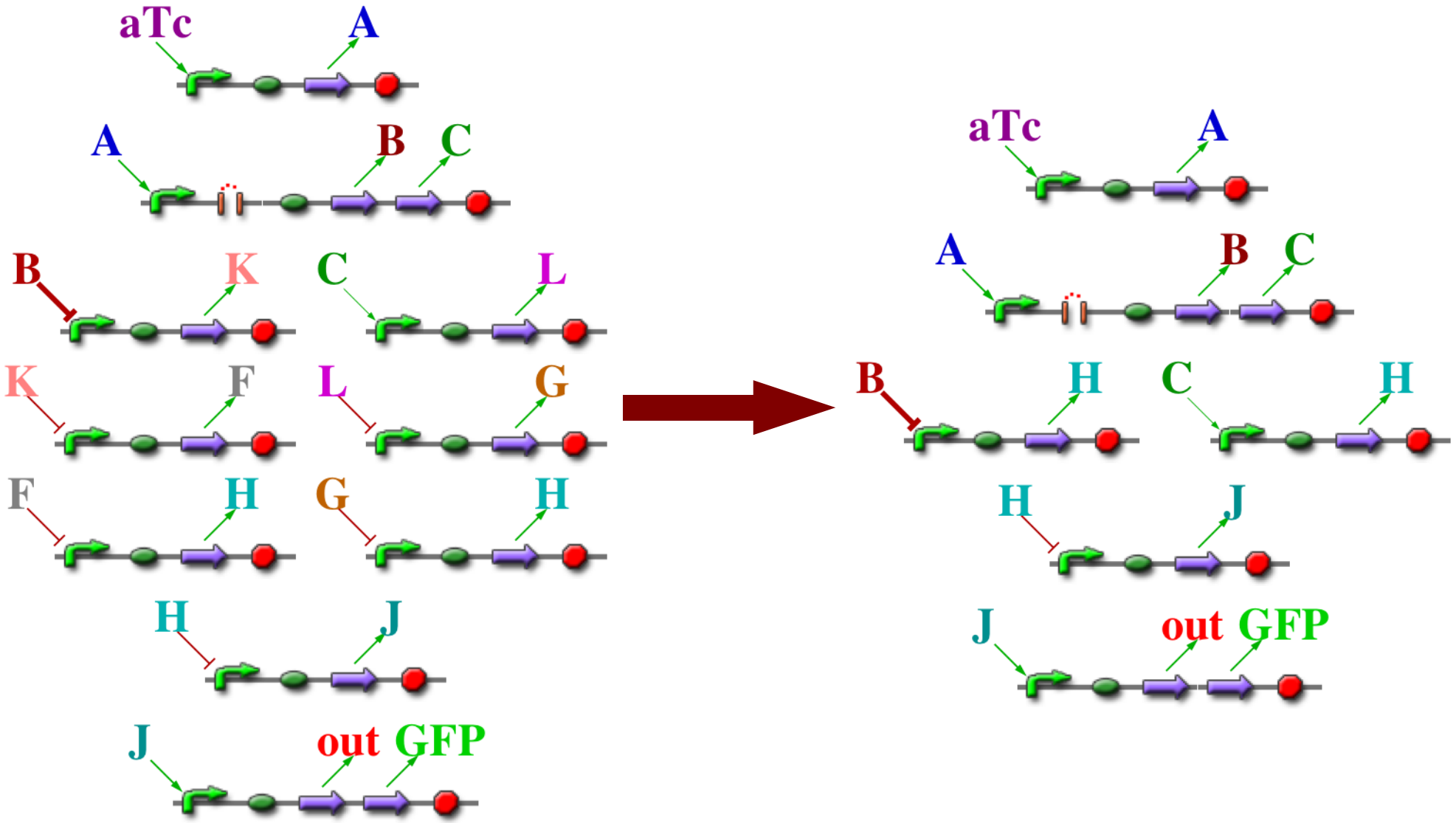
Optimize: Constant Elimination



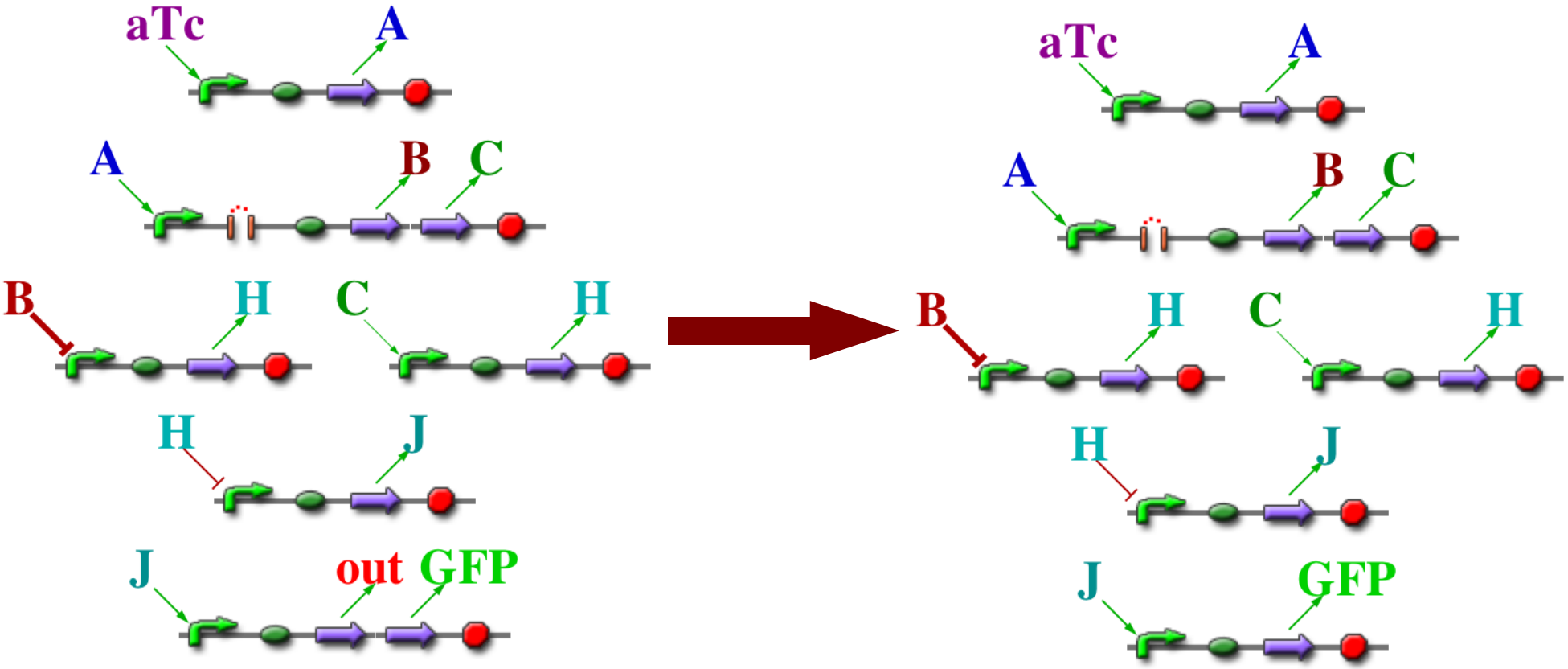
Optimize: Algebraic Simp. (1/2)



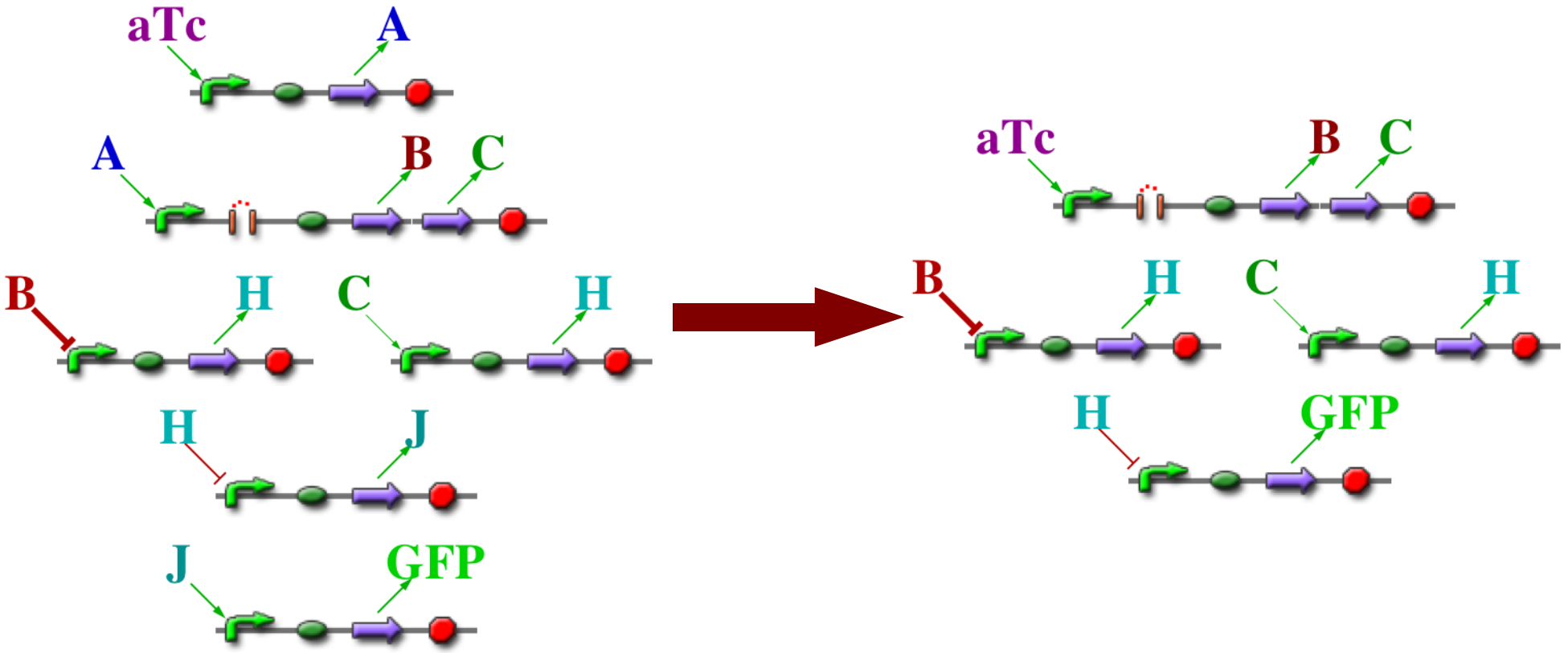
Optimize: Algebraic Simp. (2/2)



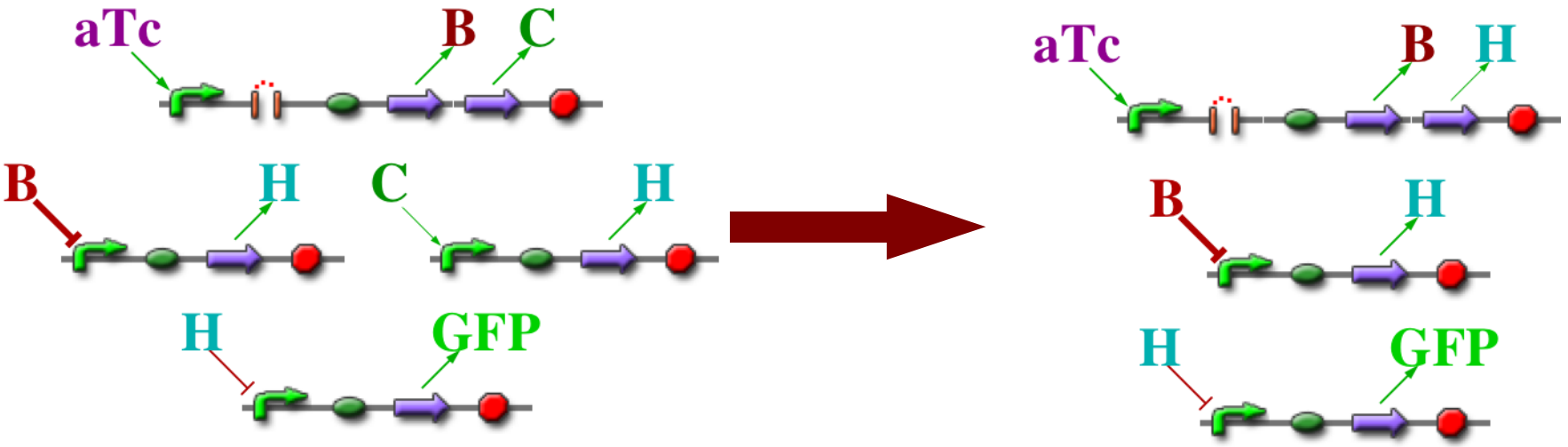
Optimize: Dead Code Elimination



Optimize: Copy Propagation



Optimize: Use-Definition Analysis



Resources Required

Resource	Hand Tuned	Naive	Optimized
Signal-carrying chemical	3	11	3
Protein coding sequence	6	14	6
Promoters	5	14	5
Intercellular messengers	2	2	2
Chemical reactions	0	2	0

Contributions

- Sketch of HLL to BioBrick compilation
  - Example: Weiss band-detect
- Mixed analog/digital computation
- Optimization can be effective!

But will it work...?