Engineering Genetic Circuits

Chris J. Myers

Lecture 12: Genetic Circuit Technology Mapping

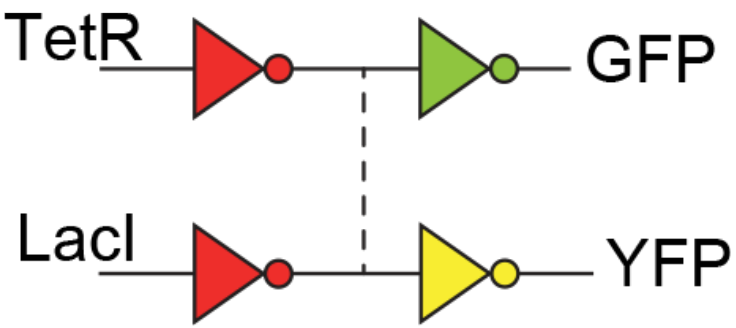
Genetic Circuit Designs

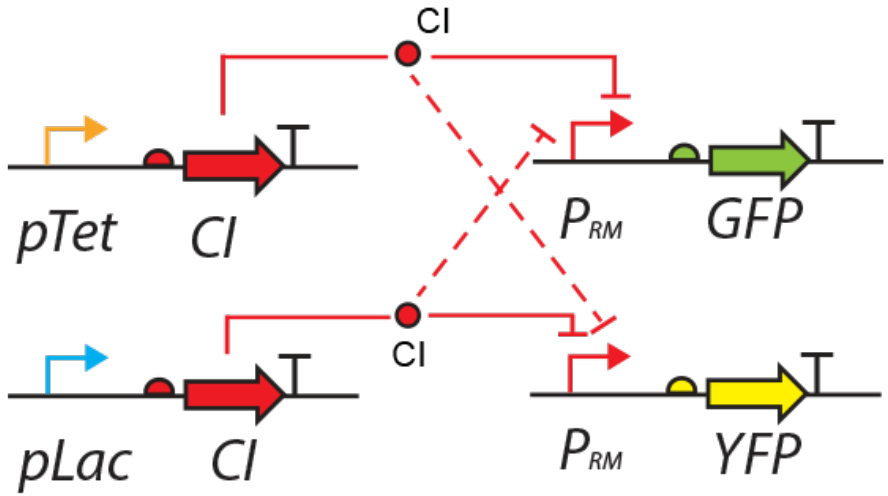
- Genetic circuits are created from biological components that mimic the behavior of Boolean logic gates.
- Genetic circuits can be built inside of a living organism (*in vivo*) or in a test tube (*in vitro*).
- Most genetic circuits that have been built are **combinational circuits**: input signals map to output signal.
- Outside of combinational circuits, memory circuits that have been constructed for **sequential circuits**: input signals are combined with states to produce the output signal.

Genetic Constraints

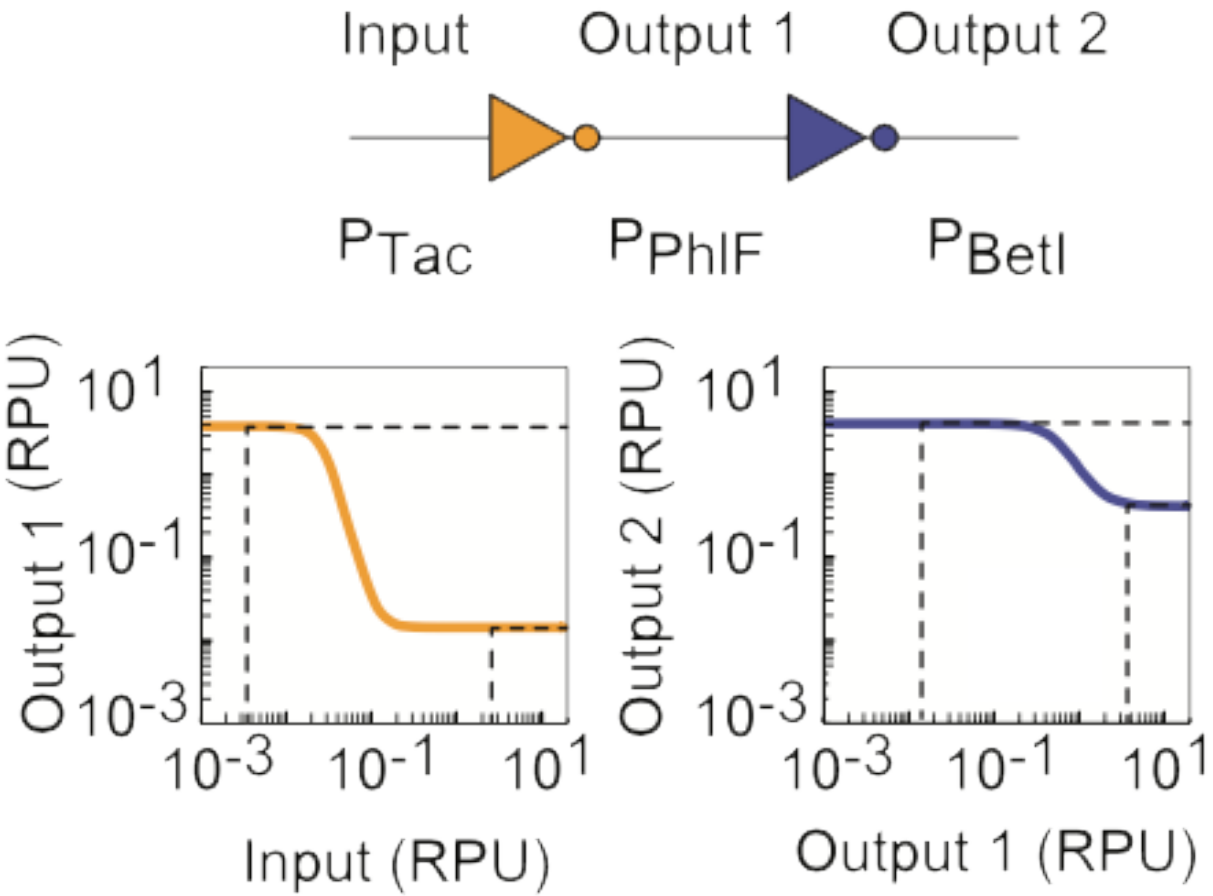
- Crosstalk
- Signal Mismatch
- Roadblocking
- Genetic Context Effects

Crosstalk



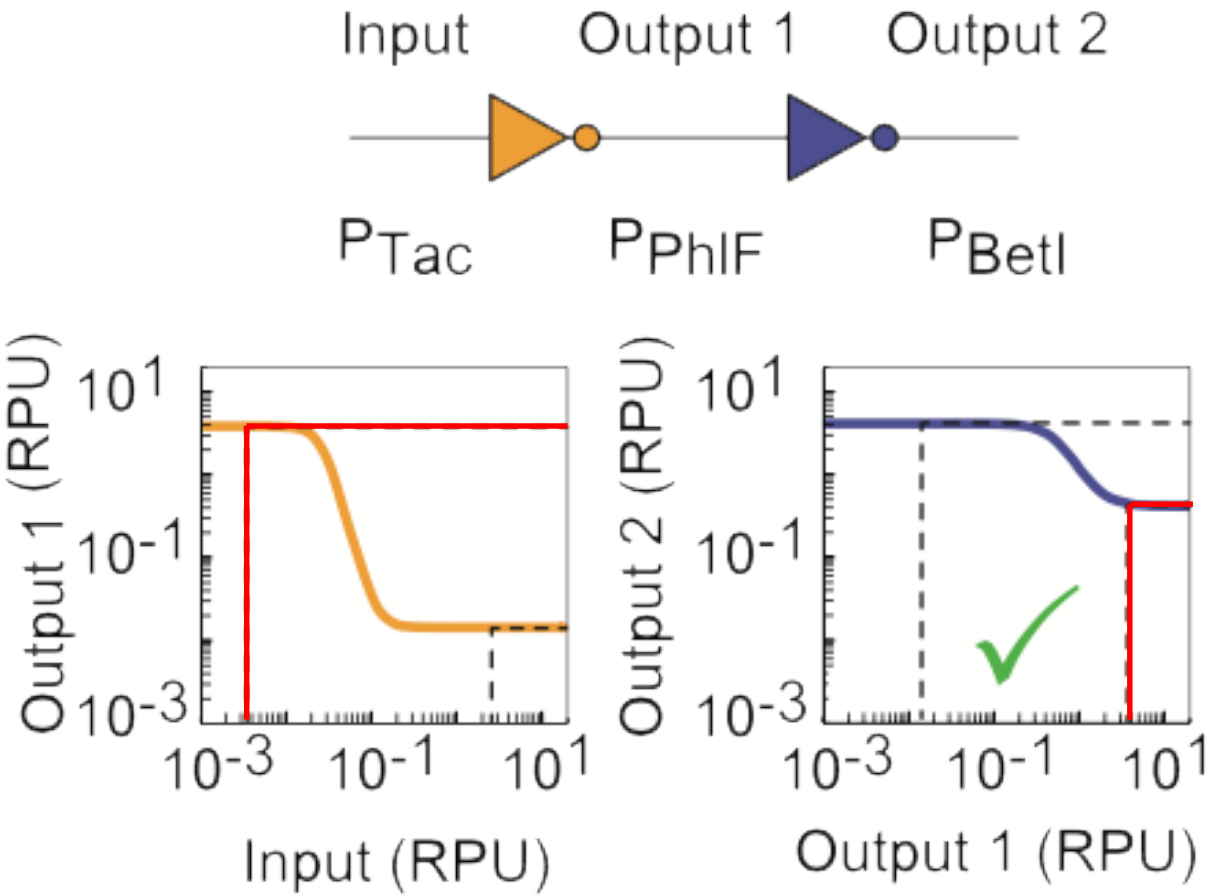


Signal Mismatch



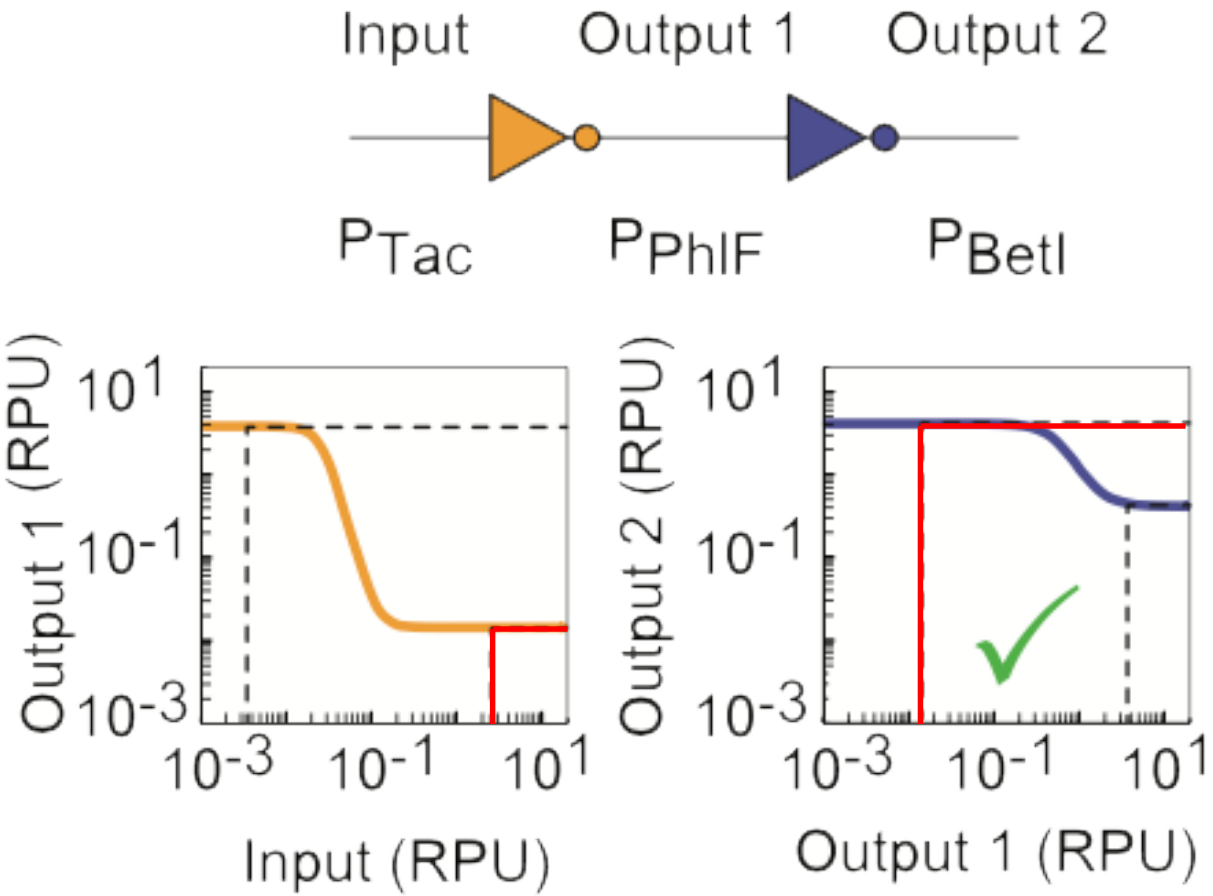
Nielsen et al., *Science*, 2016

Signal Mismatch



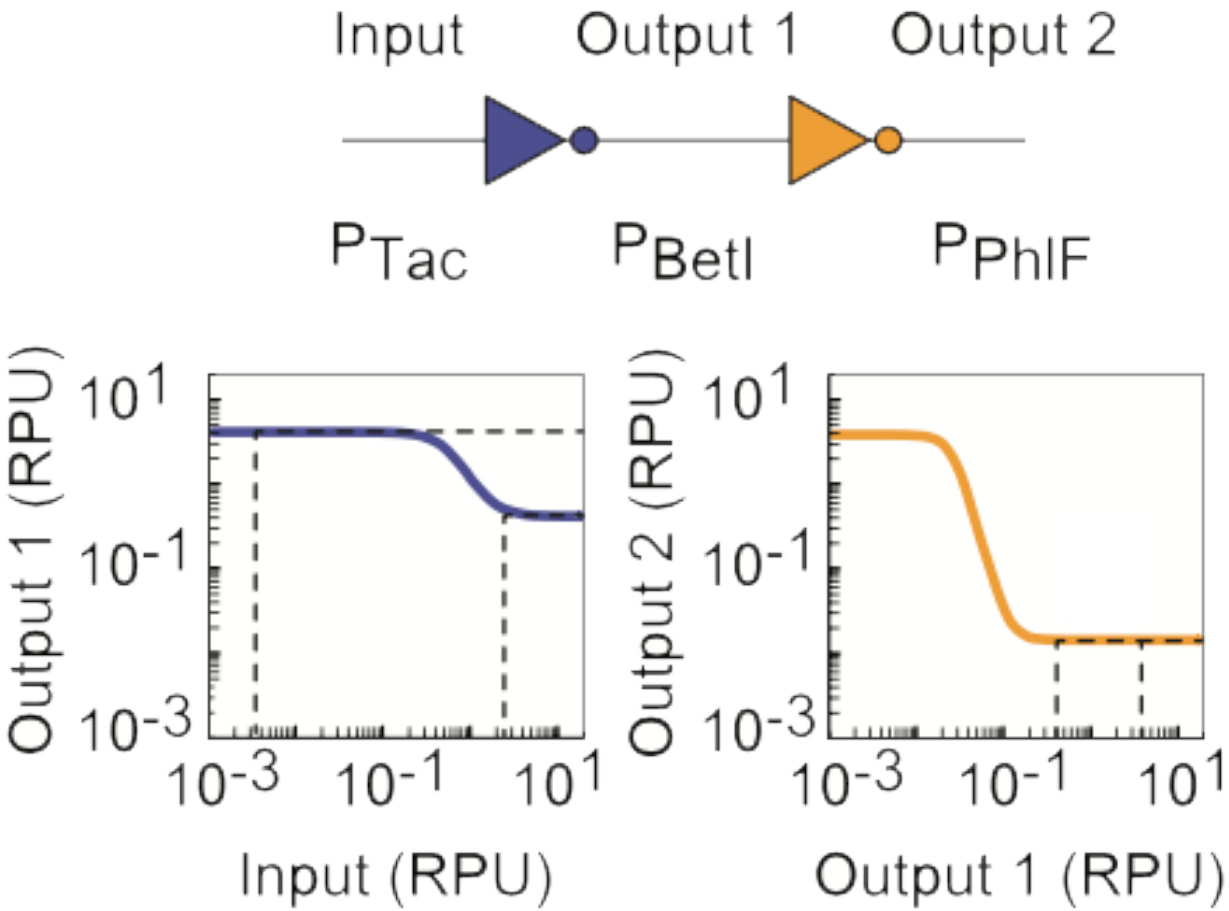
Nielsen et al., *Science*, 2016

Signal Mismatch



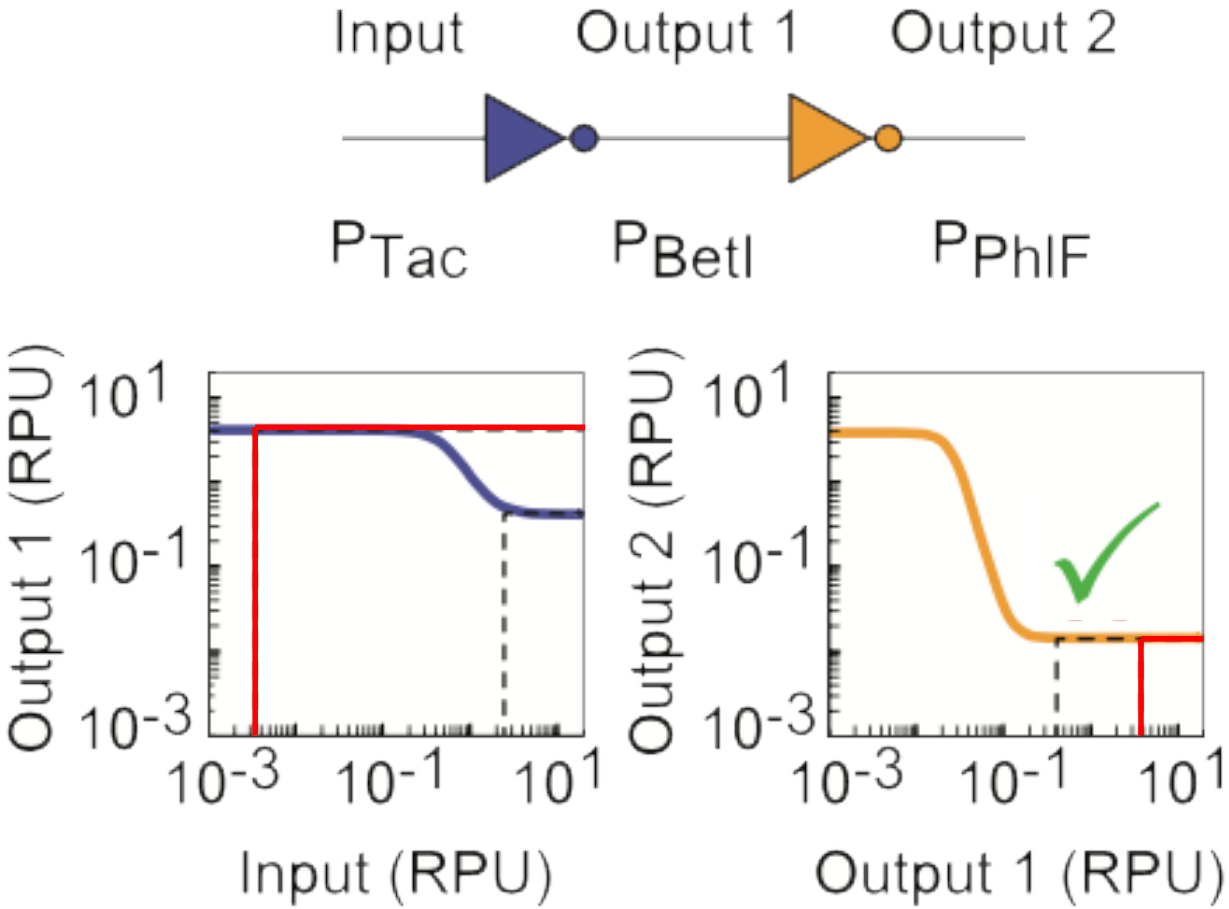
Nielsen et al., *Science*, 2016

Signal Mismatch



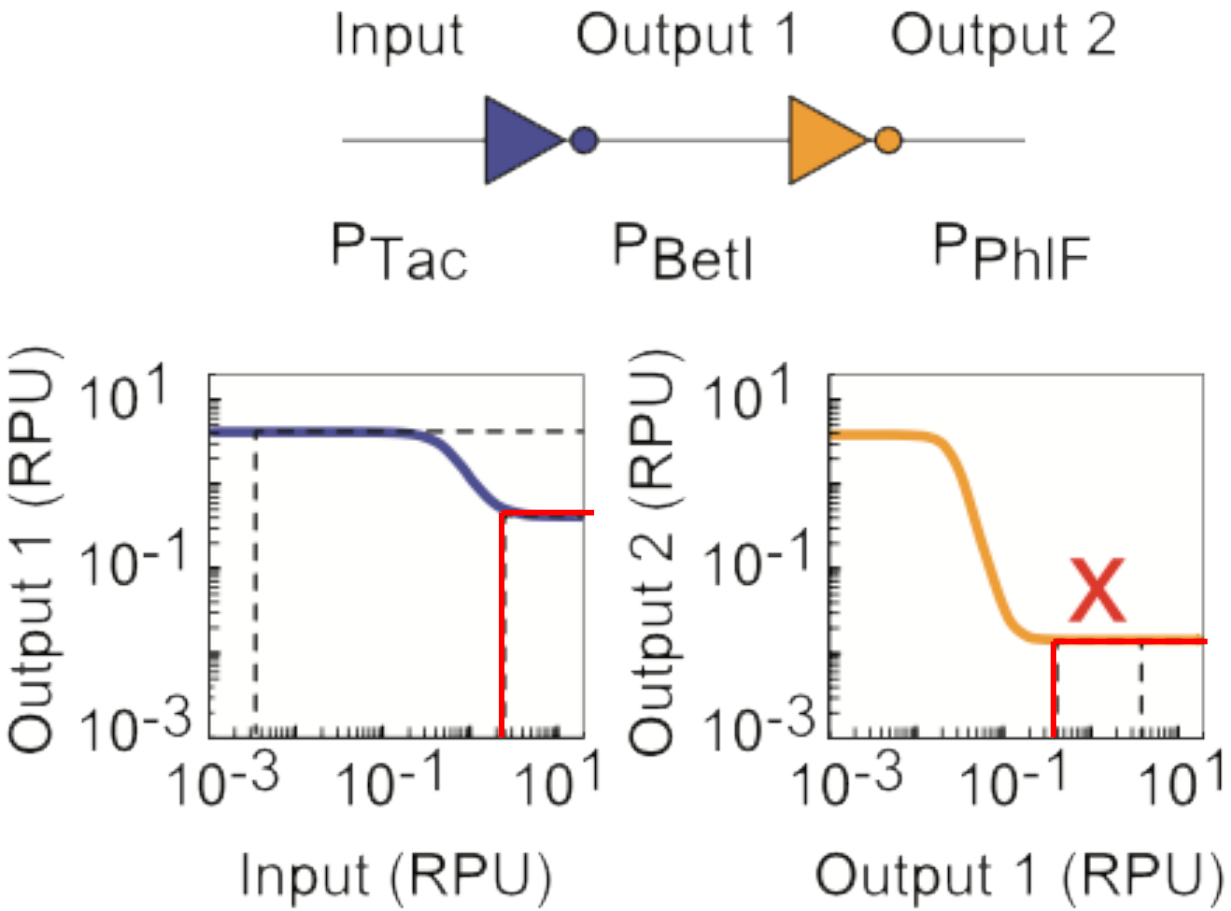
Nielsen et al., *Science*, 2016

Signal Mismatch



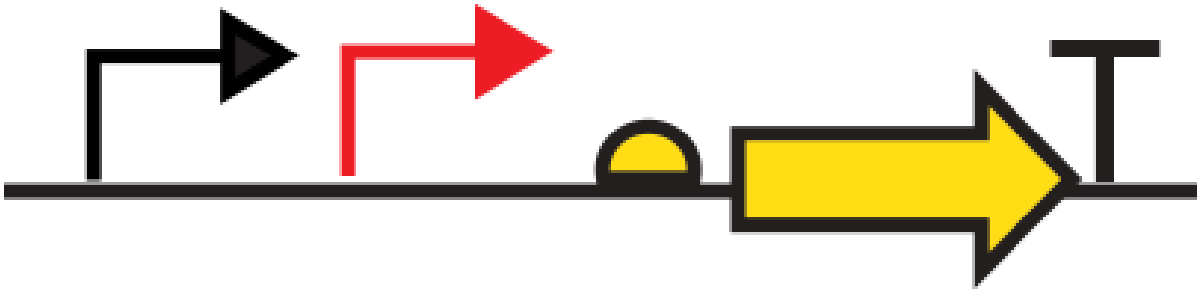
Nielsen et al., *Science*, 2016

Signal Mismatch



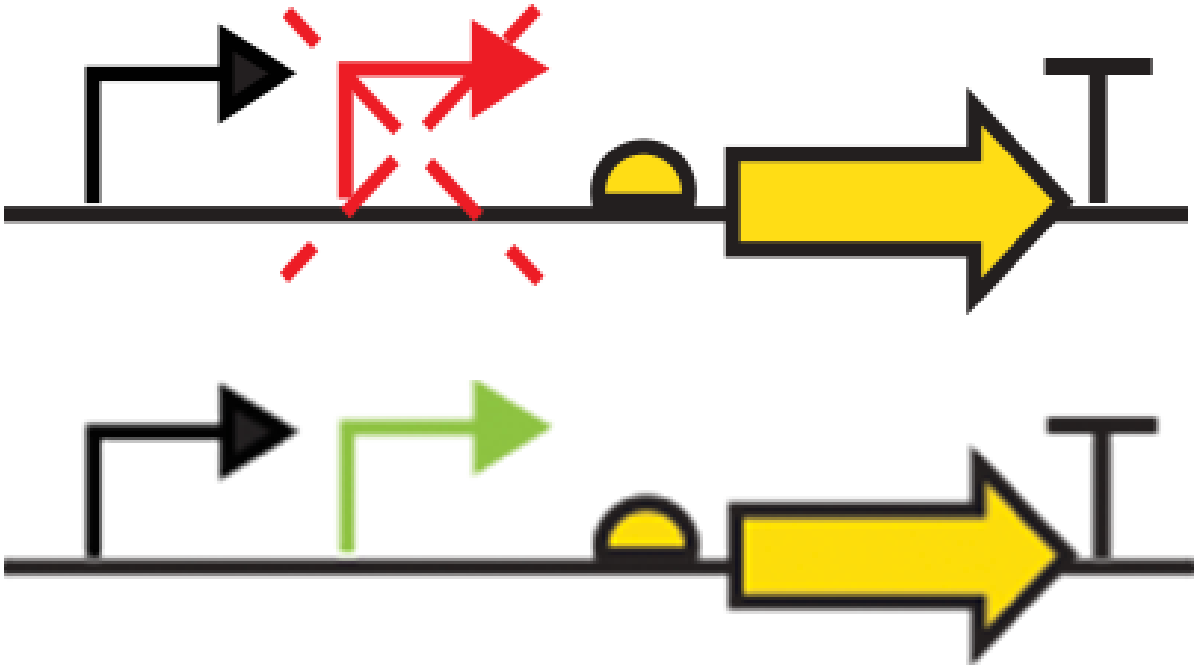
Nielsen et al., *Science*, 2016

Roadblocking



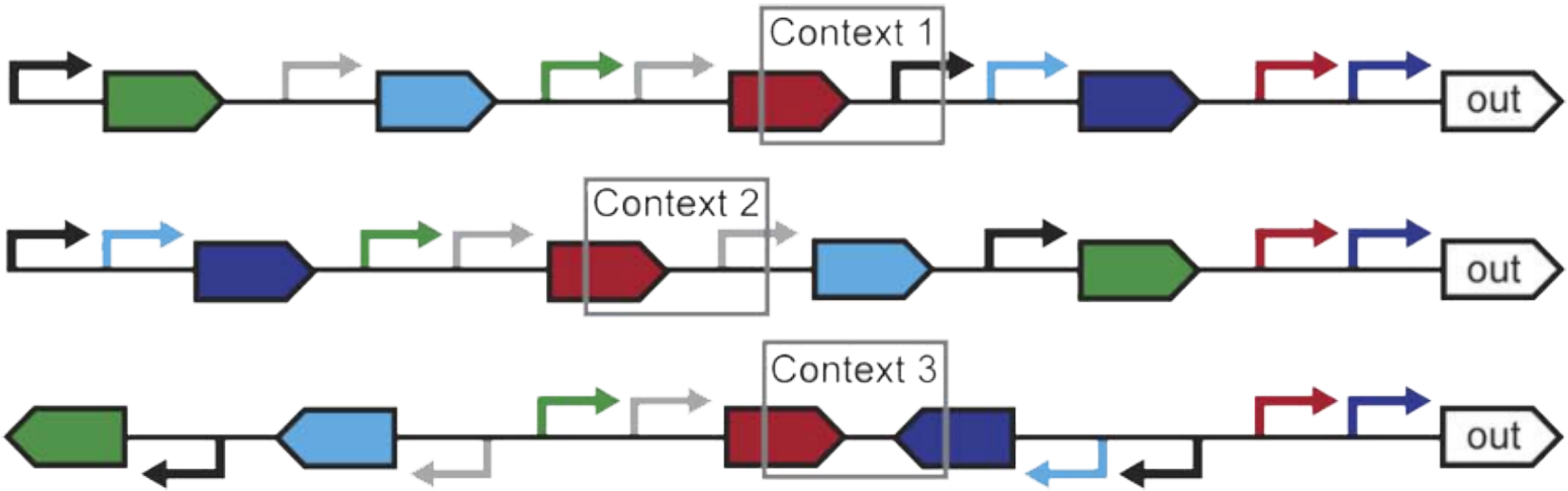
Nielsen et al., *Science*, 2016

Roadblocking



Nielsen et al., *Science*, 2016

Genetic Context Effects



Vaidyanathan et al., *IEEE*, 2015

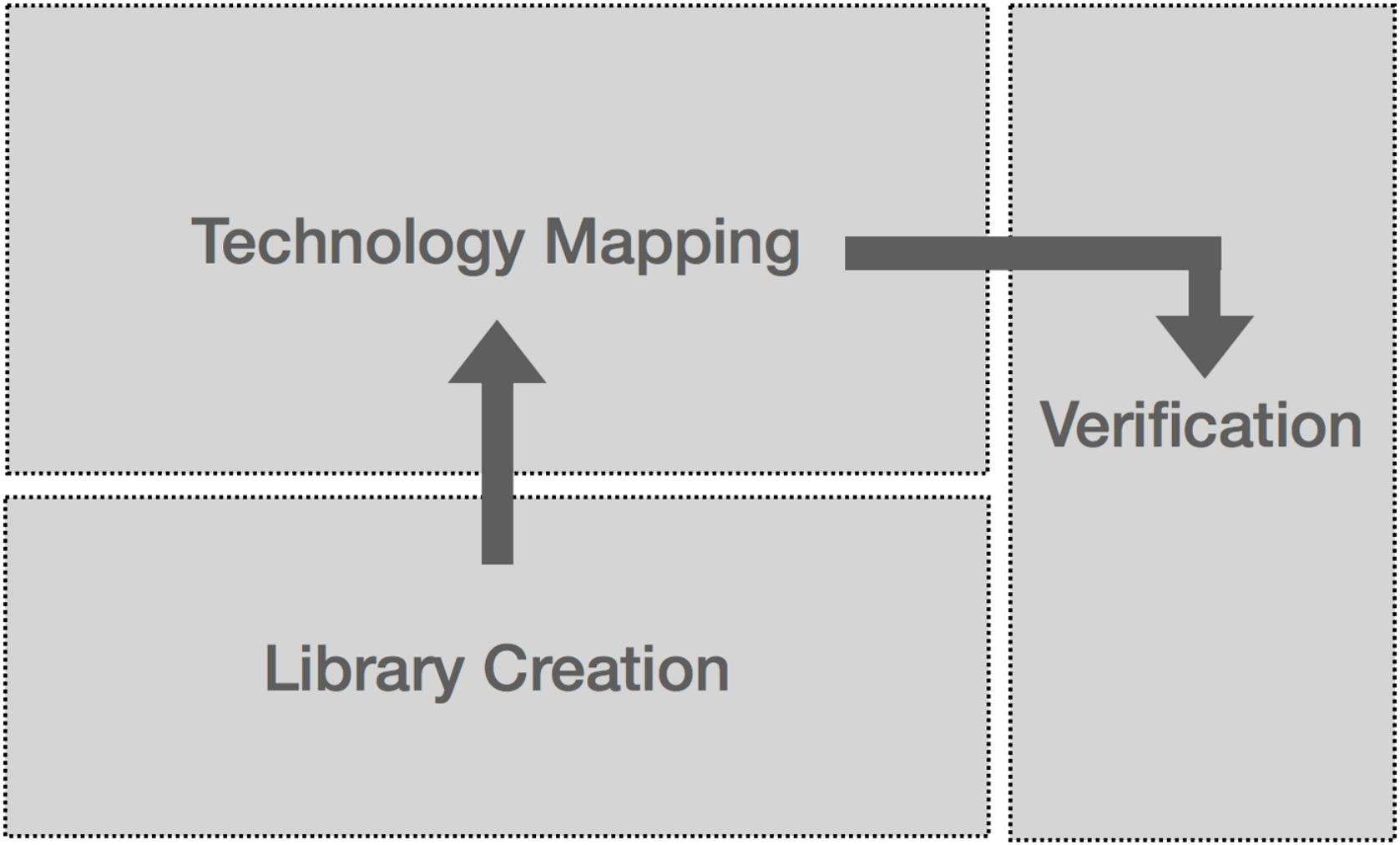
Automation and Computer Aided Design (CAD) tools

- Logic gates are used to build complex circuits.
- Complex circuit is described in some form of specification.
- Building complex circuit that meets the specification is time consuming.
- Design automation will help refine the design space before building a productive circuit to meet the specification (description of the circuit) goal.
- Models can be generated programatically through CAD tools.
- Analysis of models can help evaluate design alternatives on a computer (*in silico*).

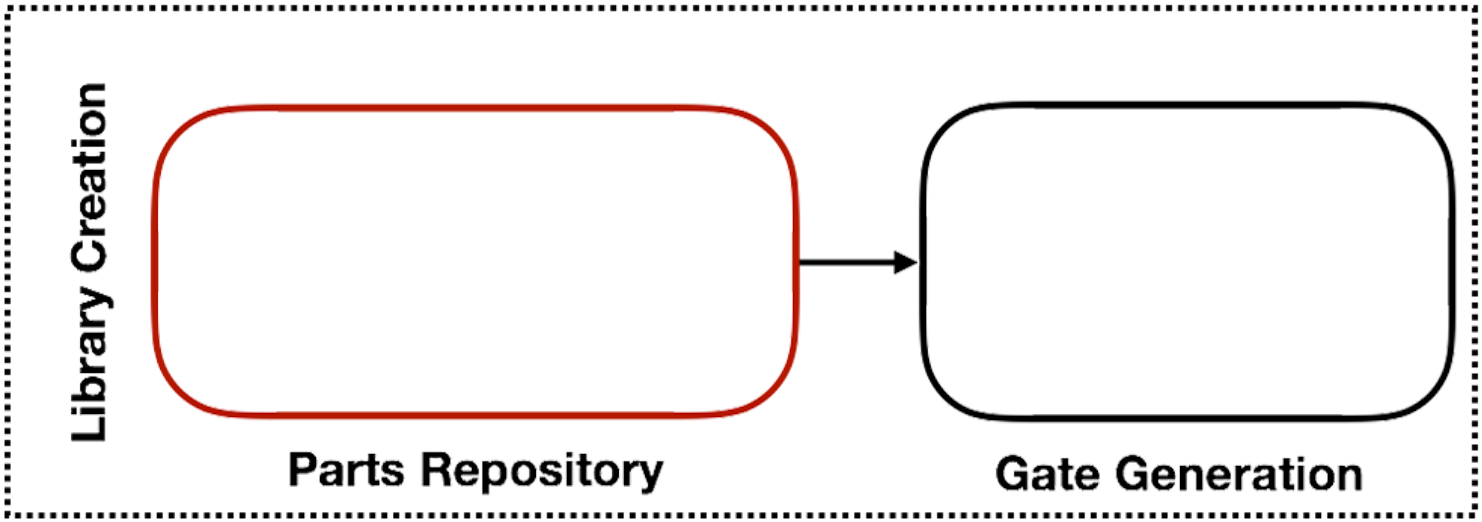
Technology Mapping

- Technology mapping process assigns physical biological parts to implement the functional design specification.
- Existing technology mapping frameworks have accounted for genetic design constraints.

Technology Mapping Template

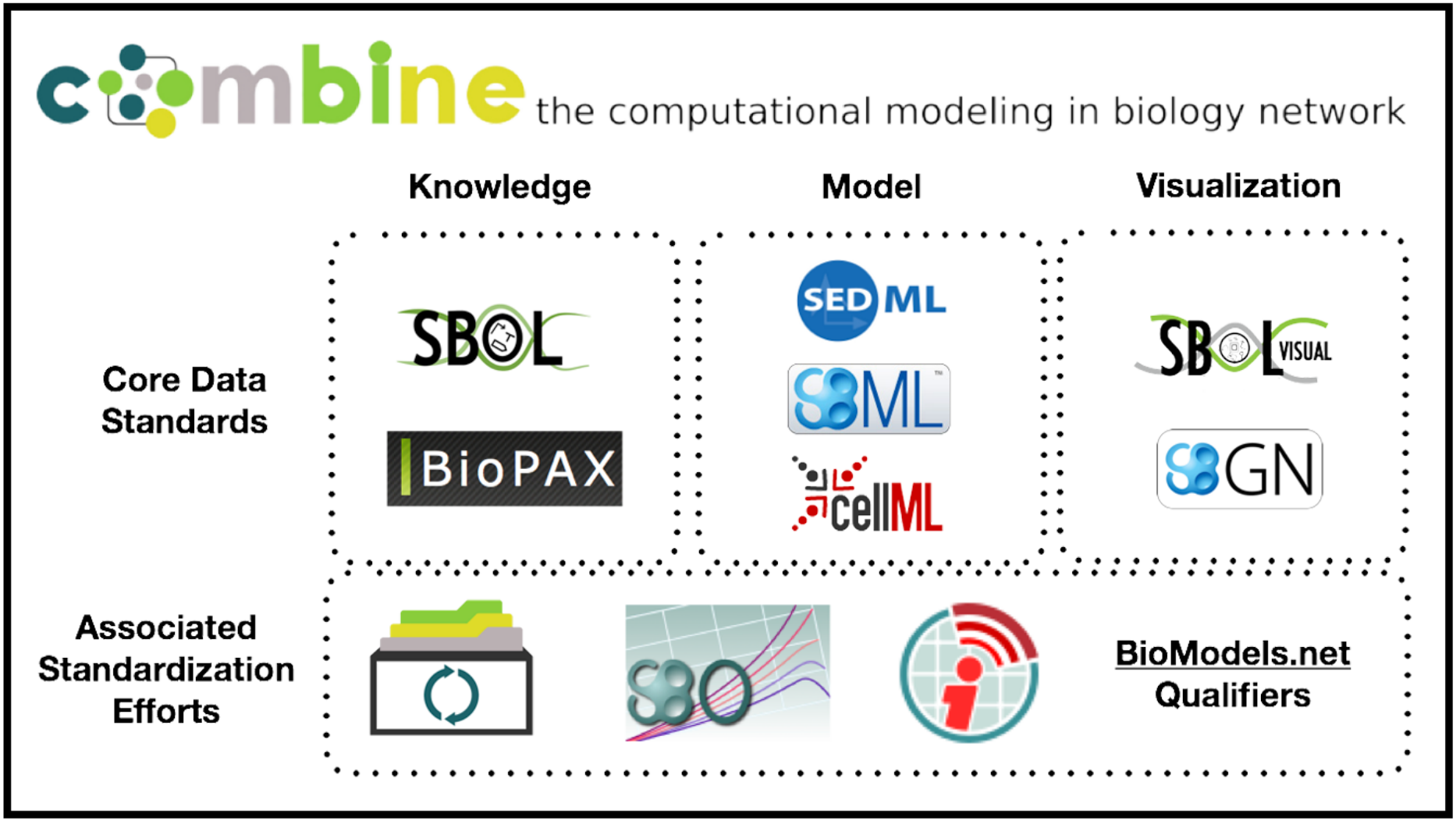


Library Creation



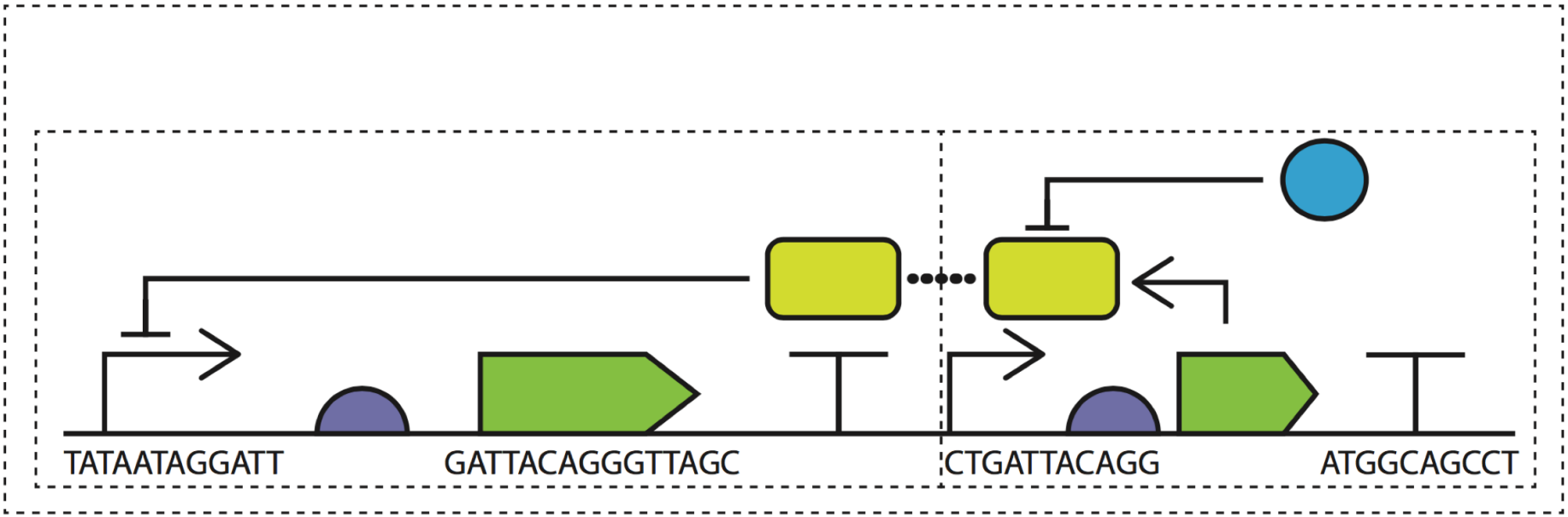
- Biological parts are encoded into data standards and have been deposited onto online repositories for programmable access.

Data Standards



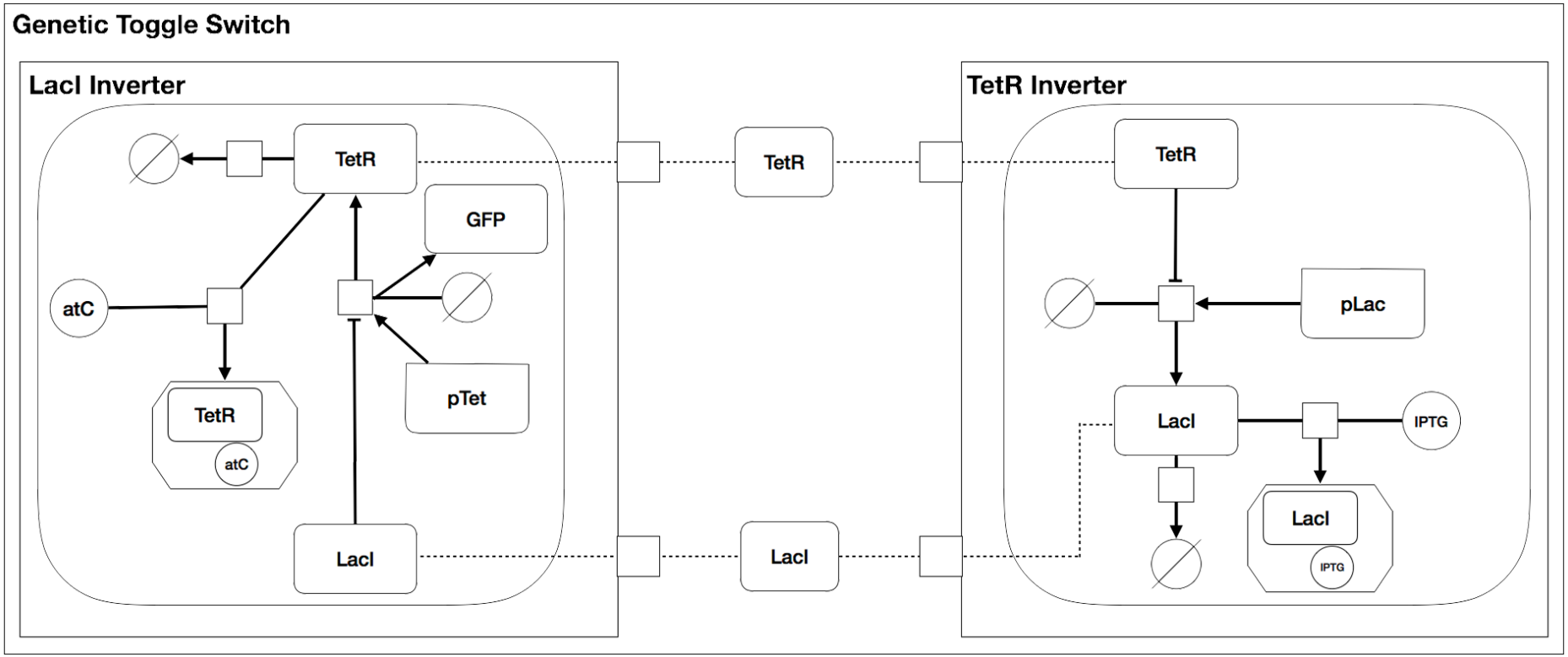
Synthetic Biology Open Language (SBOL)

- An **open standard** for representing *in silico* **biological designs**.

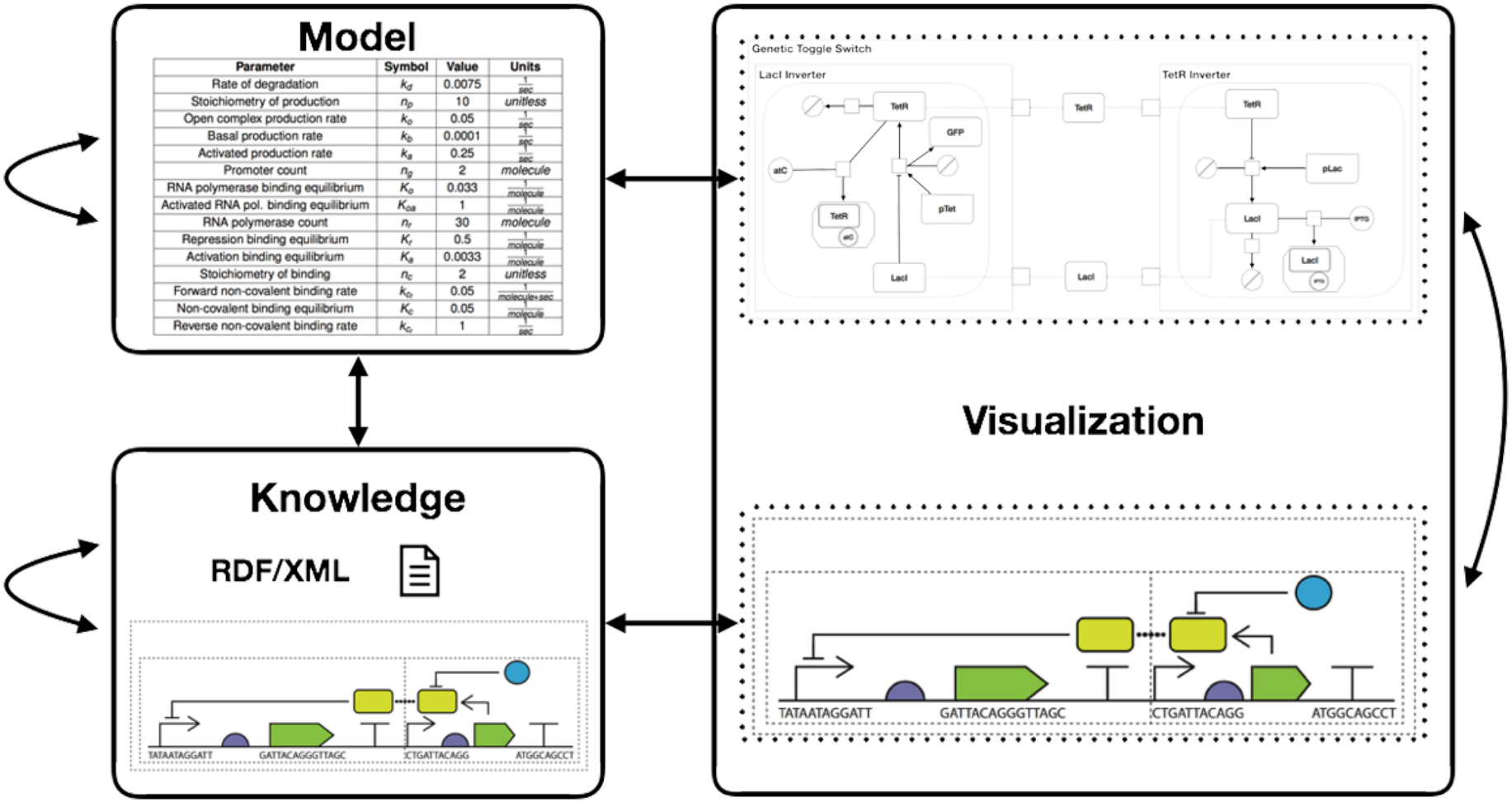


Systems Biology Markup Language (SBML)

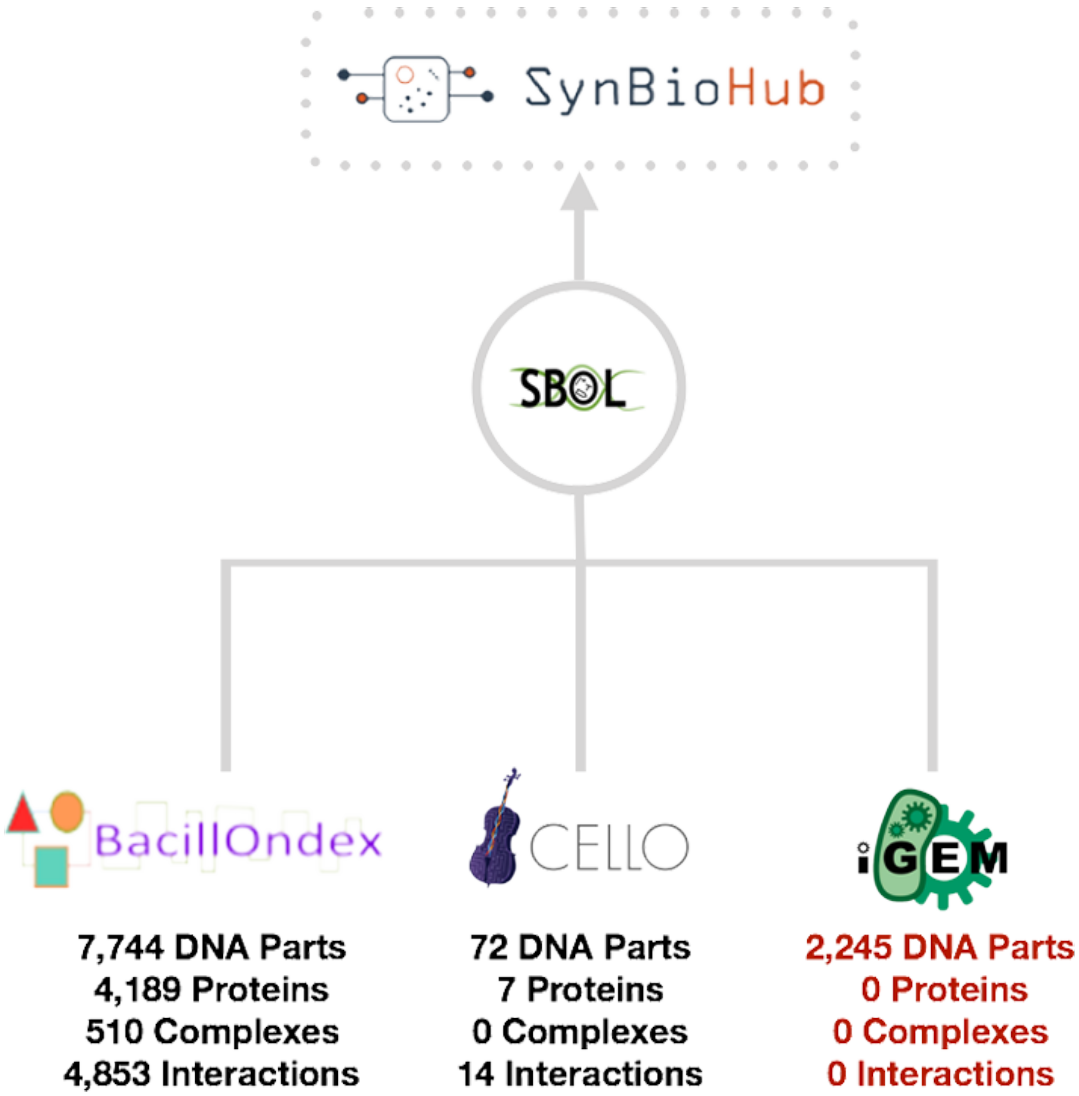
- An **open standard** to describe the behavioral models of **biological systems**.



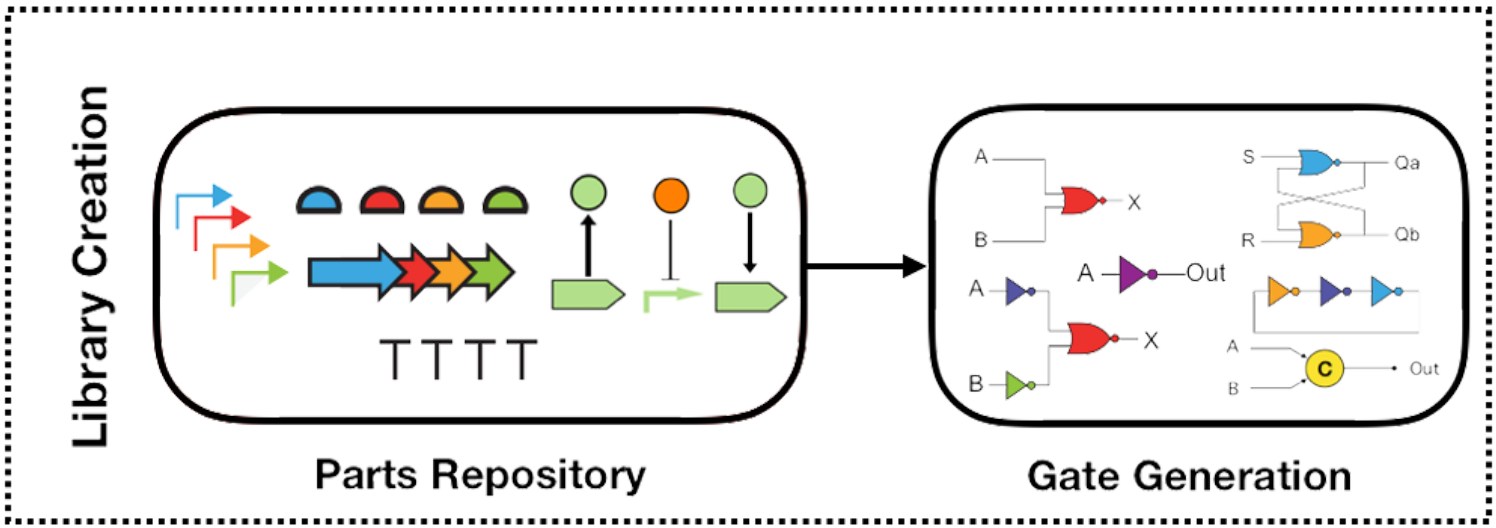
Data Standard Conversions



Available Datasets

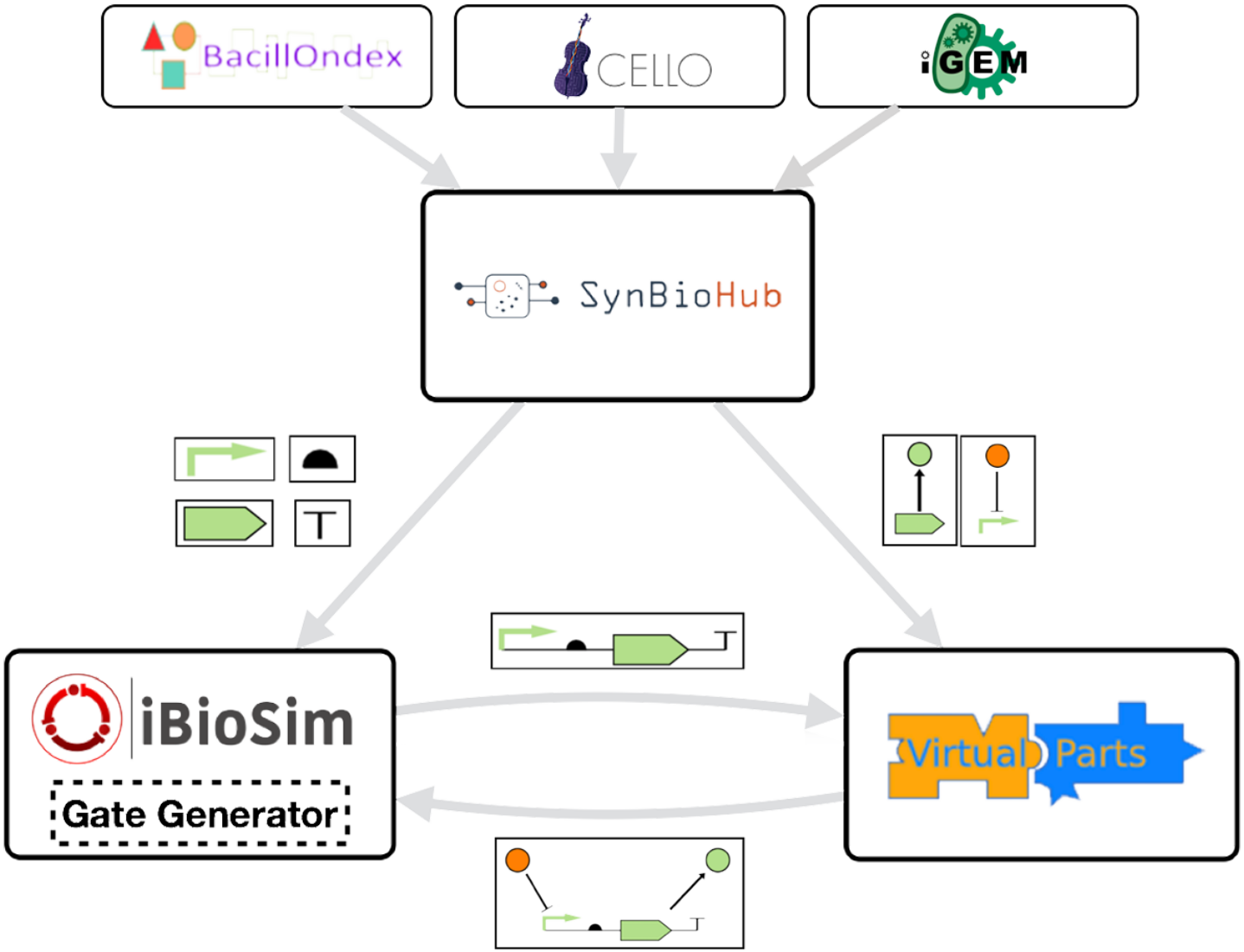


Gate Generation

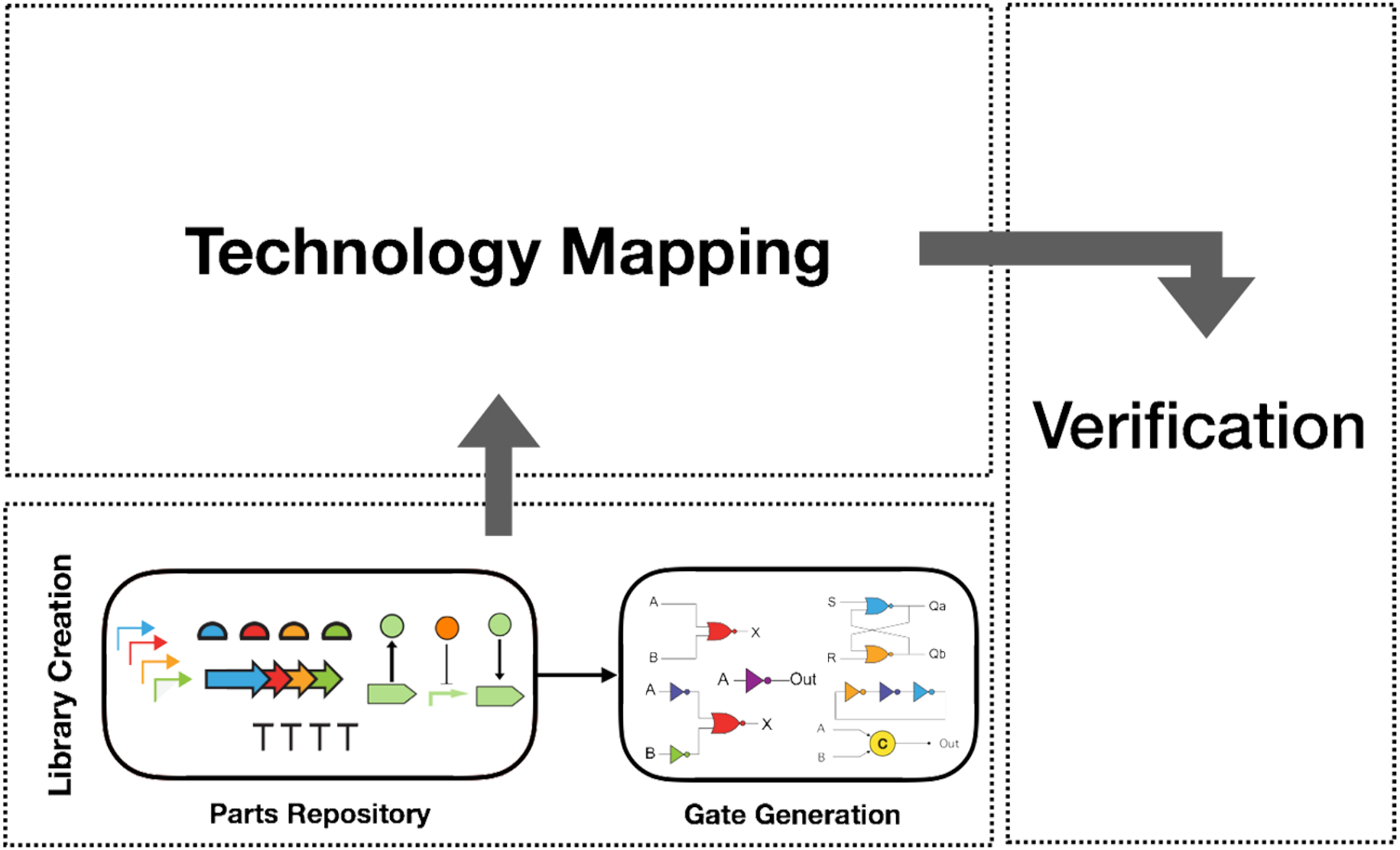


- Gate generation automates the process of producing gates from a library of parts.

Automating Library Creation



Library Creation



Existing Technology Mapping

	MatchMaker[1]	SBROME[2]	iBioSim[3]	Cello[4]	GeneTech[5]
Specification Type	Abstract genetic regulatory network	Abstract genetic regulatory network	Abstract genetic regulatory network	Verilog	Boolean expressions
Support COMBINE Standard	SBOL1	–	SBOL2 and SBML	SBOL2	–
Library	Arbitrary logic gates	Parts	Arbitrary logic gates	NOT and NOR gates	NOT and NOR gates
Signal Mismatch	✓	✗	✗	✓	✗
Crosstalk	✗	✓	✓	✓	✓
Genetic Context Effects	✗	✗	✗	✓	✗
Roadblock	✗	✗	✗	✓	✗

✓Supported in tool.

✗Not supported in tool.

[1] Yaman et al, *ACS Publications*, 2012

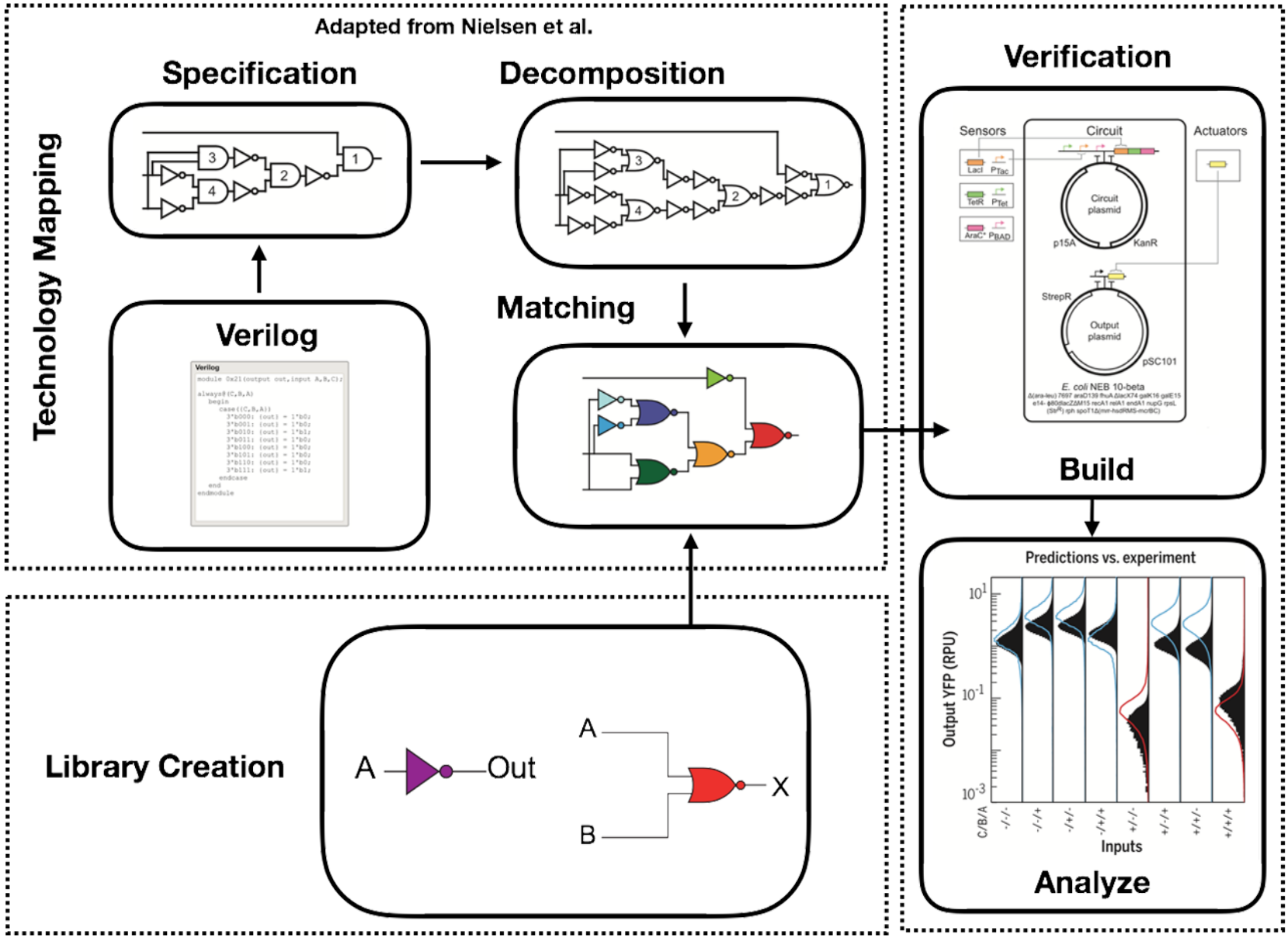
[2] Huynh et al., *ACS Synthetic Biology*, 2013

[3] Roehner et al., *ACS Synthetic Biology*, 2014

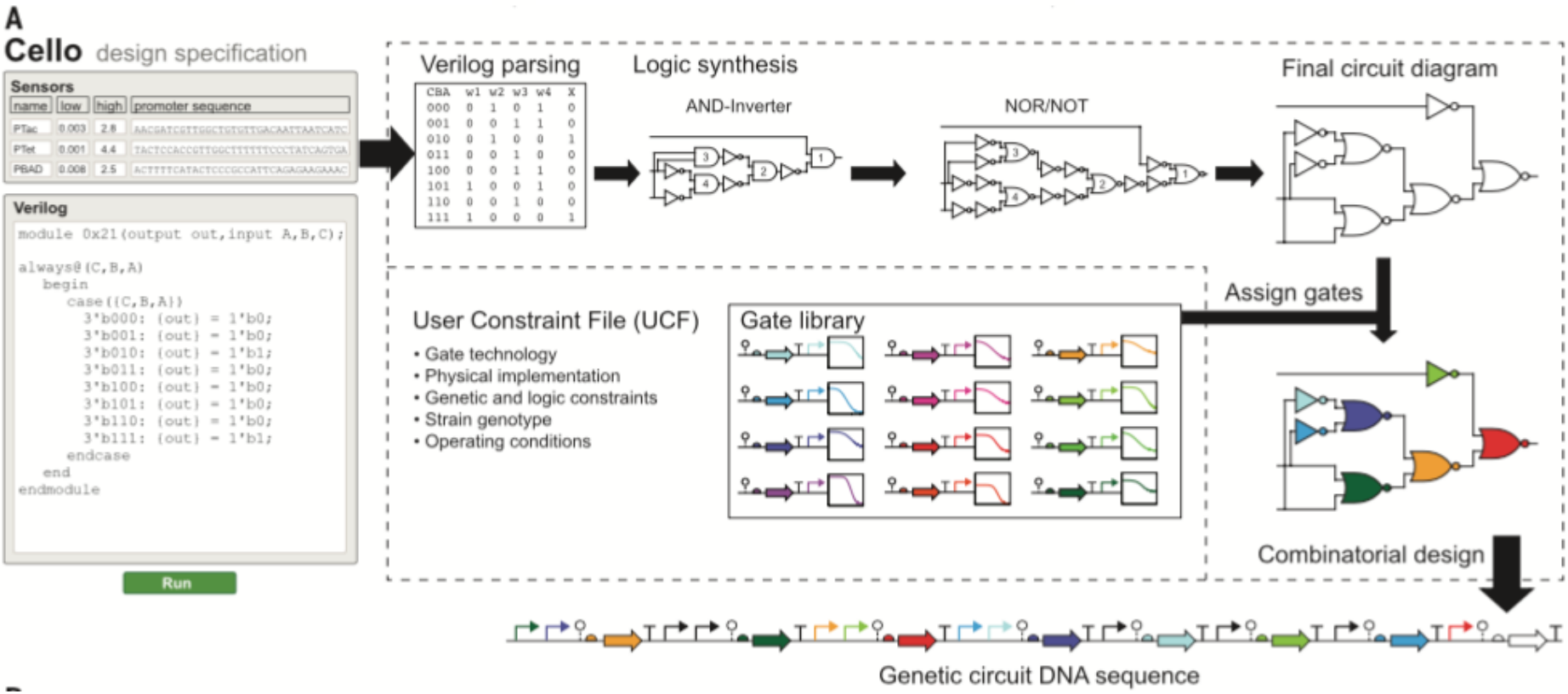
[4] Nielsen et al., *Science*, 2016

[5] Baig et al., *9th International Workshop on Bio-Design Automation*, 2017

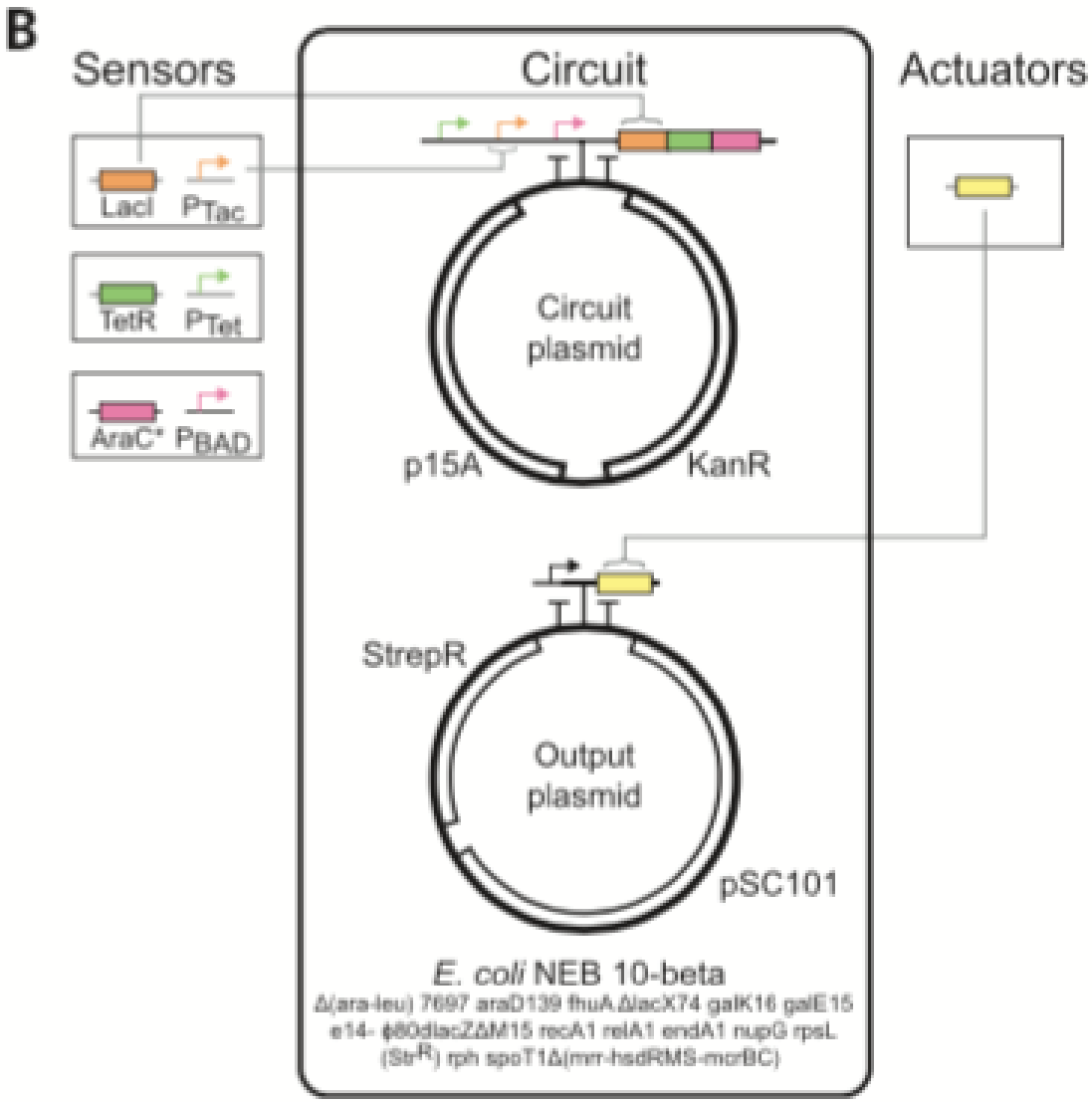
Cello's Technology Mapping



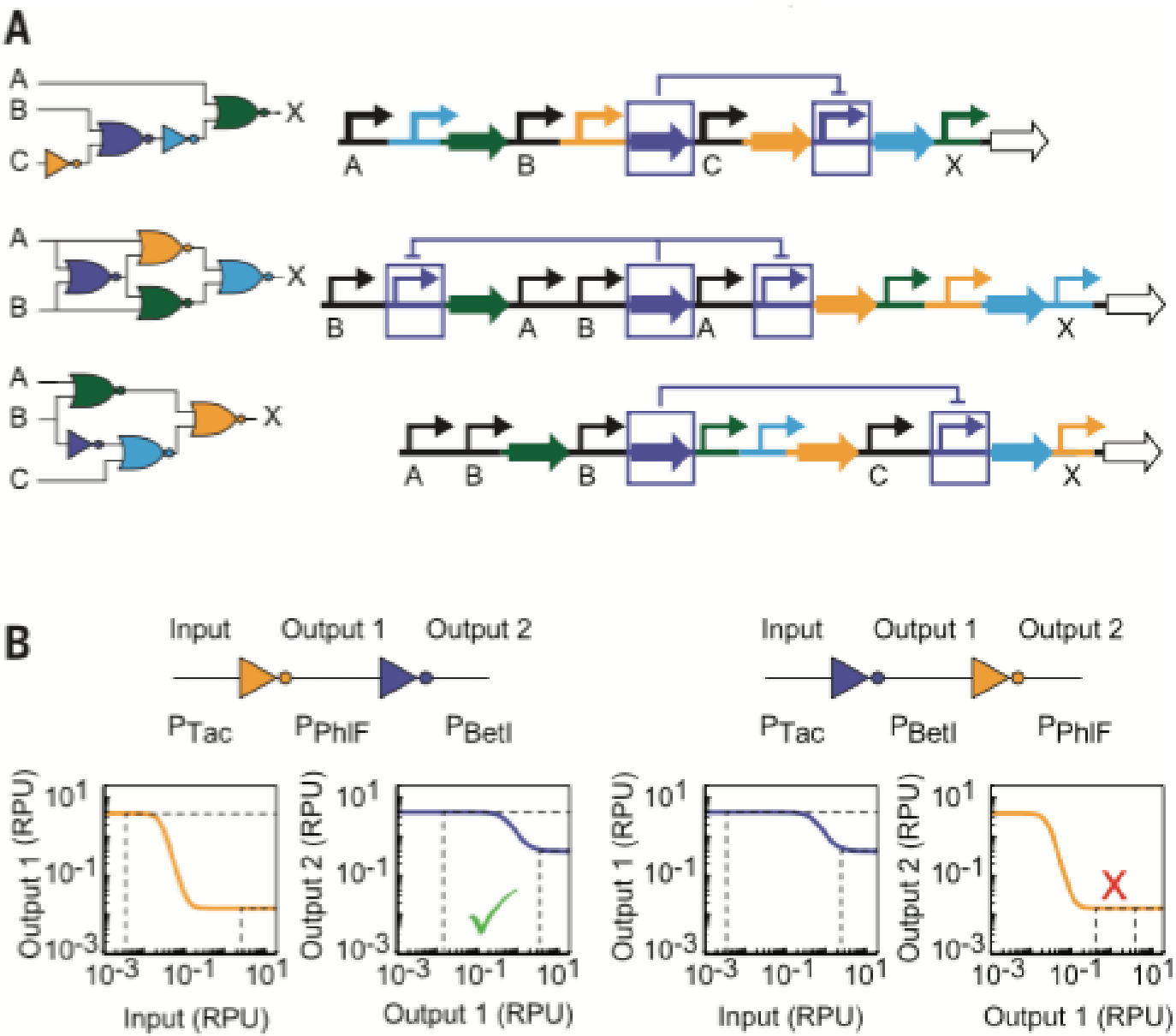
Overview of Cello



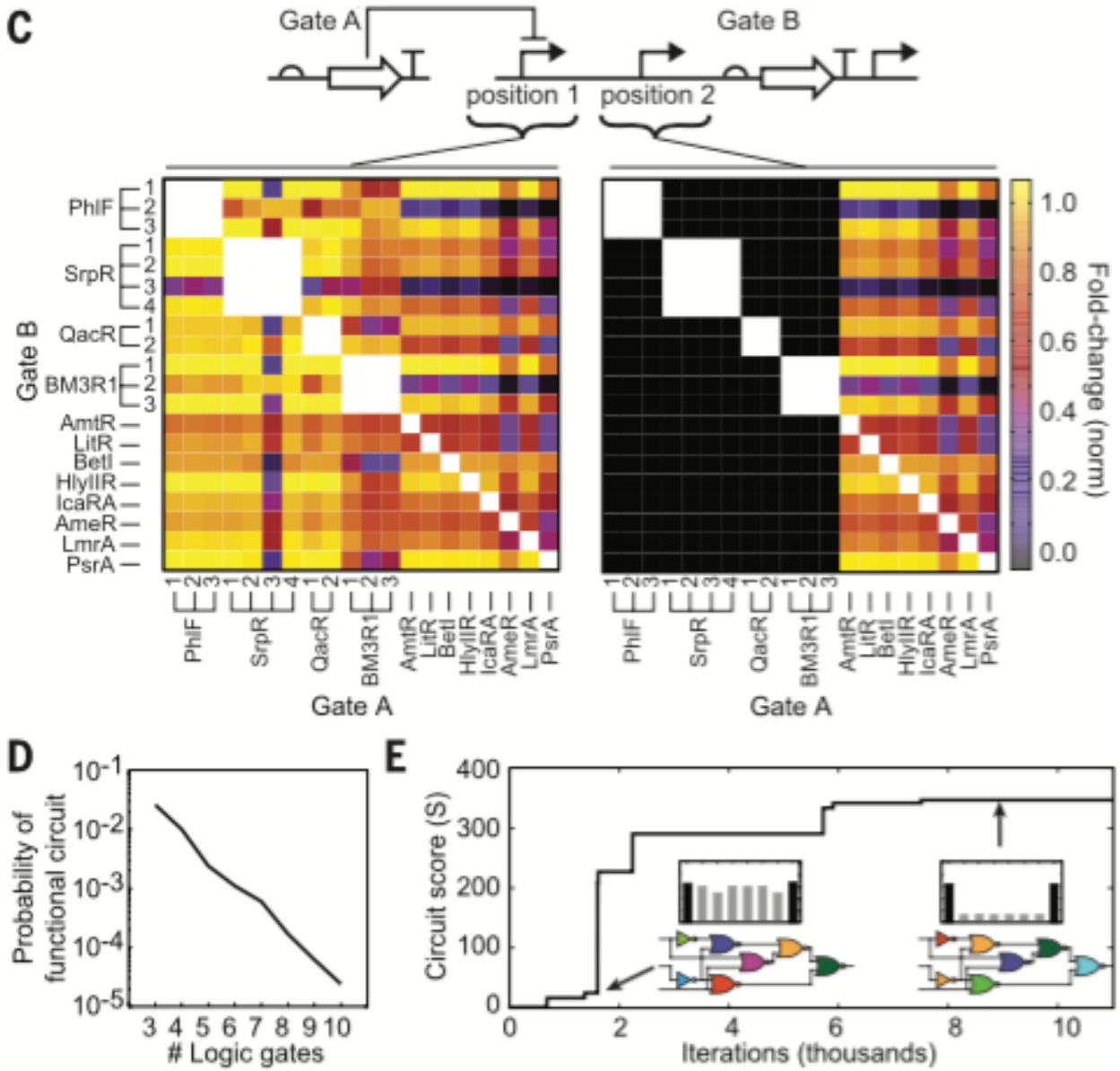
Overview of Cello (cont)



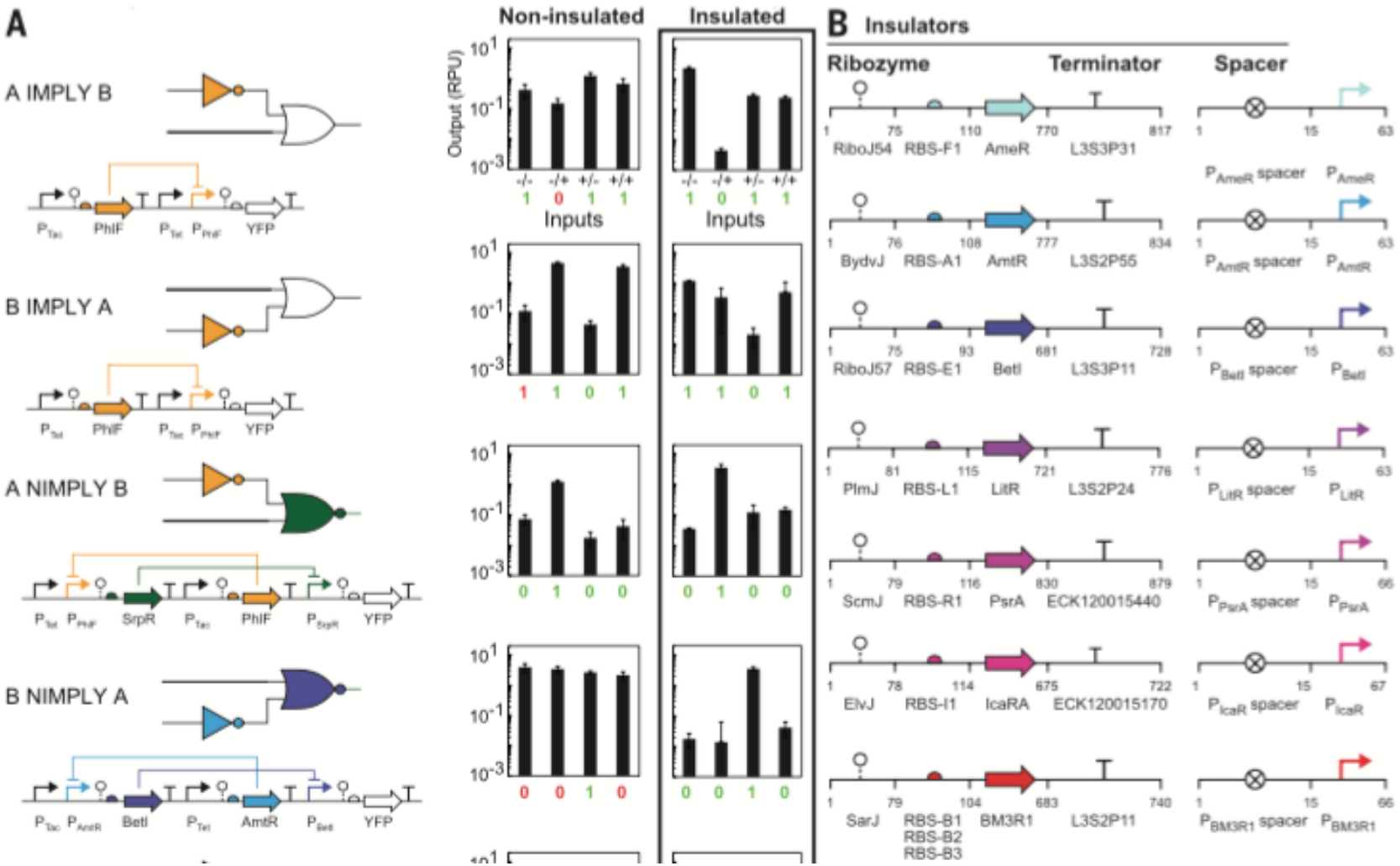
Genetic Gate Assignment



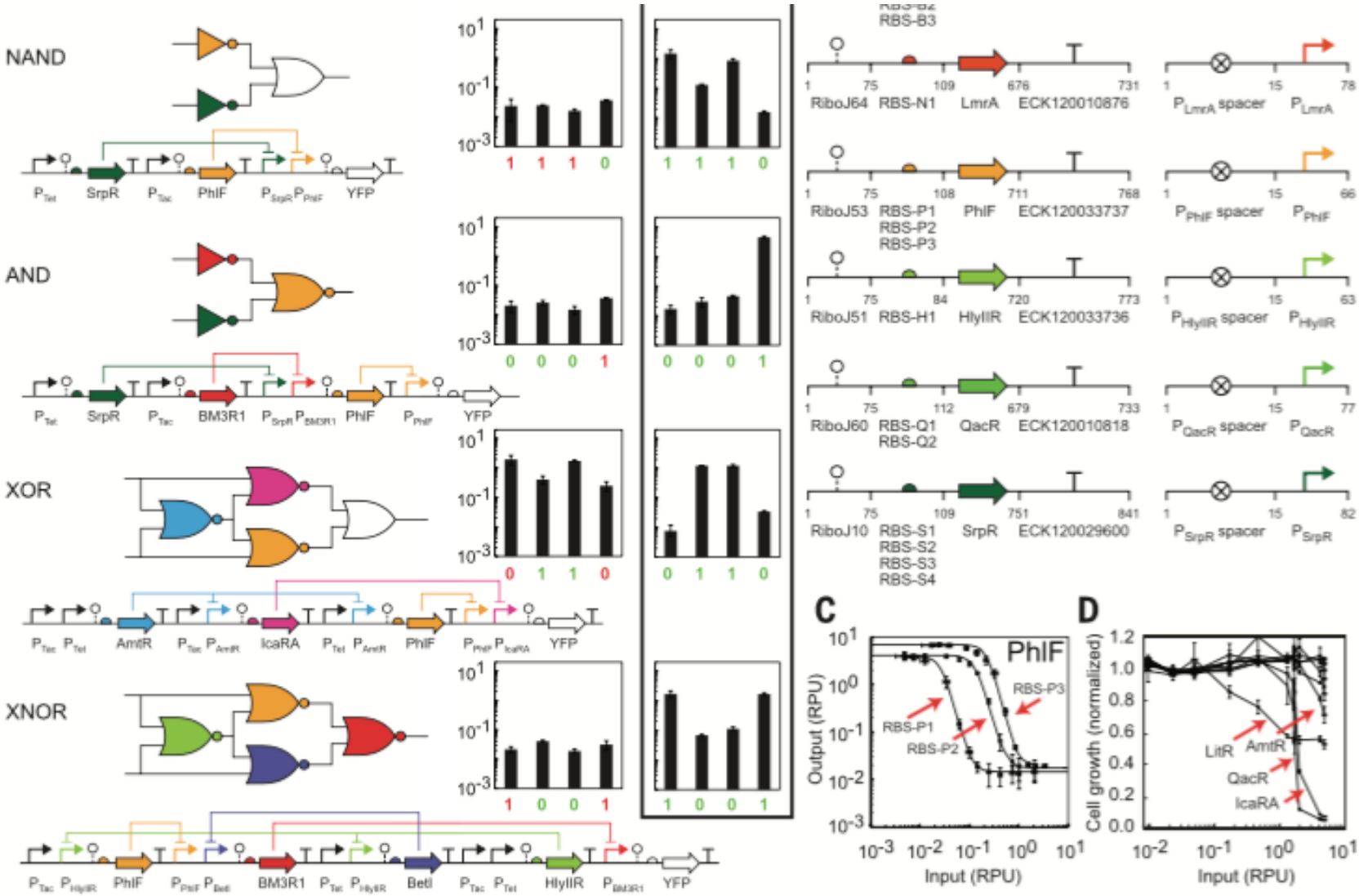
Genetic Gate Assignment (cont)



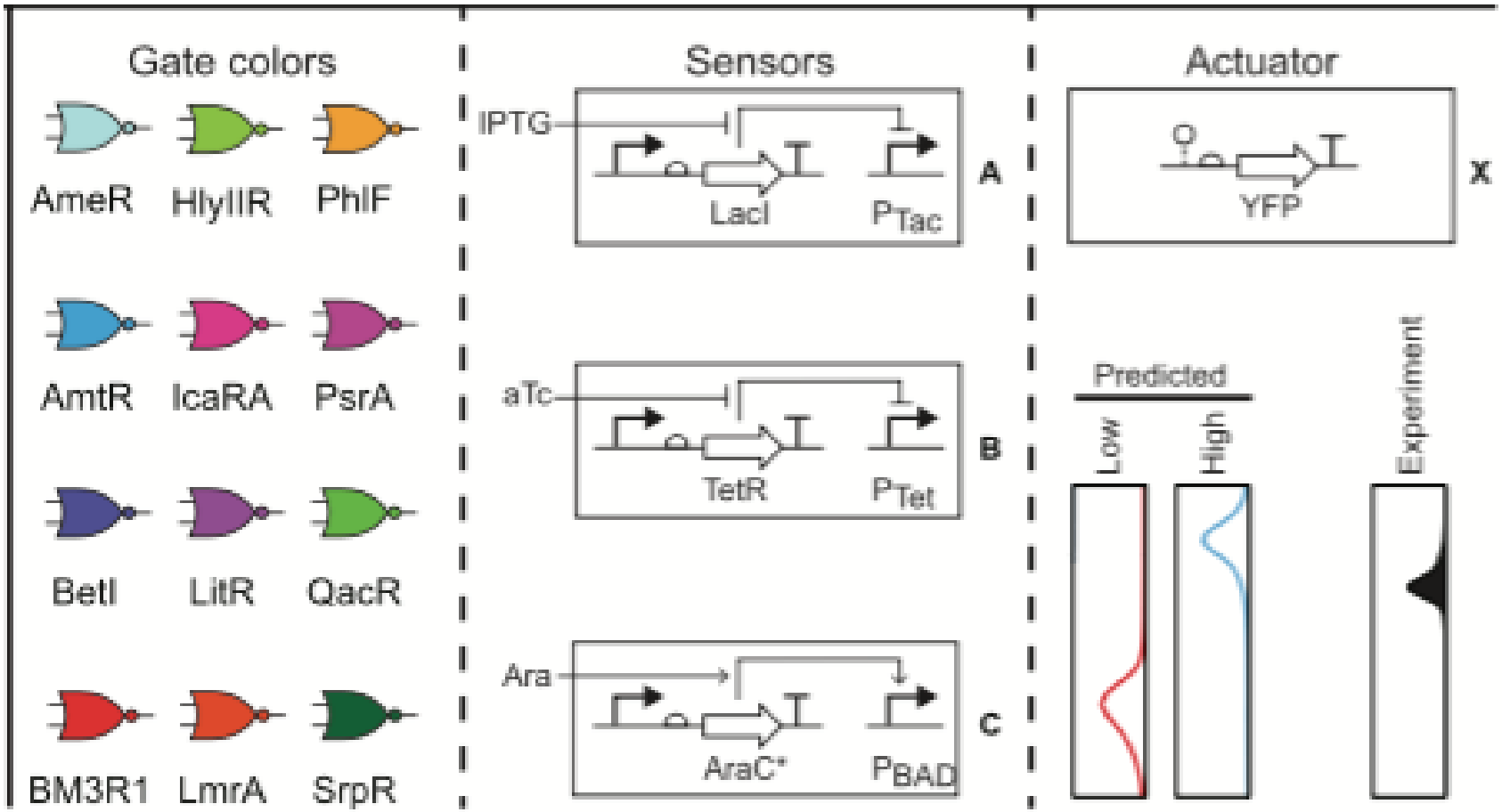
Impact of Gate Isolation



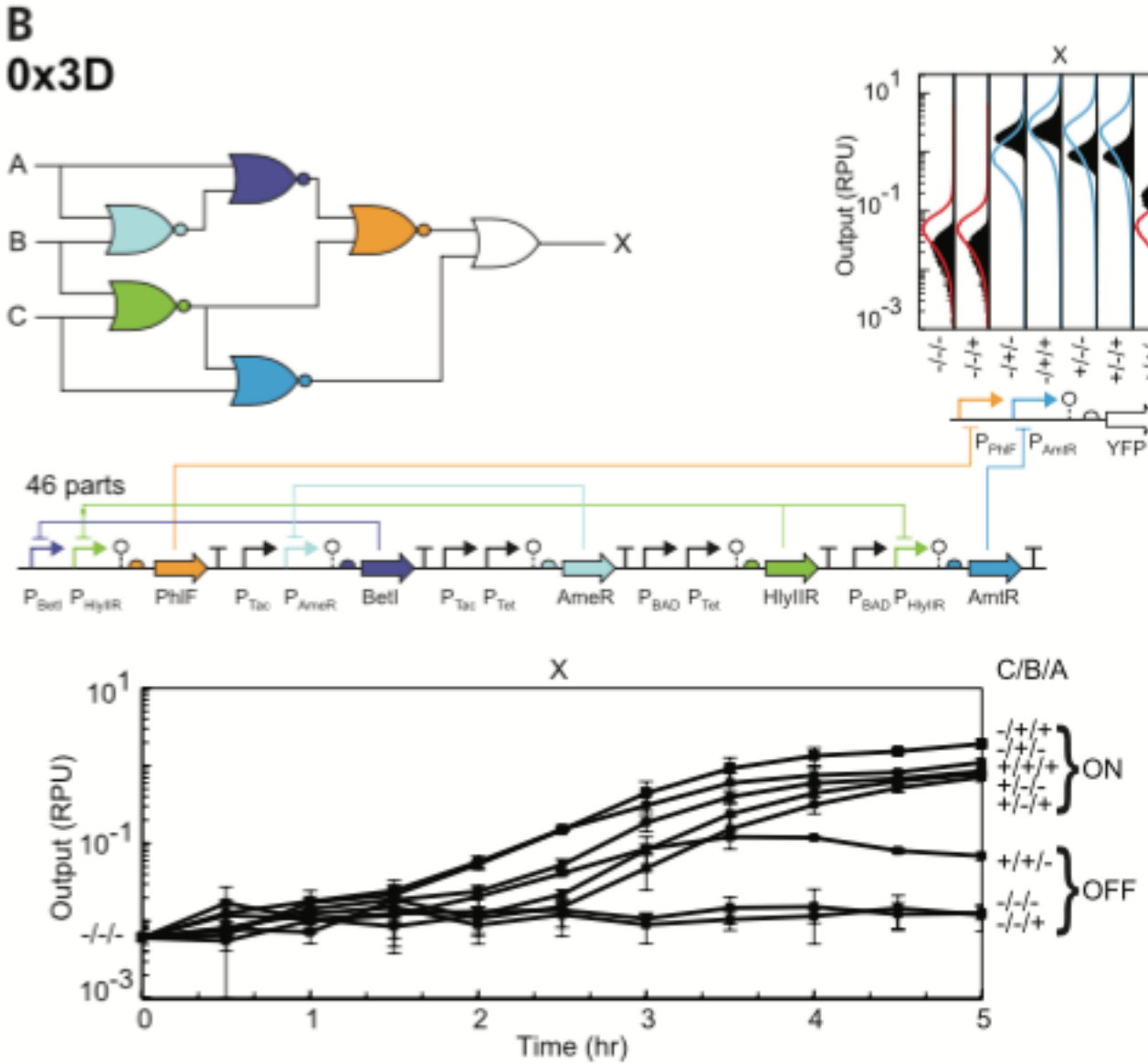
Impact of Gate Isolation (cont)



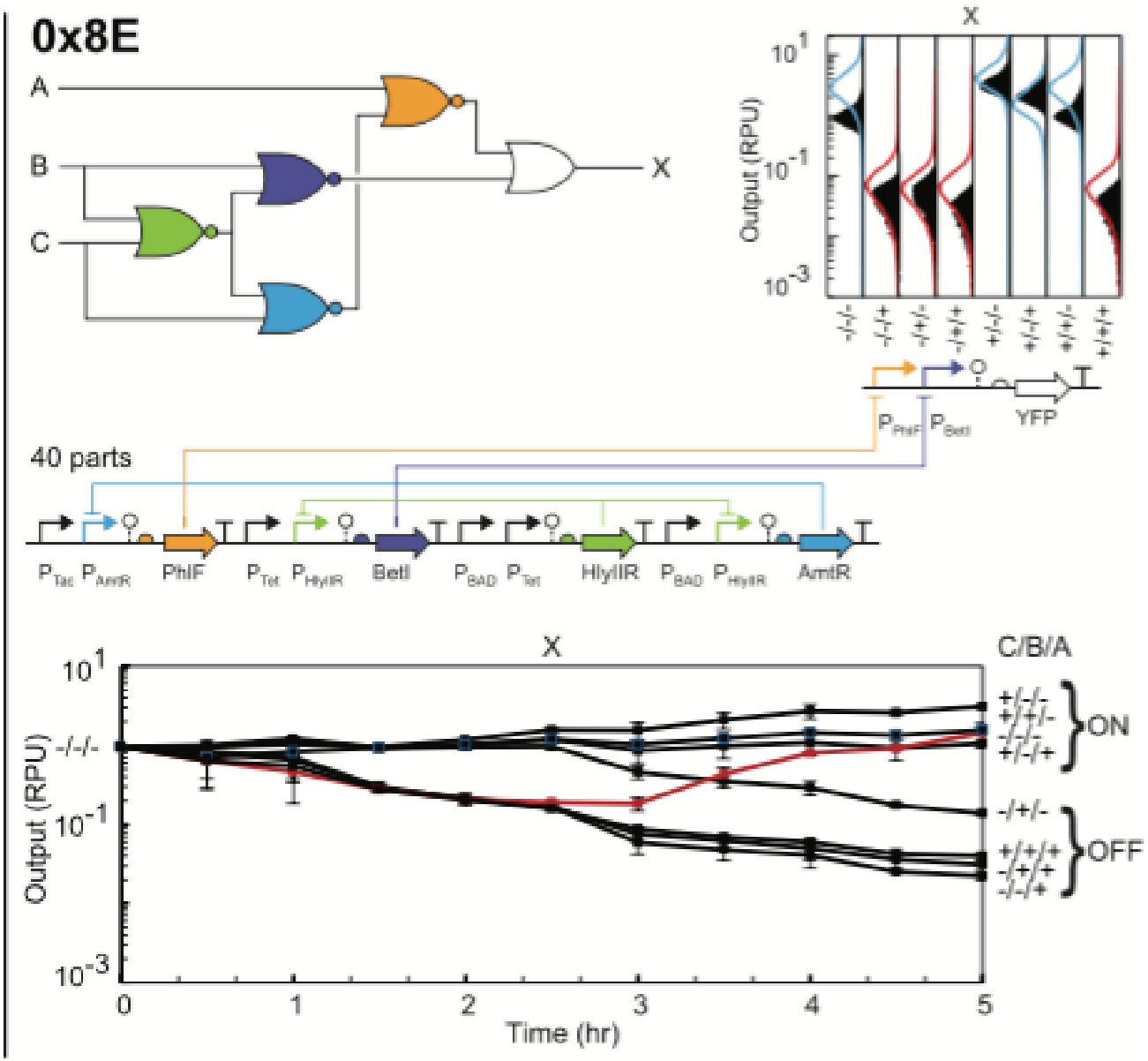
Cello Gate Library



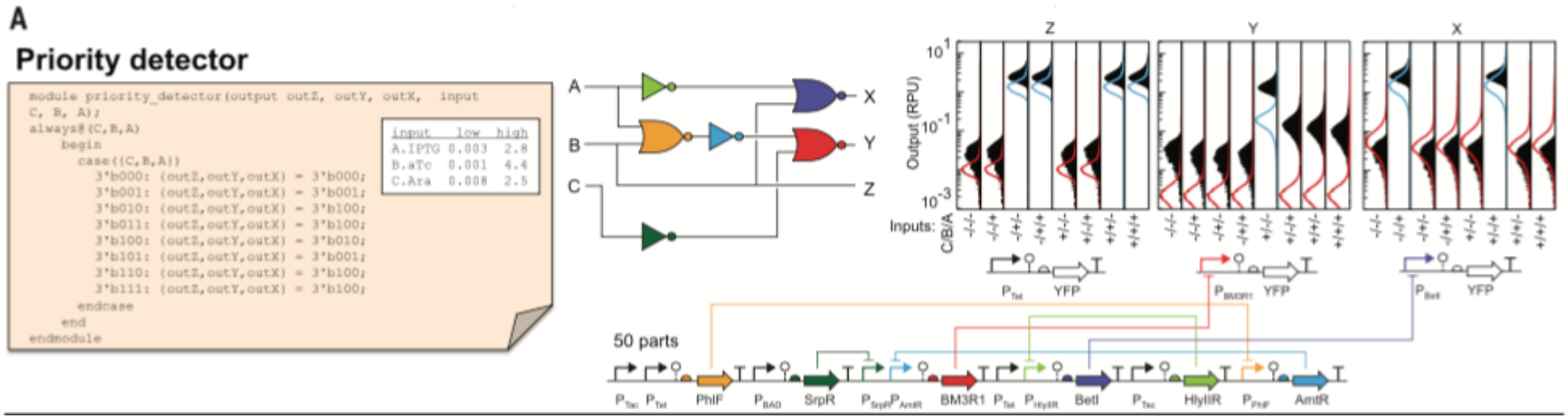
Cello Circuit Example 1



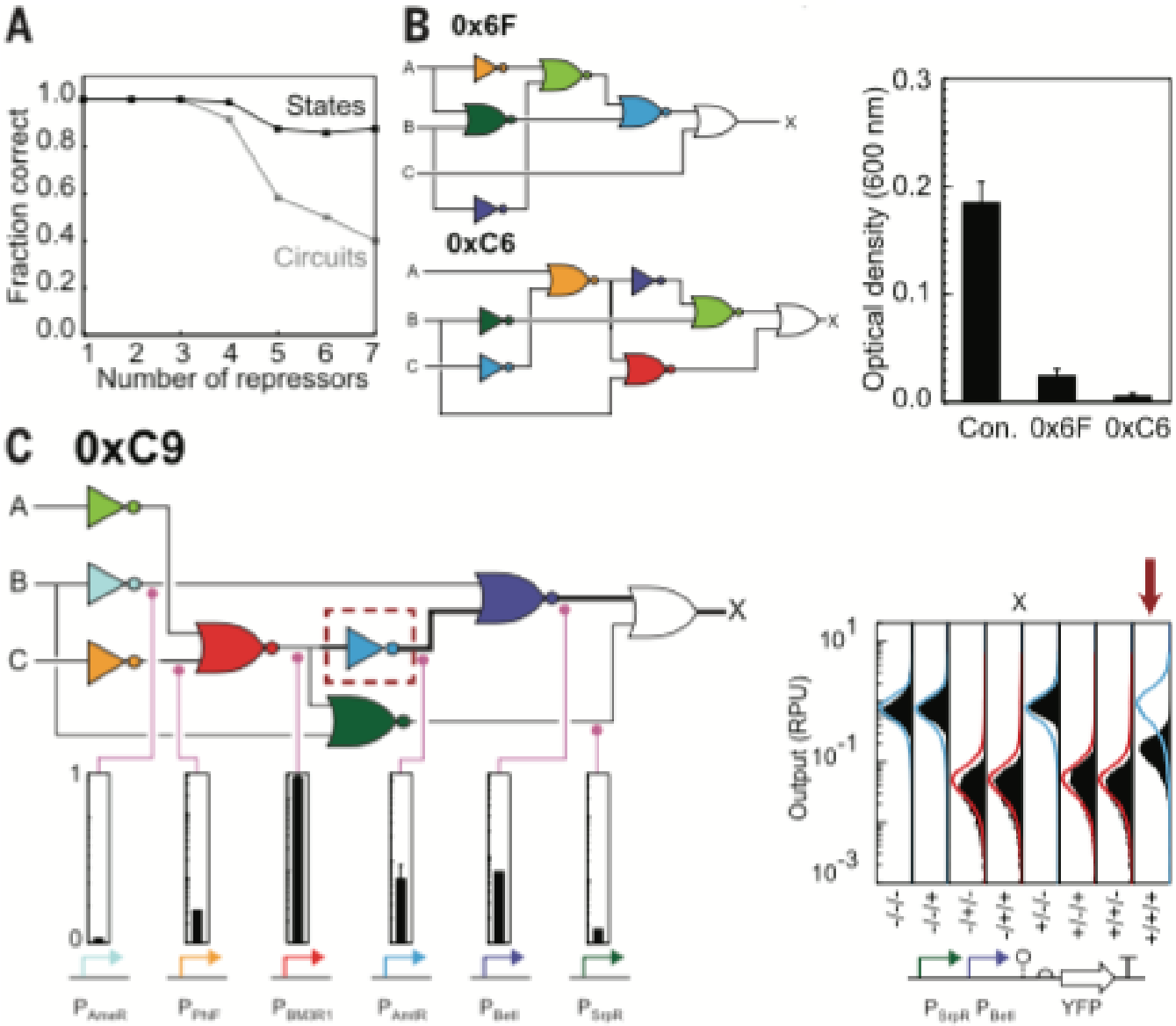
Cello Circuit Example 2



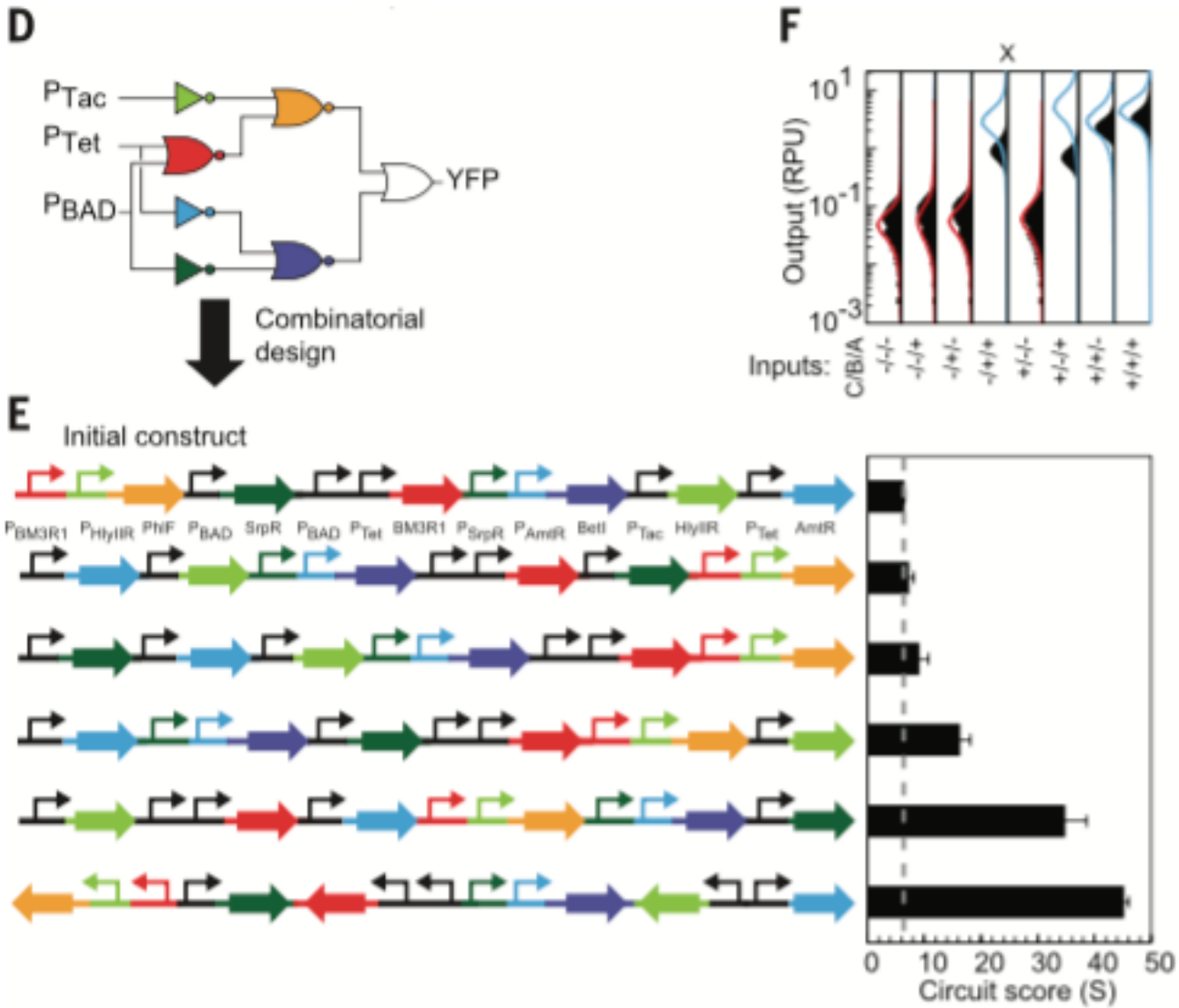
Cello Priority Detector



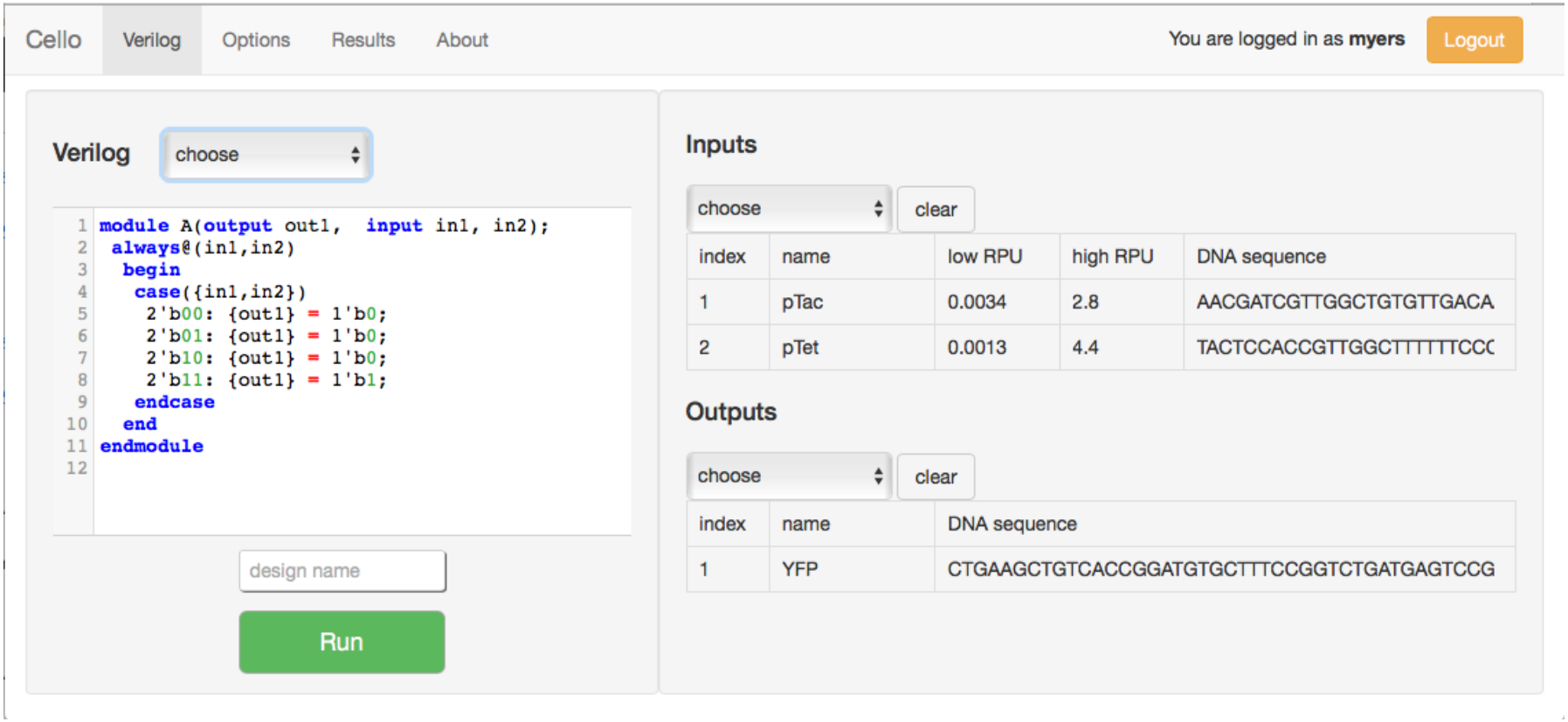
Analysis of Circuit Failures



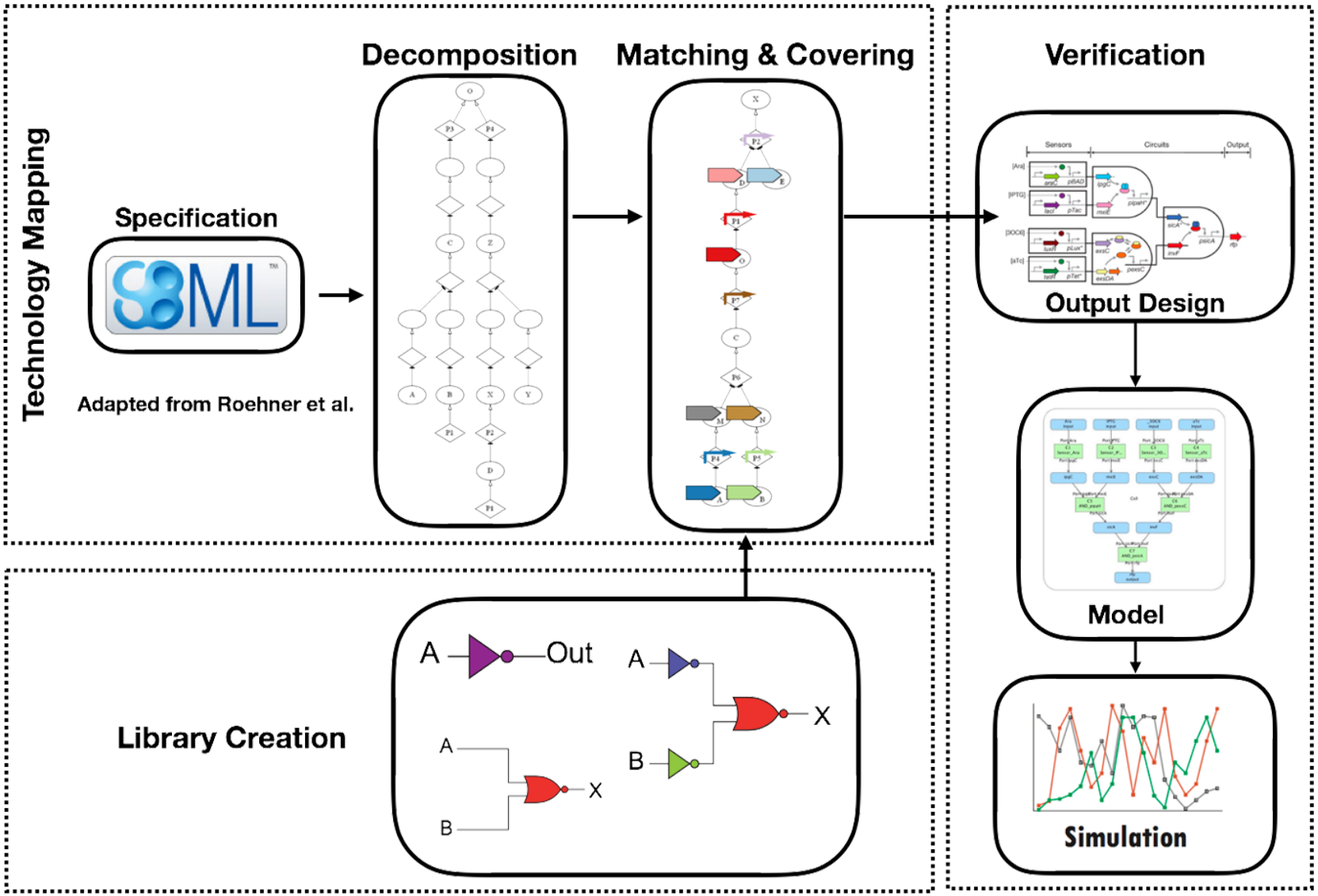
Analysis of Circuit Failures (cont)



cellocad.org



iBioSim's Technology Mapping



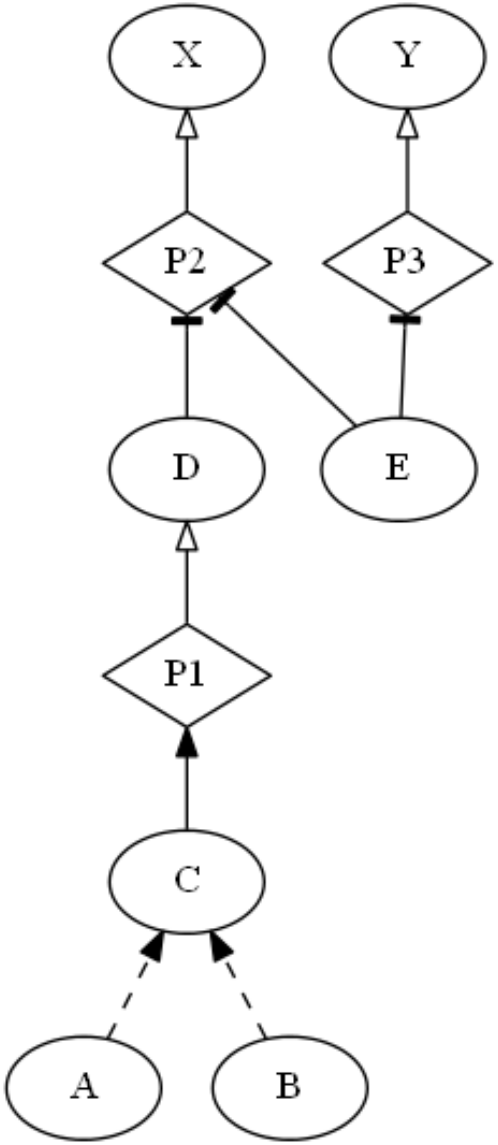
DAG-Based Technology Mapping

1. DAG Representation
2. Partitioning and Decomposition
3. Matching and Covering

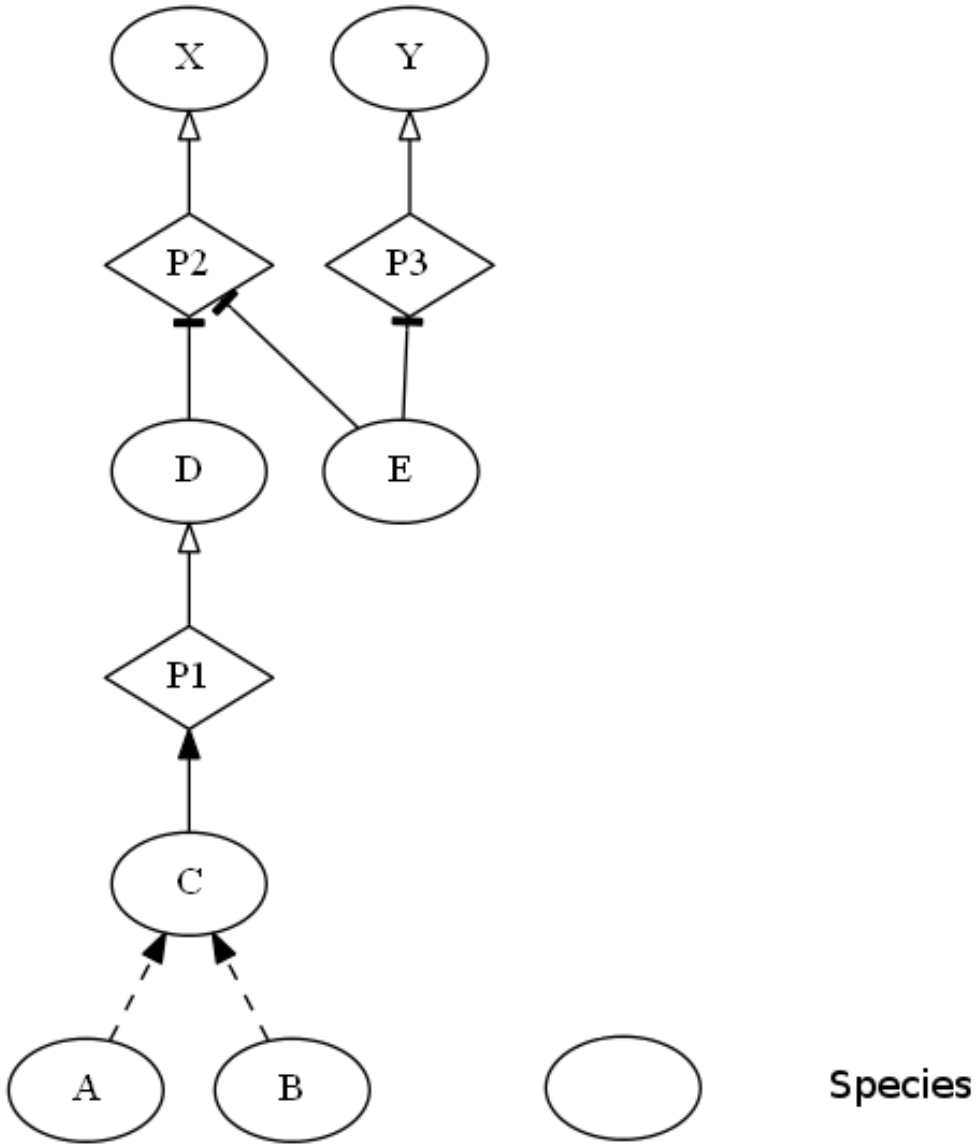
DAG-Based Technology Mapping

1. DAG Representation
2. Partitioning and Decomposition
3. Matching and Covering

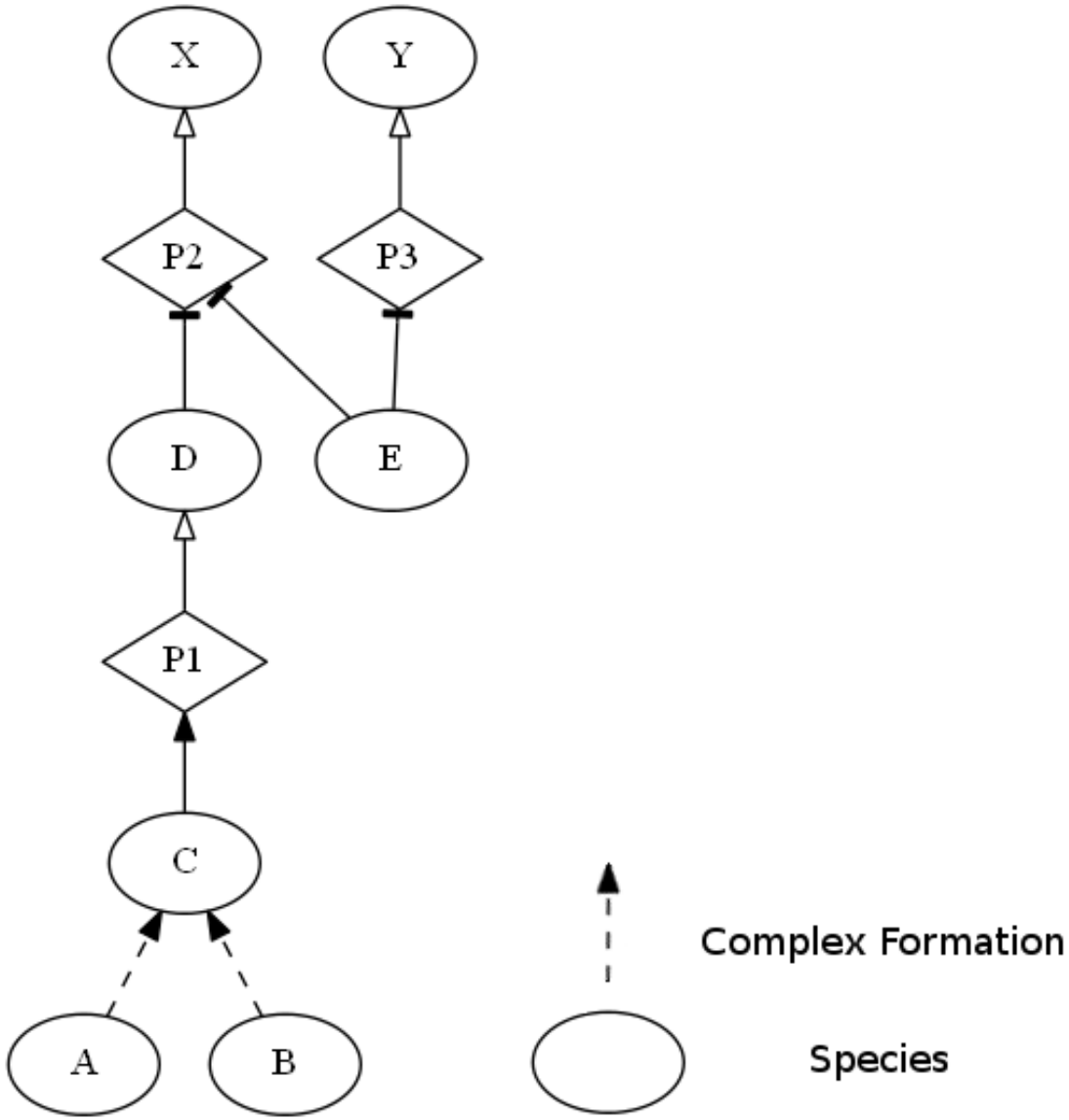
DAG Representation of Specification



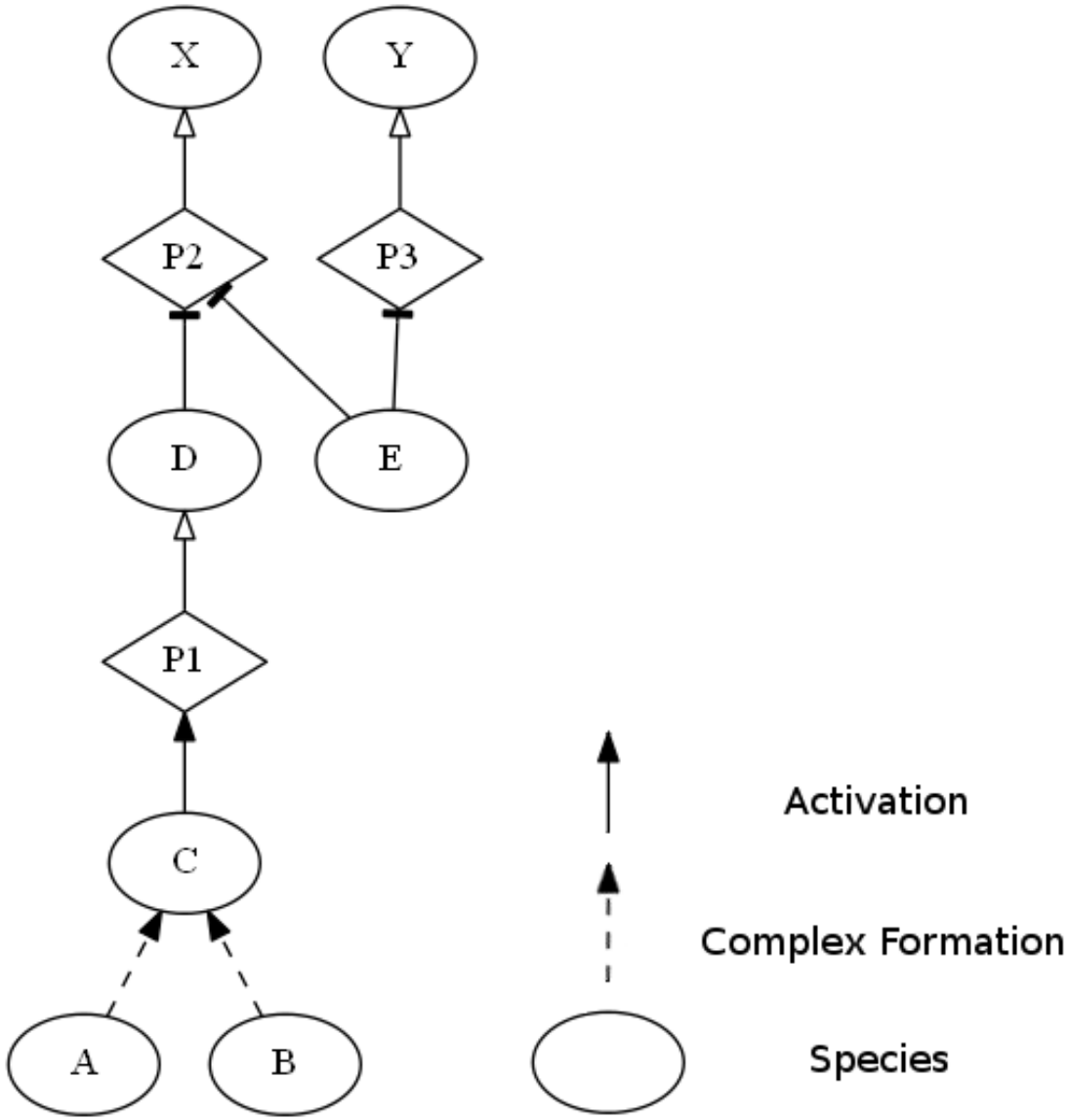
DAG Representation of Specification



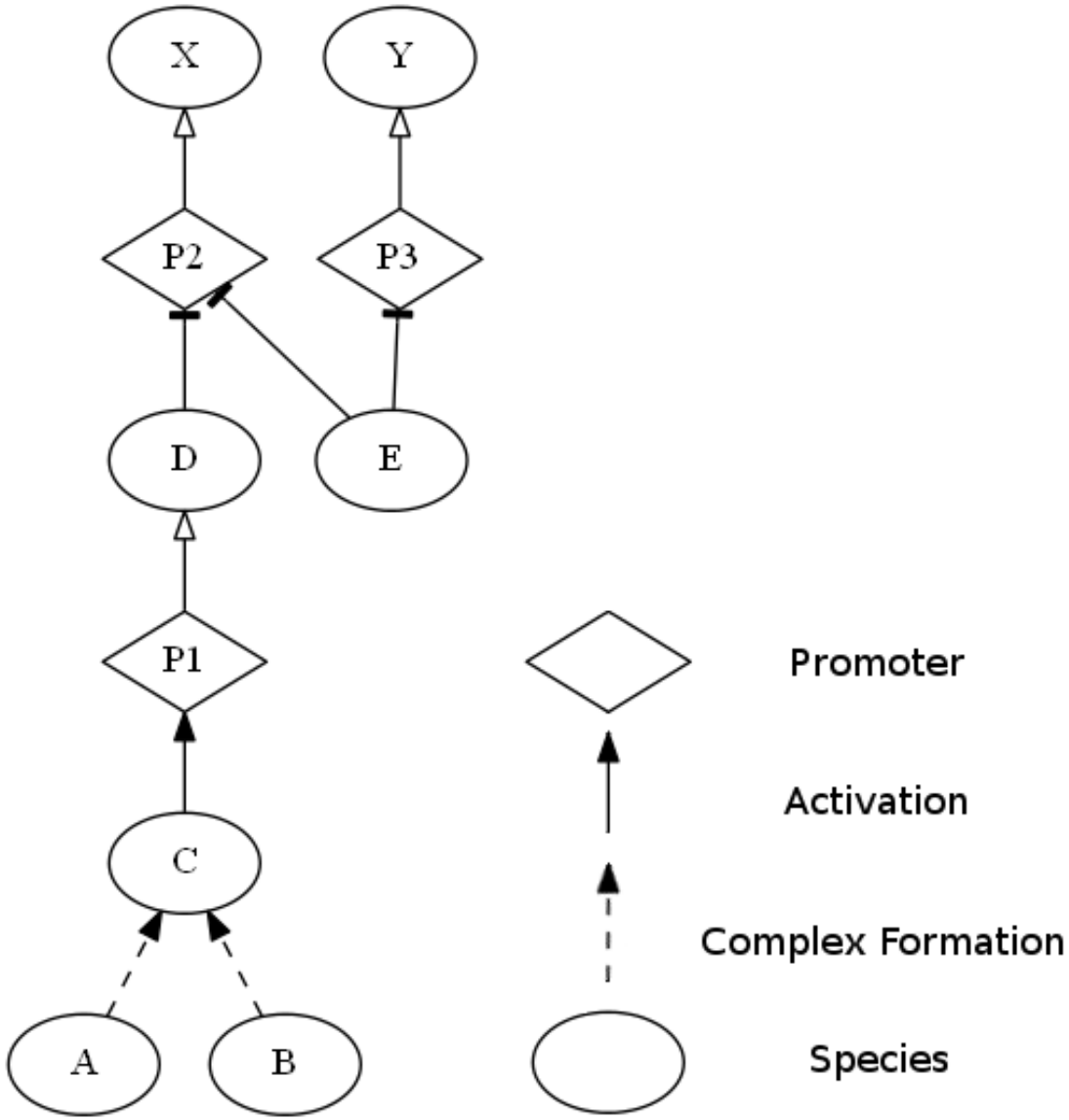
DAG Representation of Specification



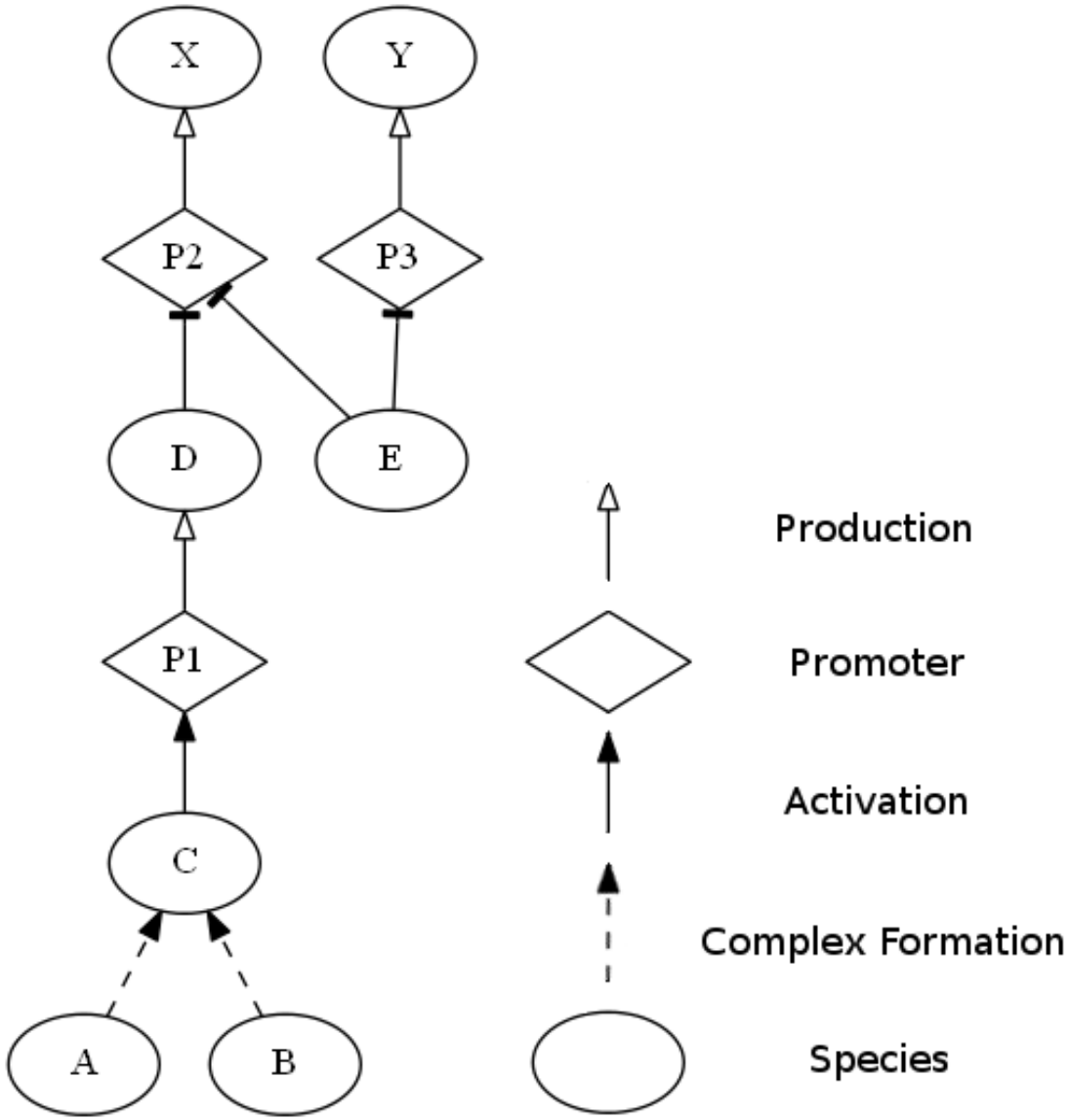
DAG Representation of Specification



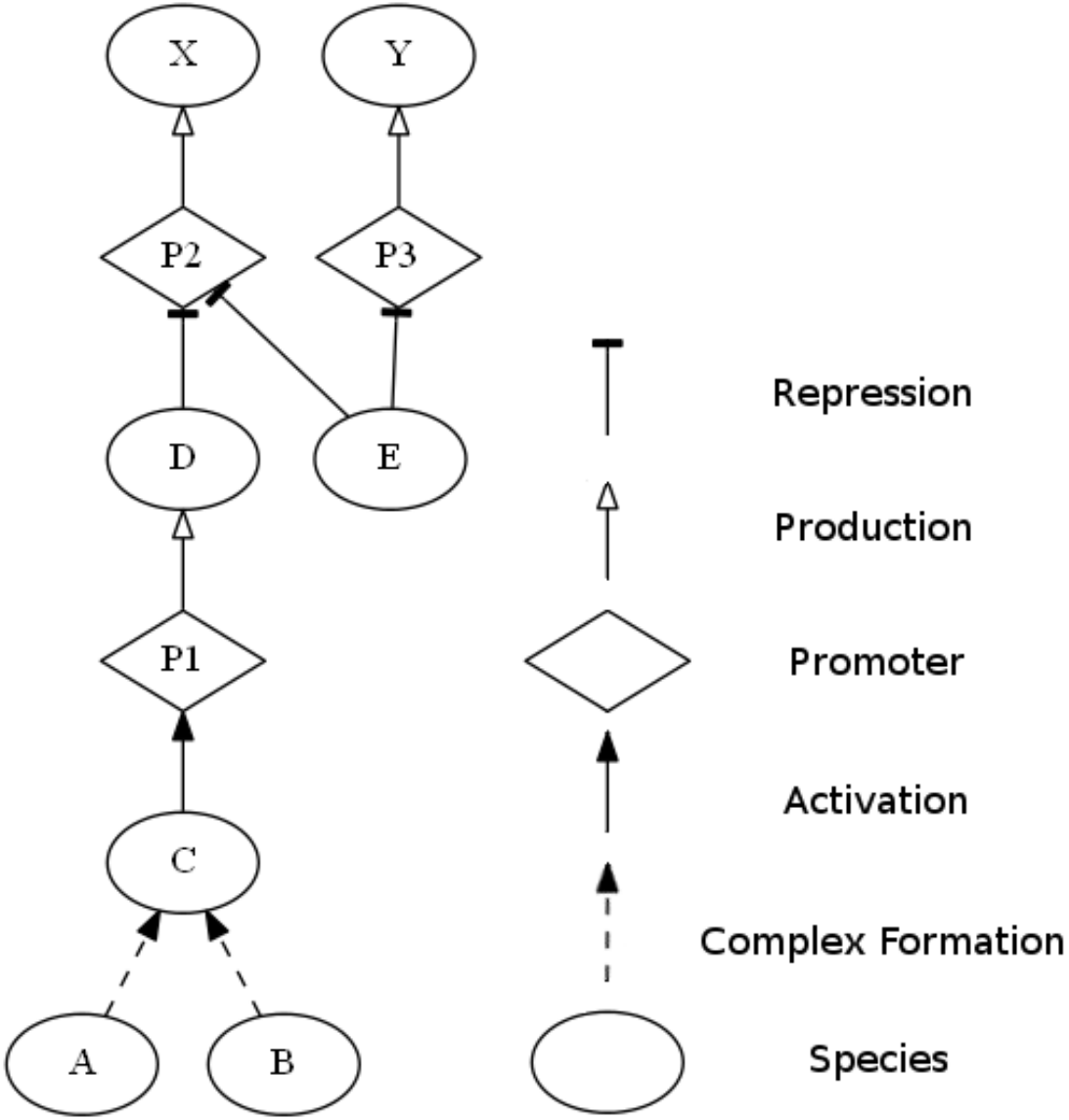
DAG Representation of Specification



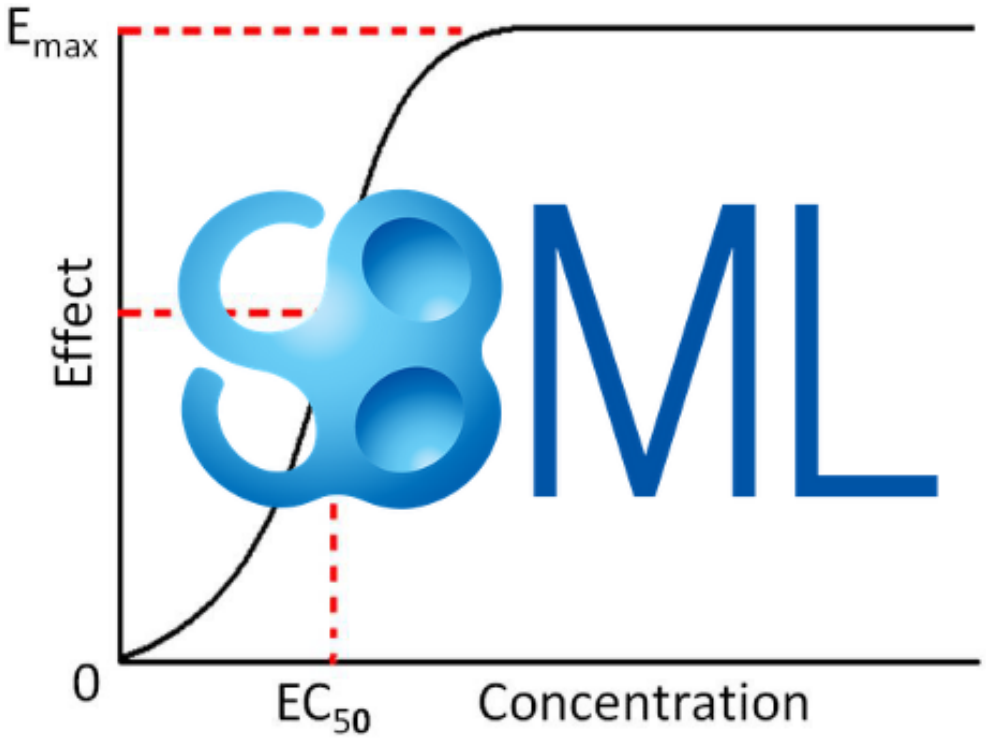
DAG Representation of Specification



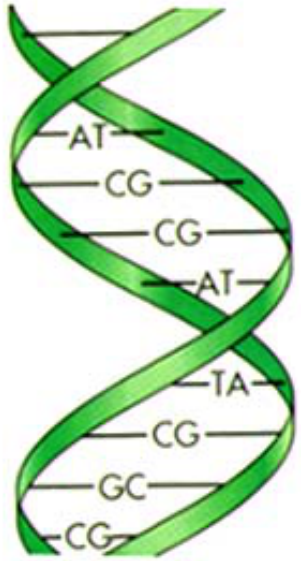
DAG Representation of Specification



Library

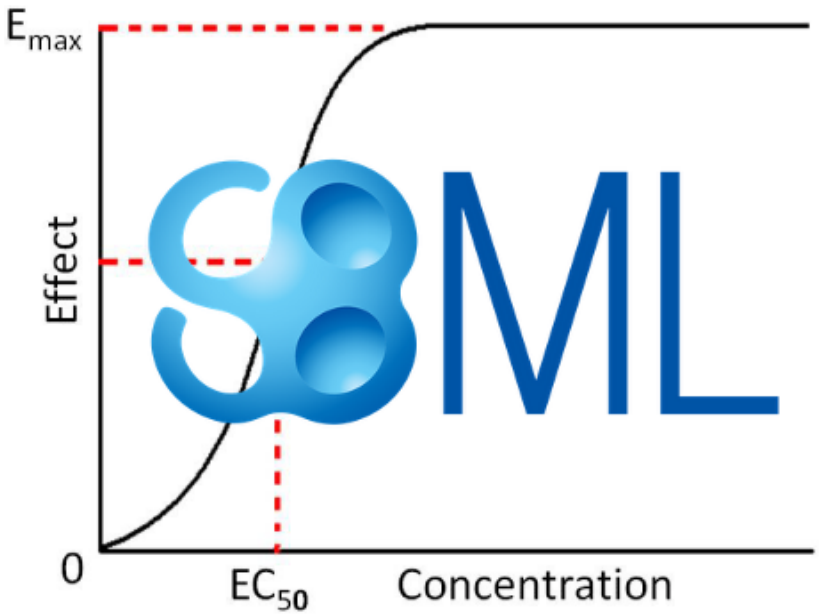


RDF/XML
Annotation



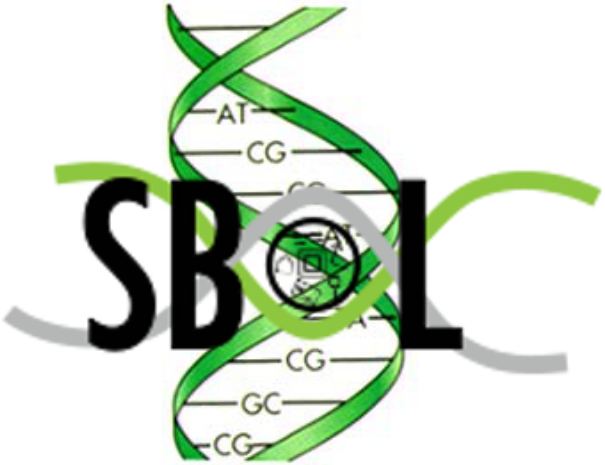
Roehner et al., ACS Synthetic Biology (2013).

Library



SBML

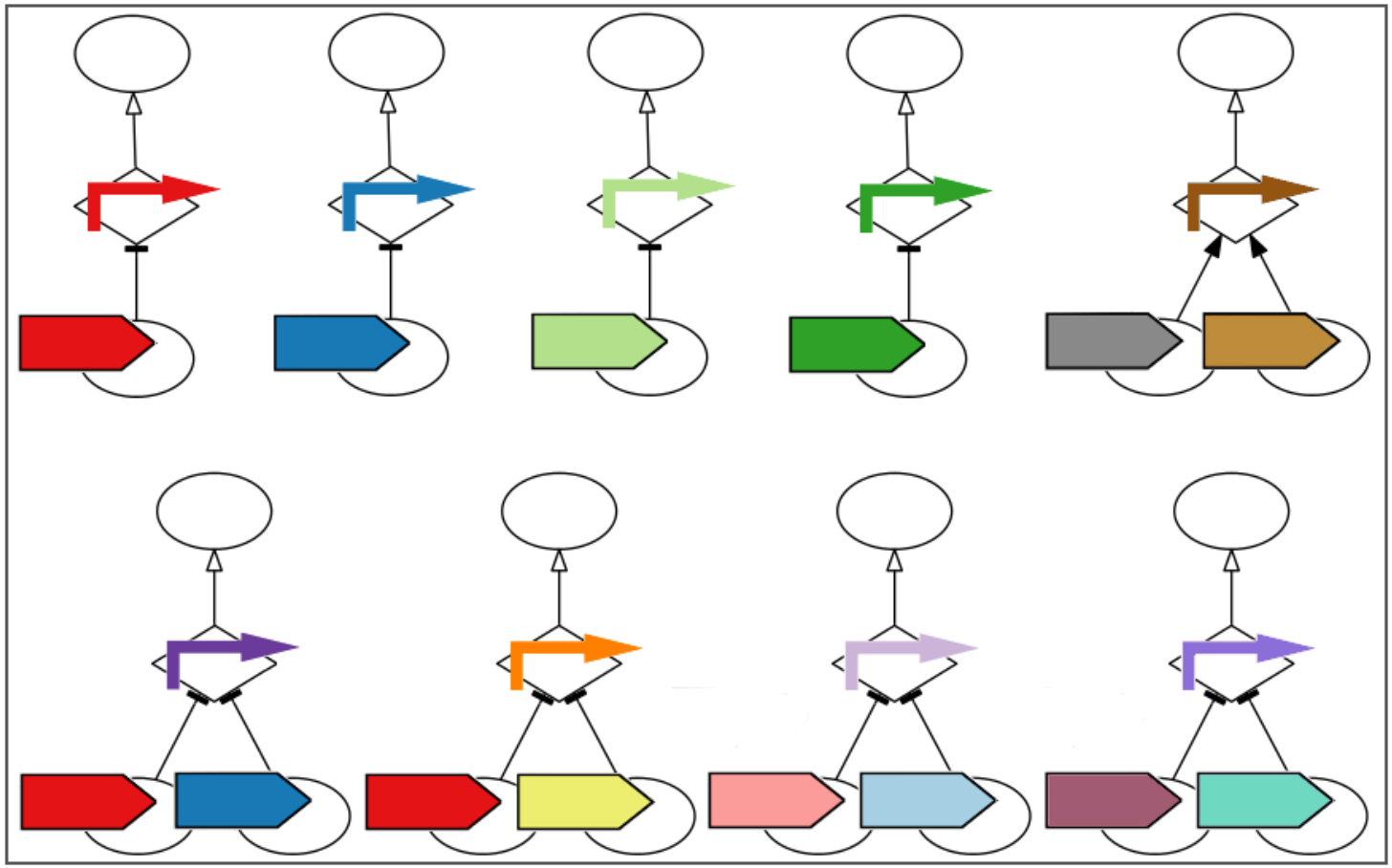
RDF/XML
Annotation



SBOL

Roehner et al., ACS Synthetic Biology (2013).

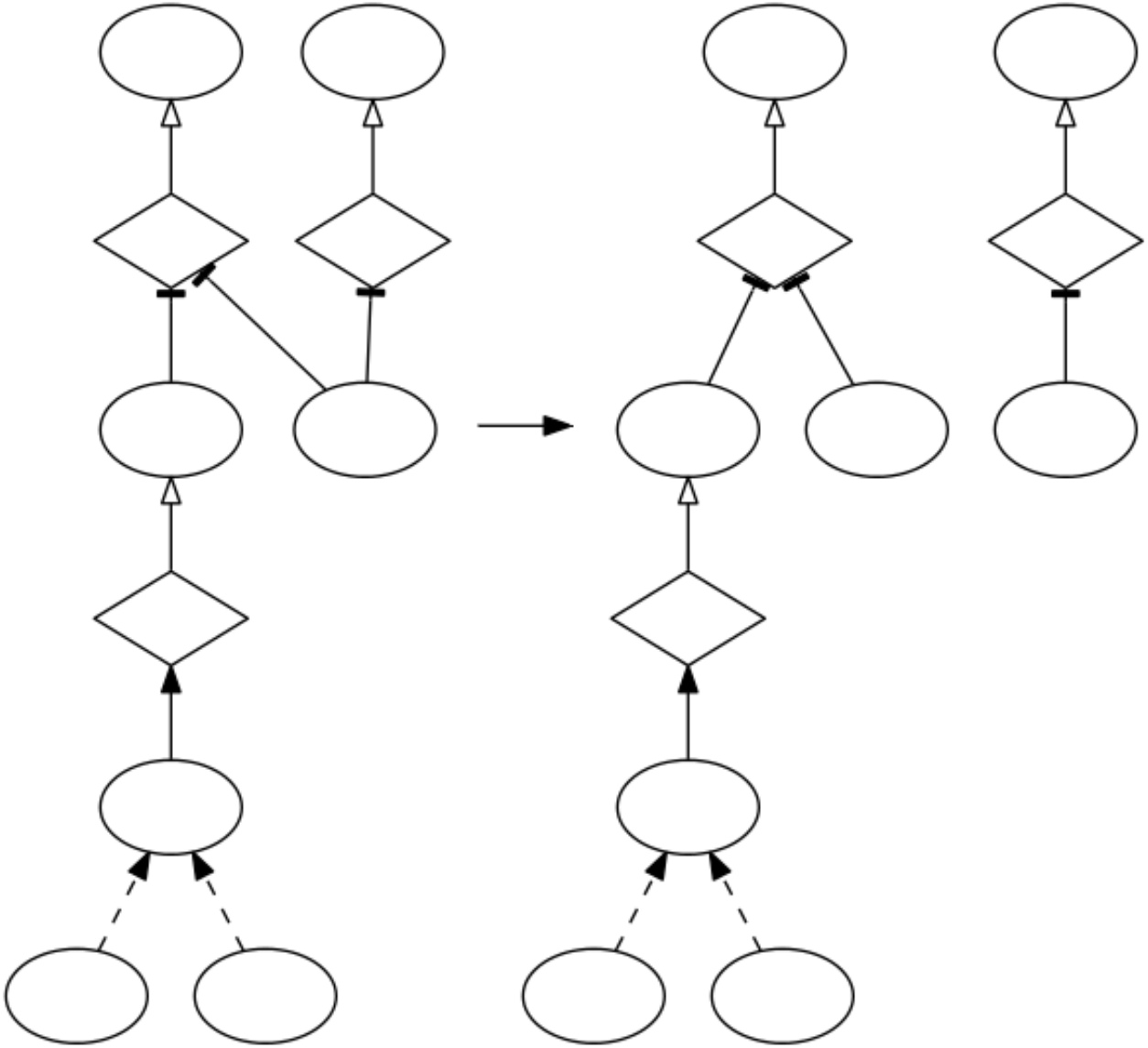
DAG Representation of Library



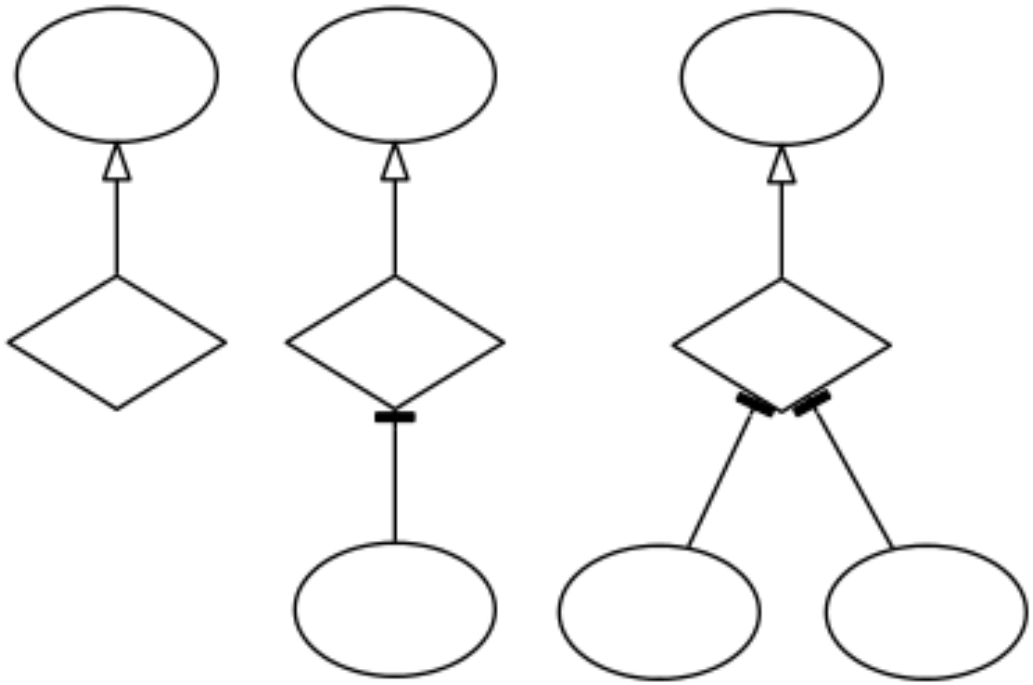
DAG-Based Technology Mapping

1. DAG Representation
2. Partitioning and Decomposition
3. Matching and Covering

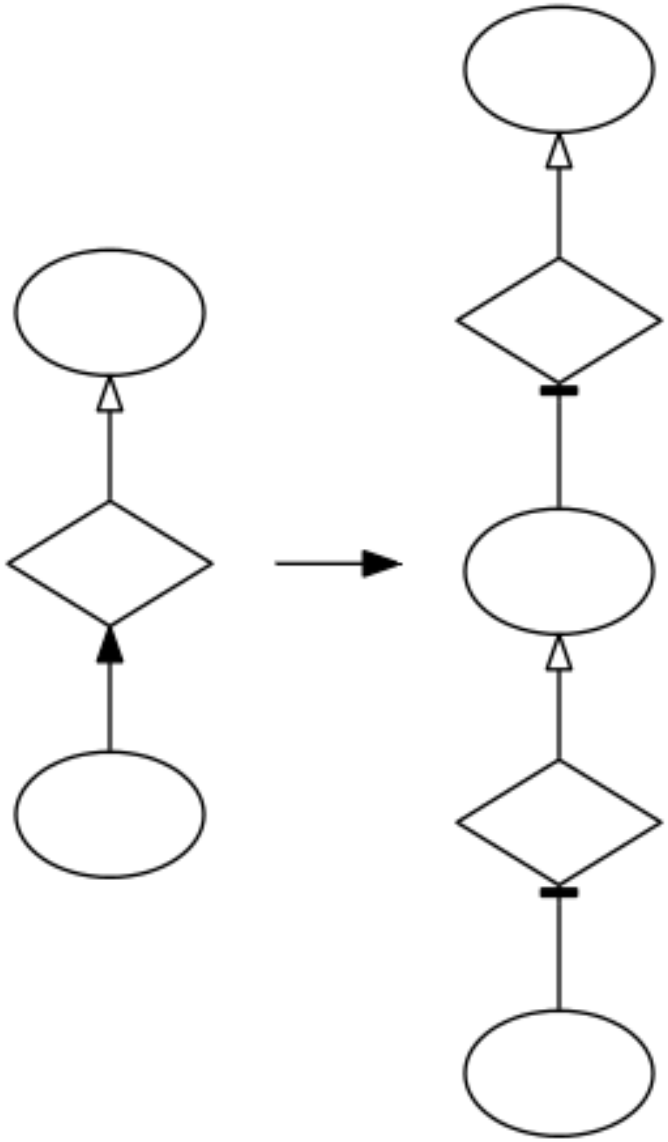
Partitioning of Specification



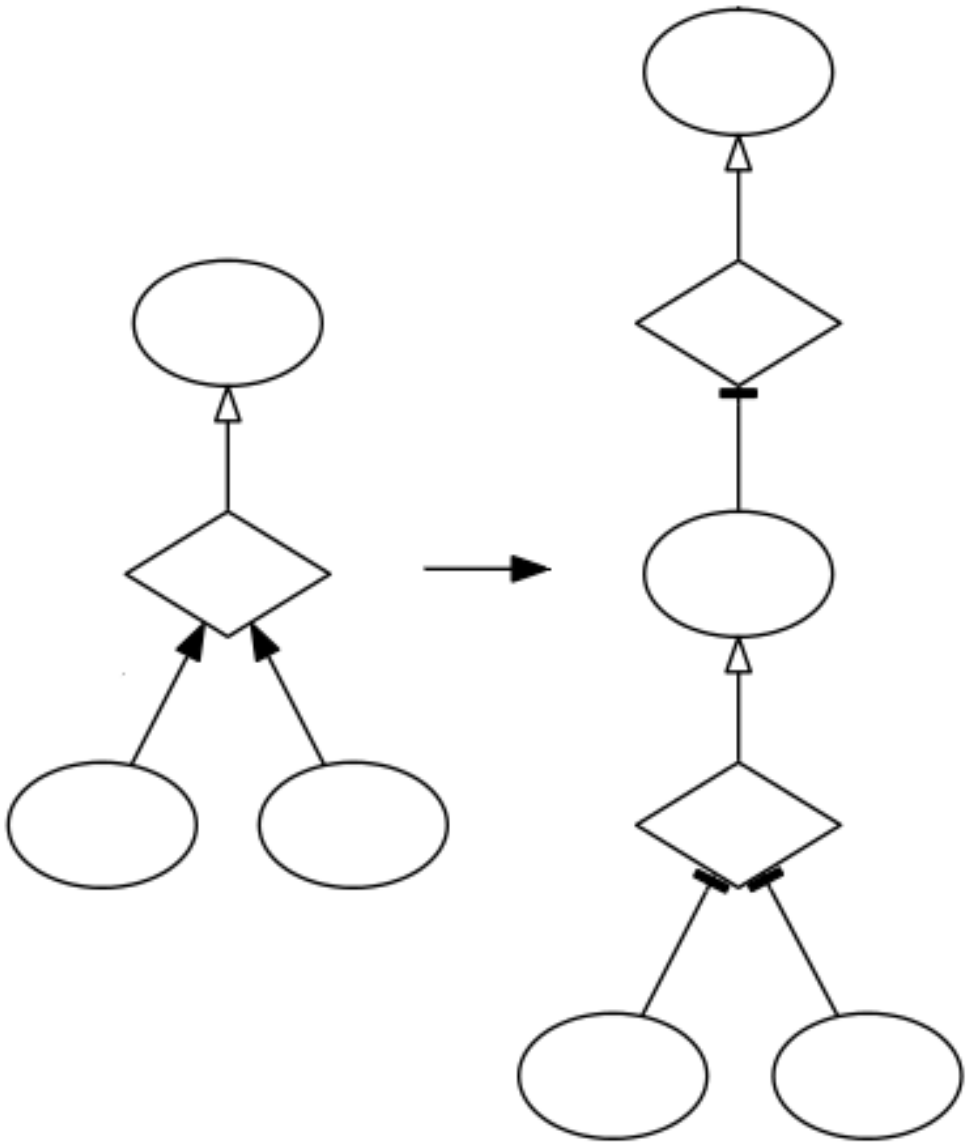
Decomposition into Canonical Form



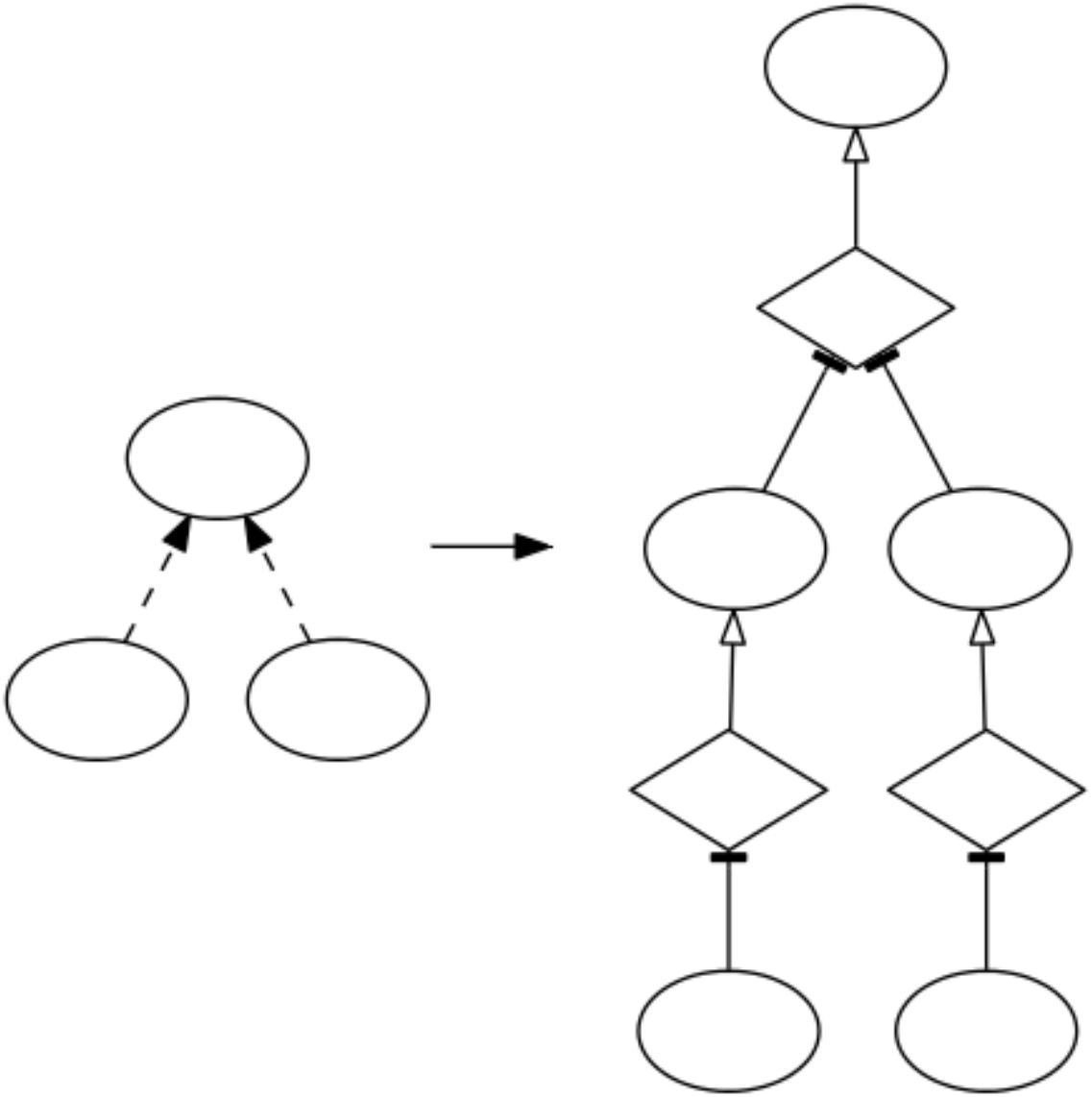
Decomposition of Activation



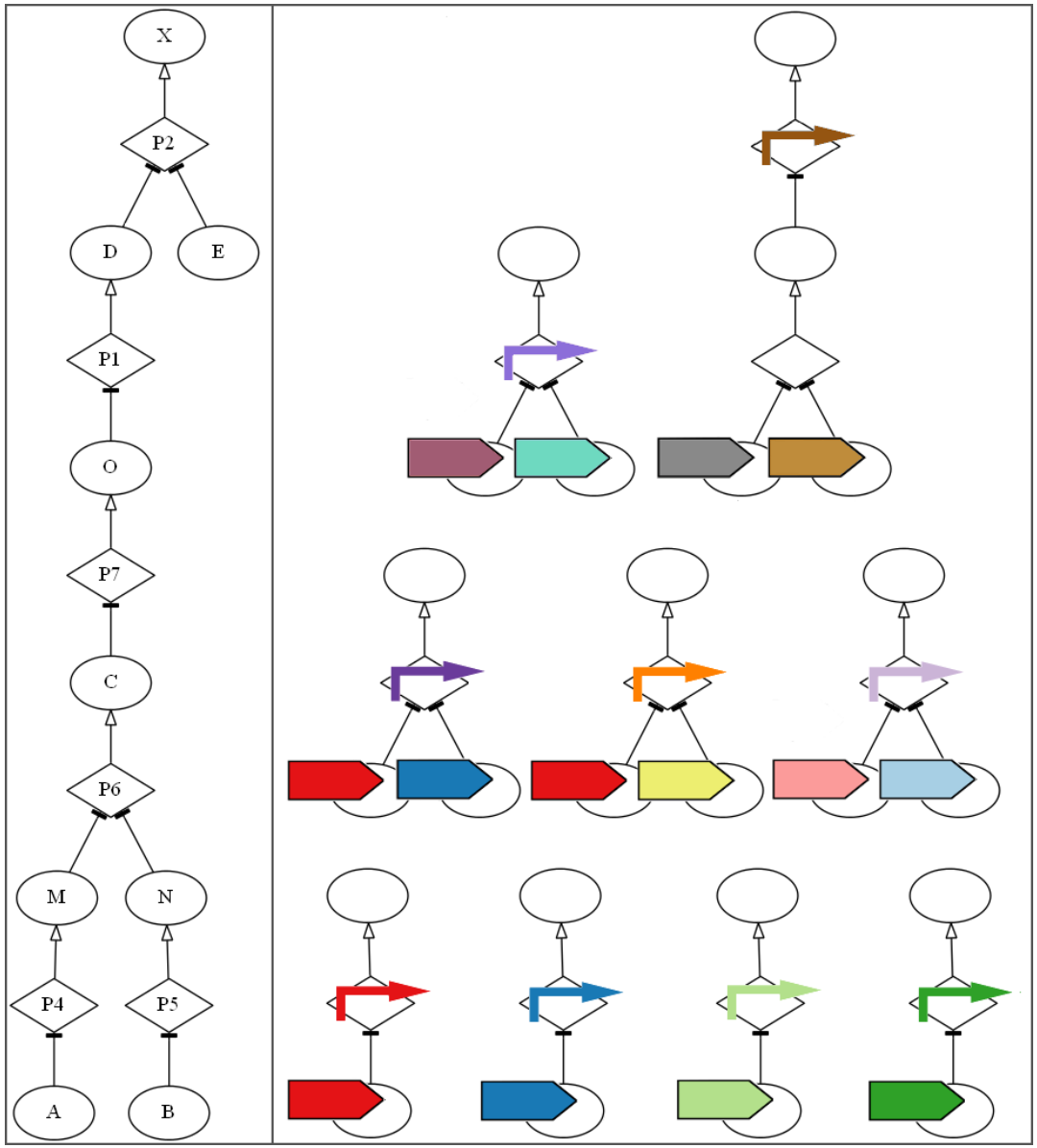
Decomposition of Dual Activation (OR Gate)



Decomposition of Complex Formation (AND Gate)



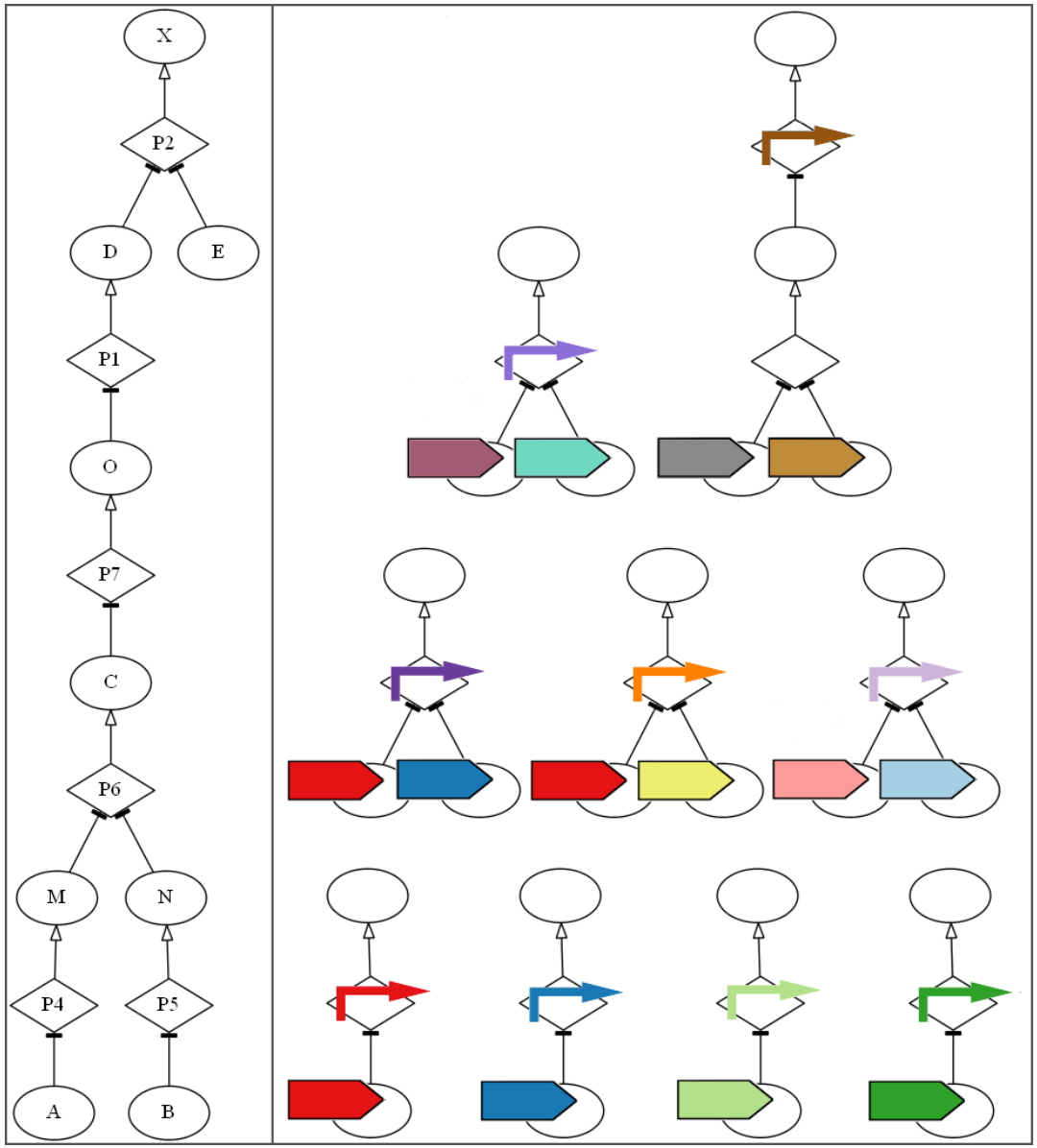
Partitioned, Decomposed Specification and Library



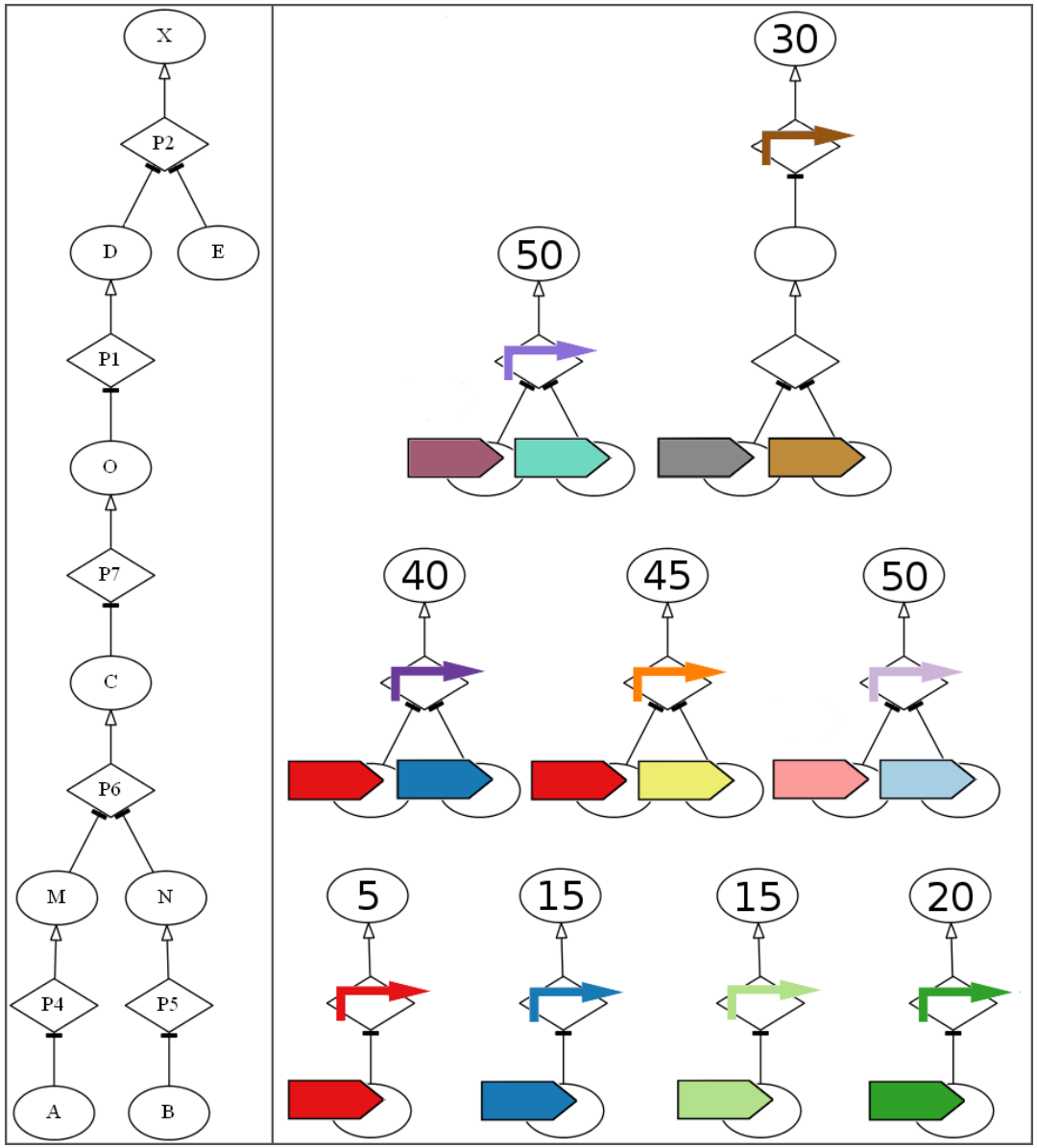
DAG-Based Technology Mapping

1. DAG Representation
2. Partitioning and Decomposition
3. Matching and Covering

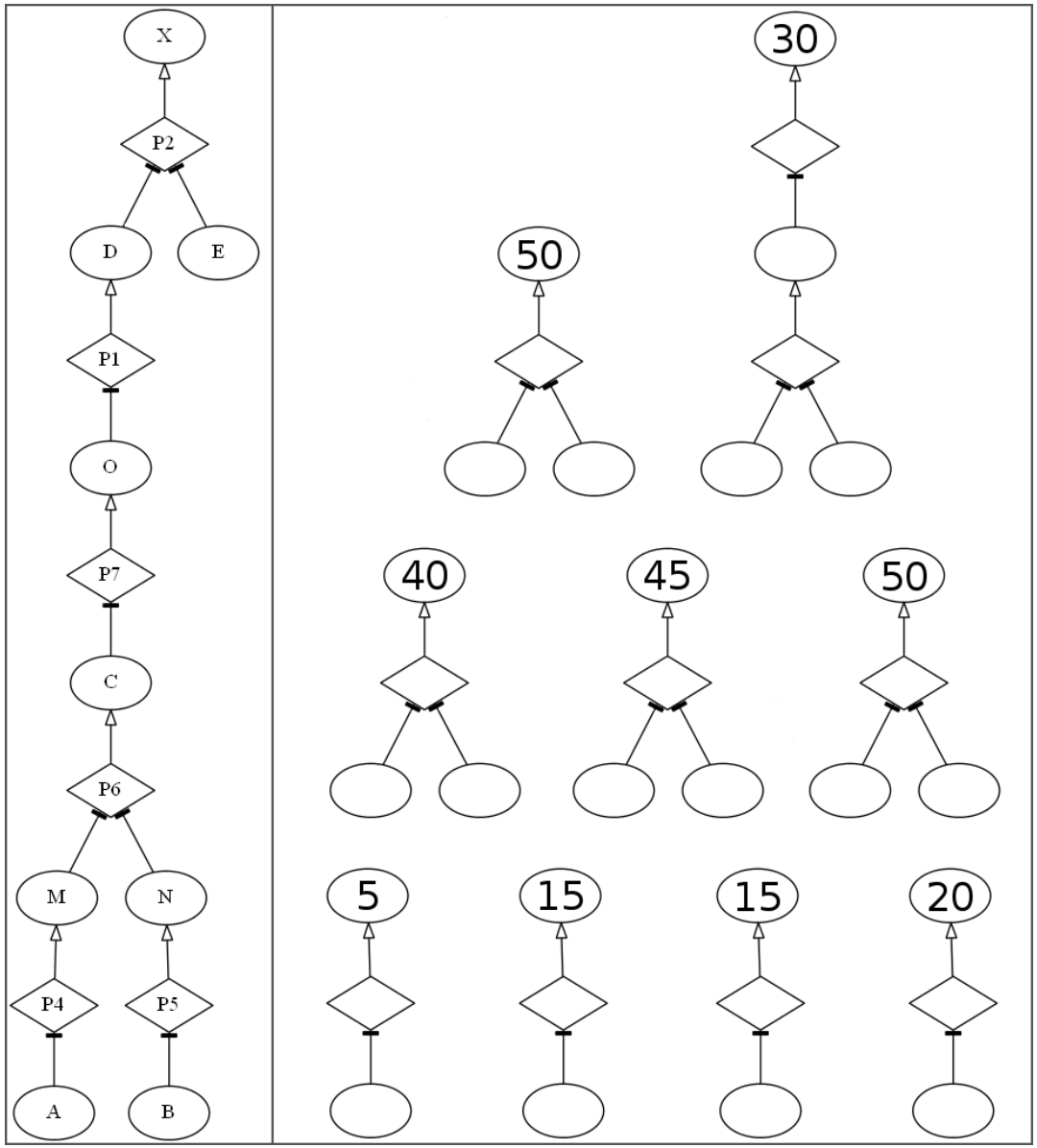
Matching



Matching

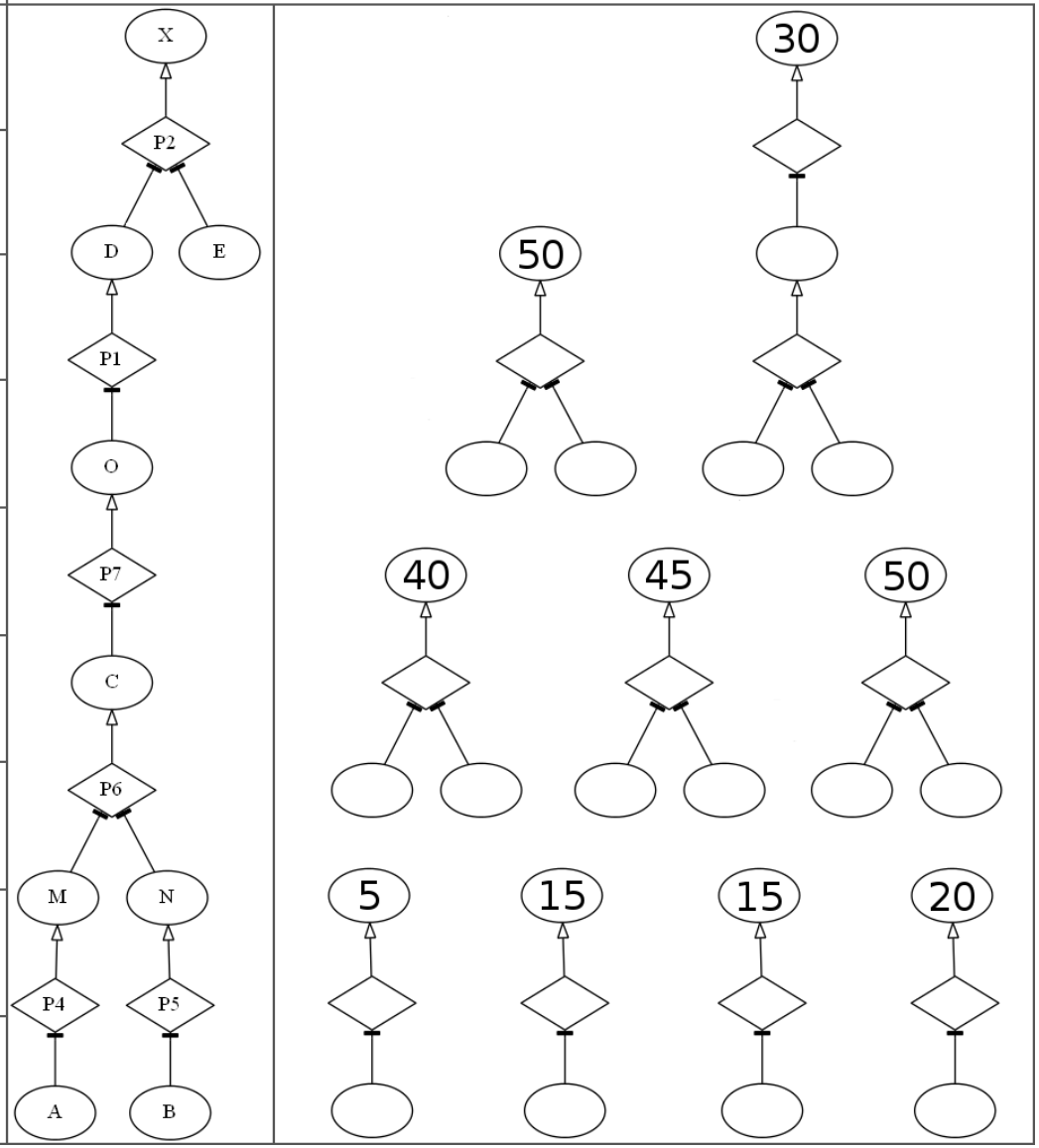


Matching



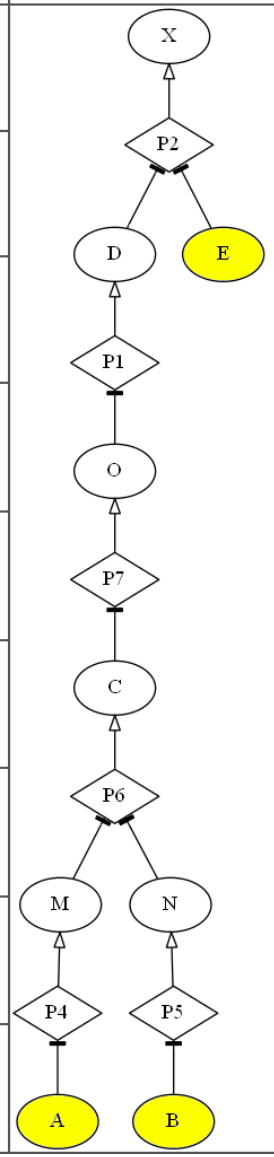
Matching

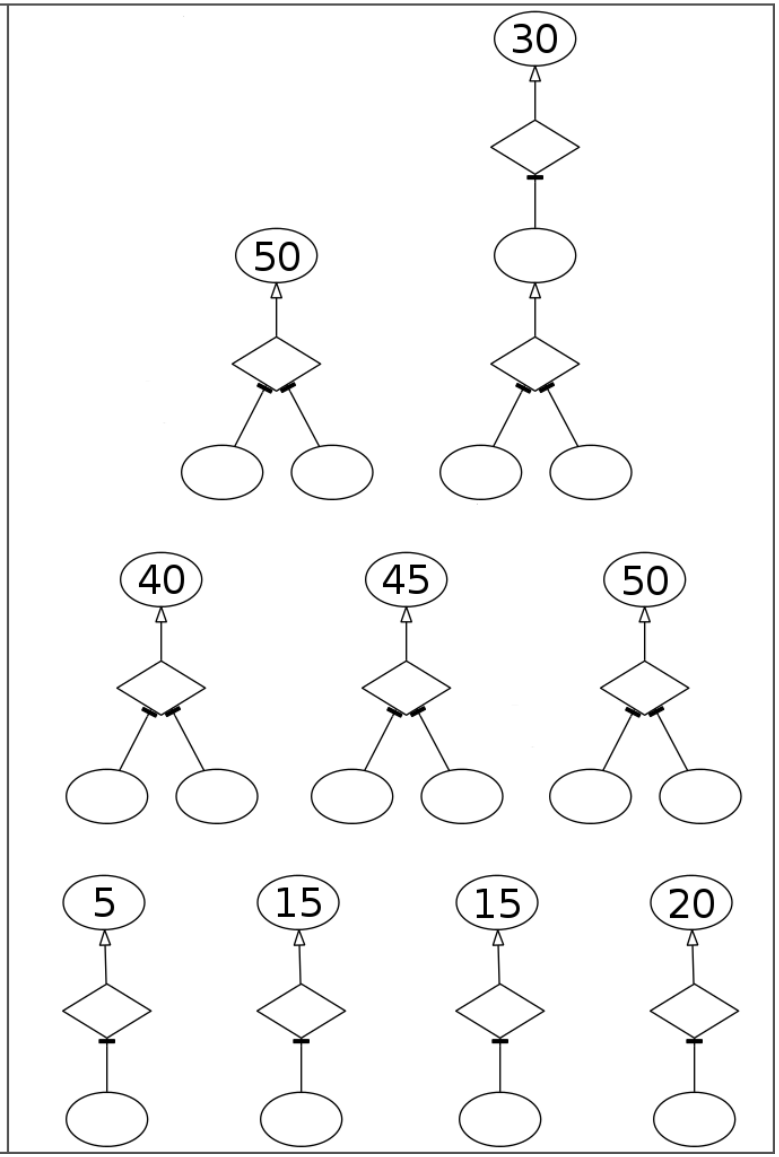
Node	Bound
X	
D	
O	
C	
M	
N	
A	
B	
E	



Matching

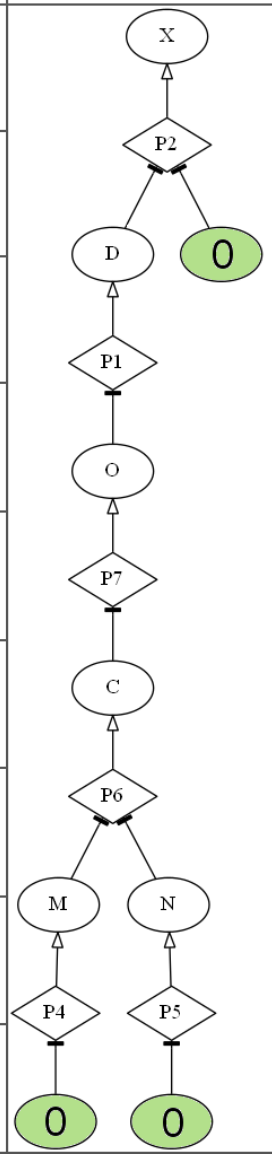
Node	Bound
X	
D	
O	
C	
M	
N	
A	
B	
E	

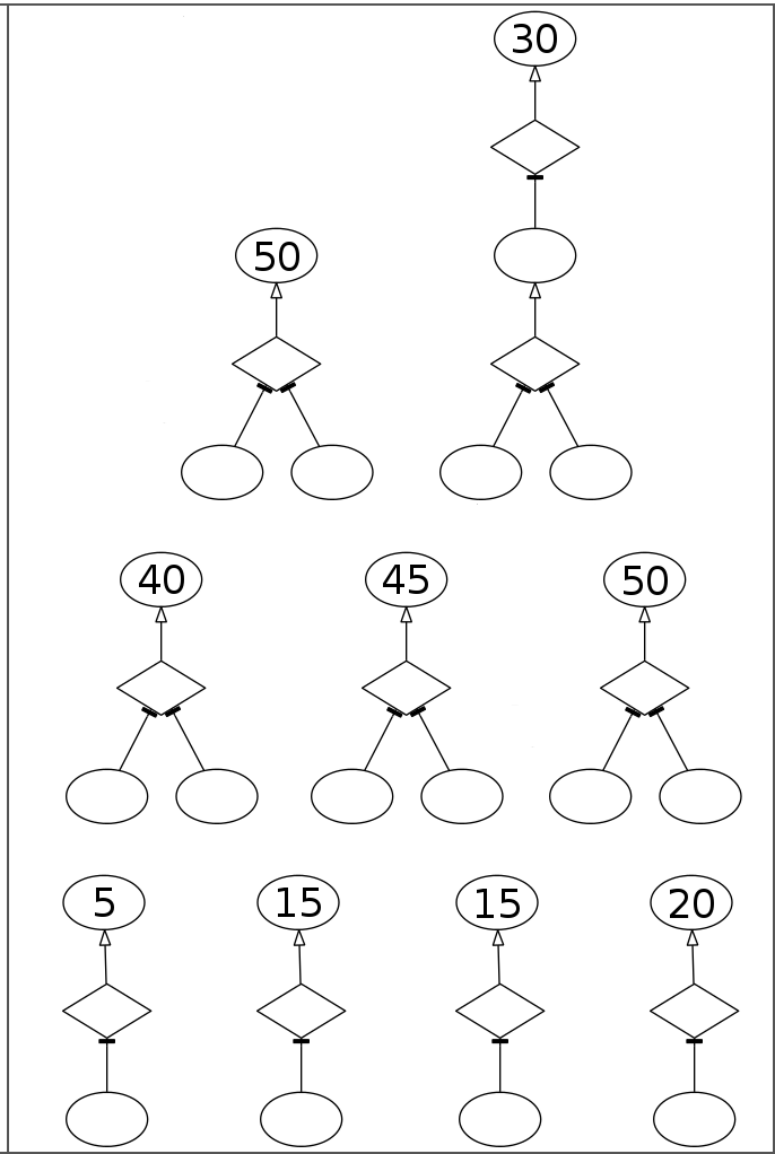




Matching

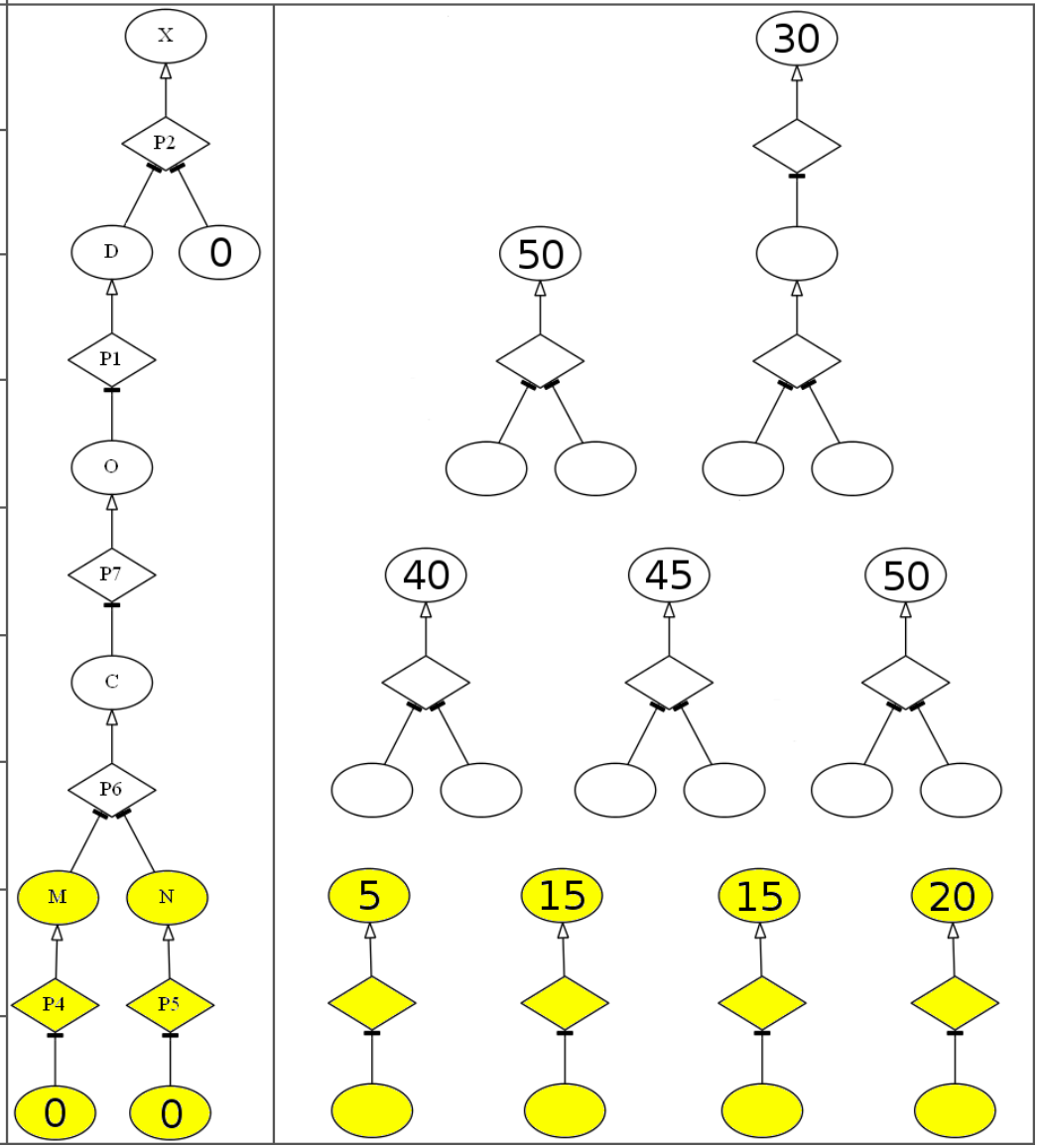
Node	Bound
X	
D	
O	
C	
M	
N	
A	0
B	0
E	0





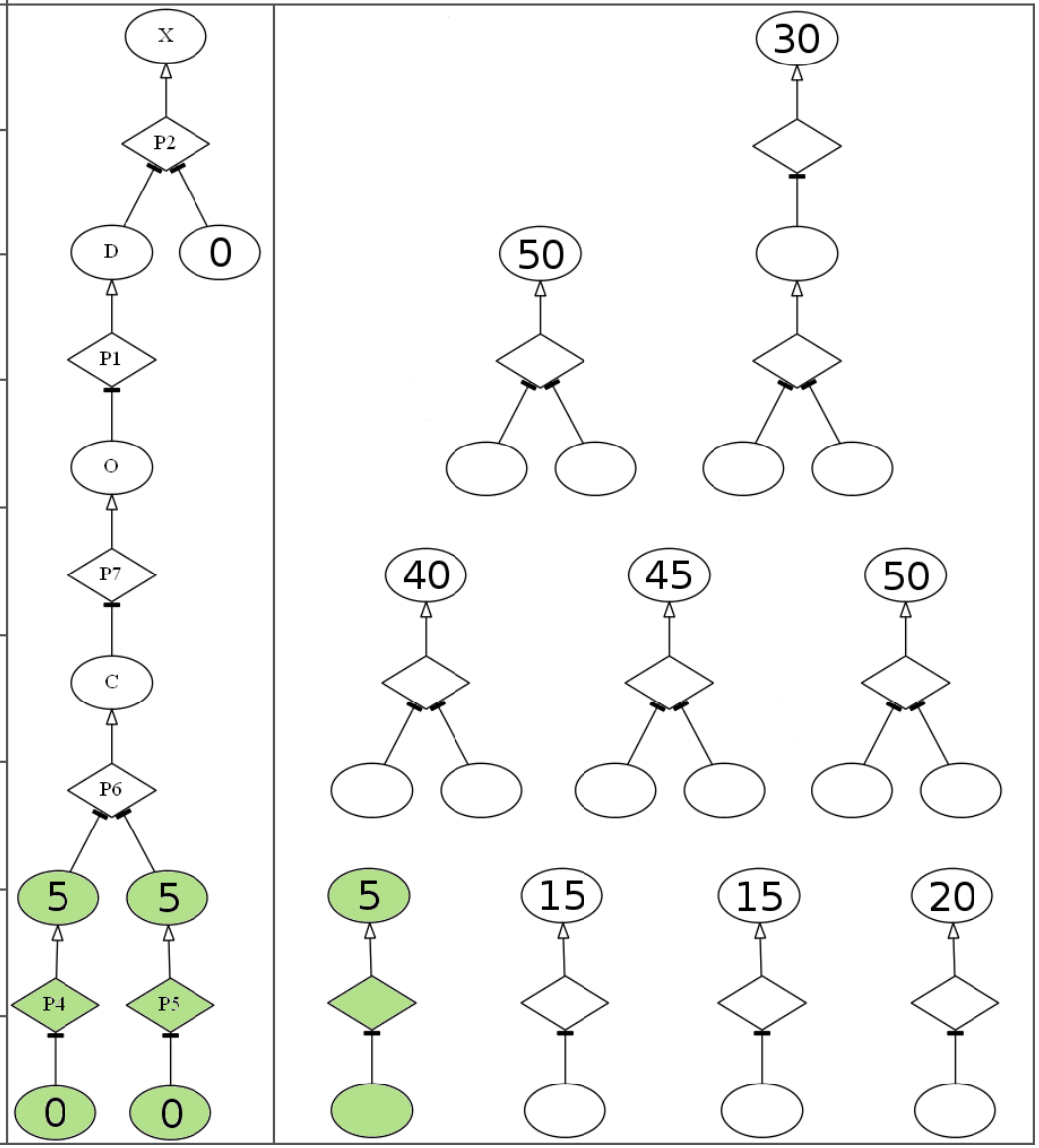
Matching

Node	Bound
X	
D	
O	
C	
M	
N	
A	0
B	0
E	0



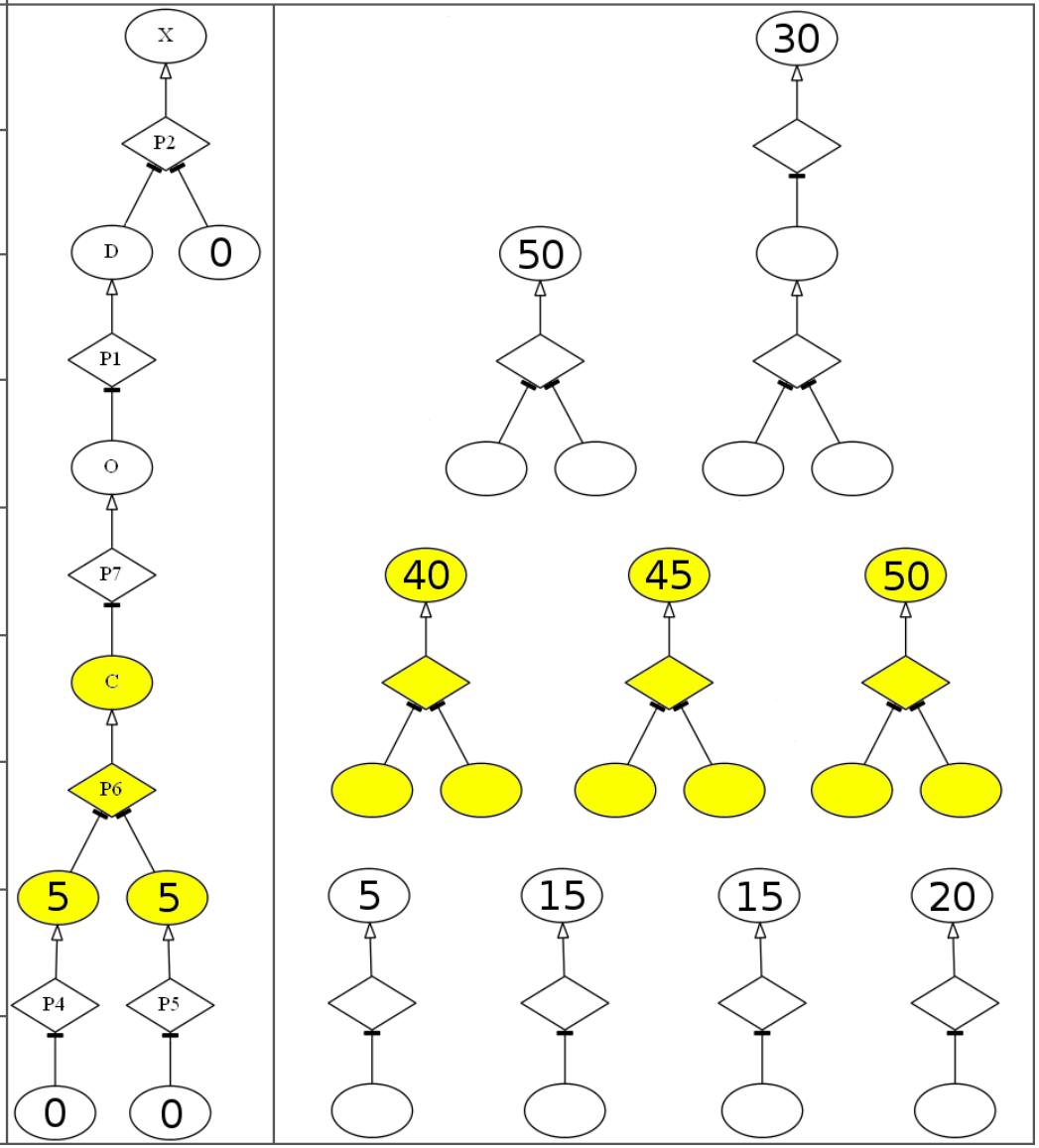
Matching

Node	Bound
X	
D	
O	
C	
M	5
N	5
A	0
B	0
E	0



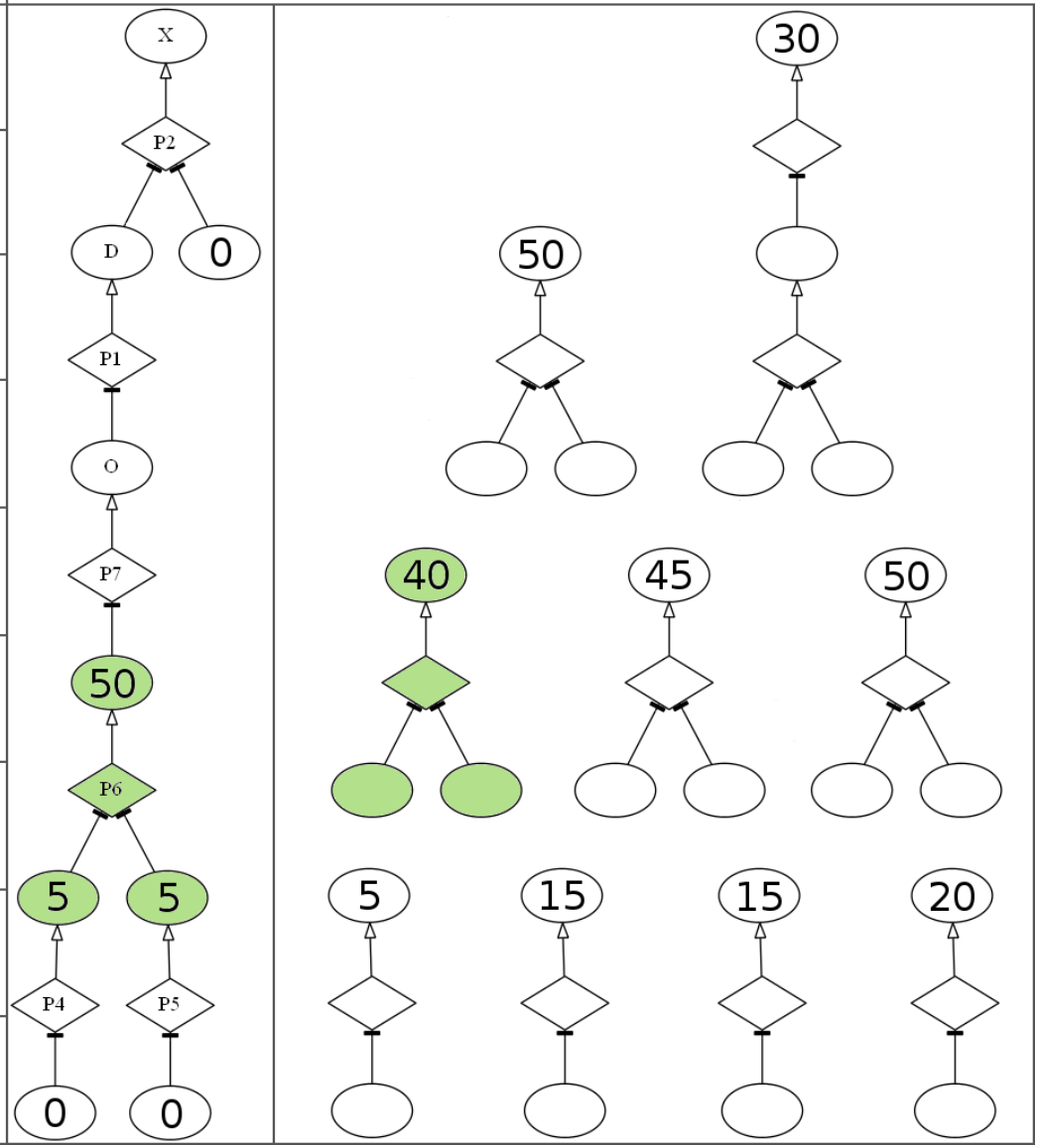
Matching

Node	Bound
X	
D	
O	
C	
M	5
N	5
A	0
B	0
E	0



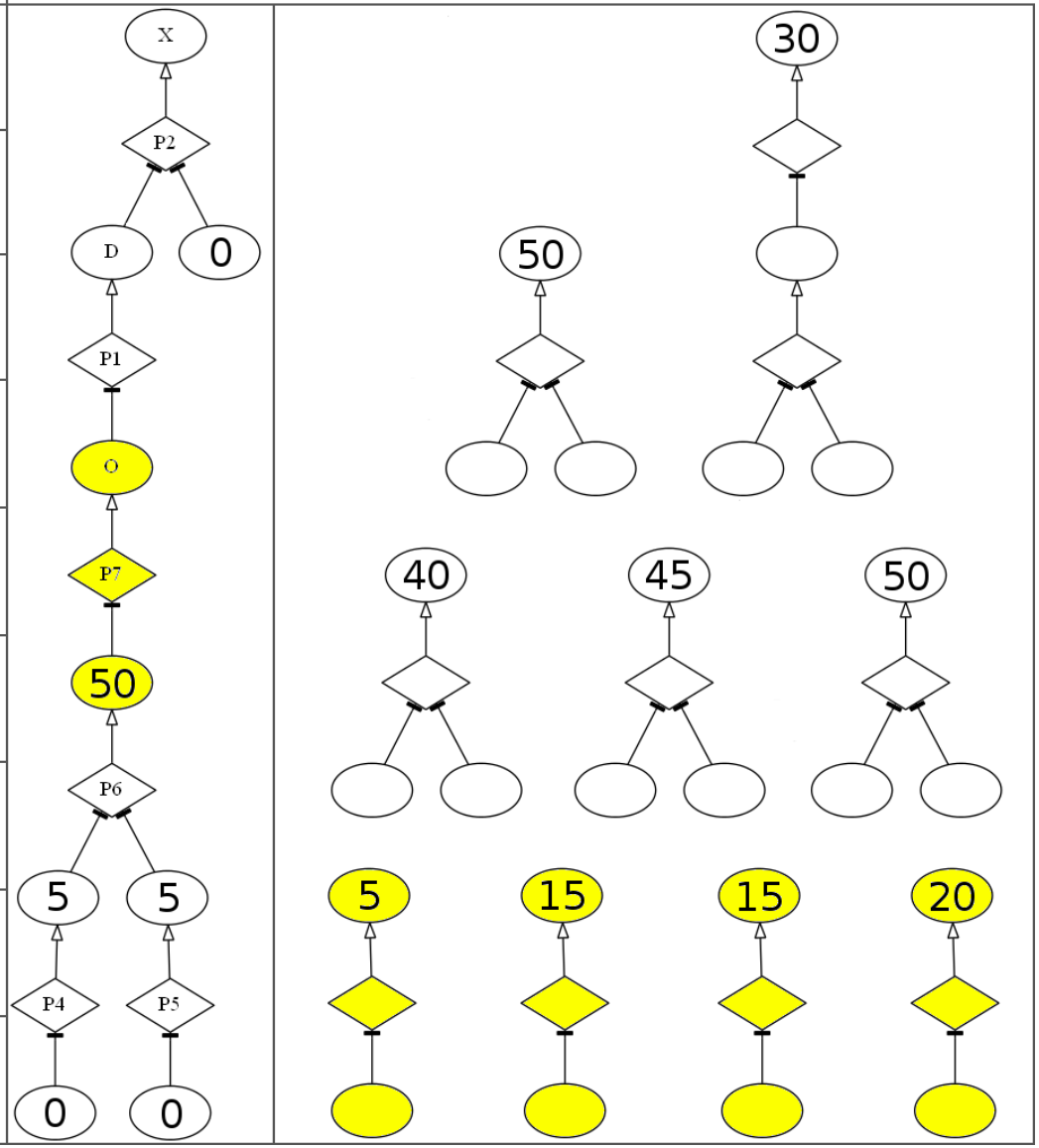
Matching

Node	Bound
X	
D	
O	
C	50
M	5
N	5
A	0
B	0
E	0



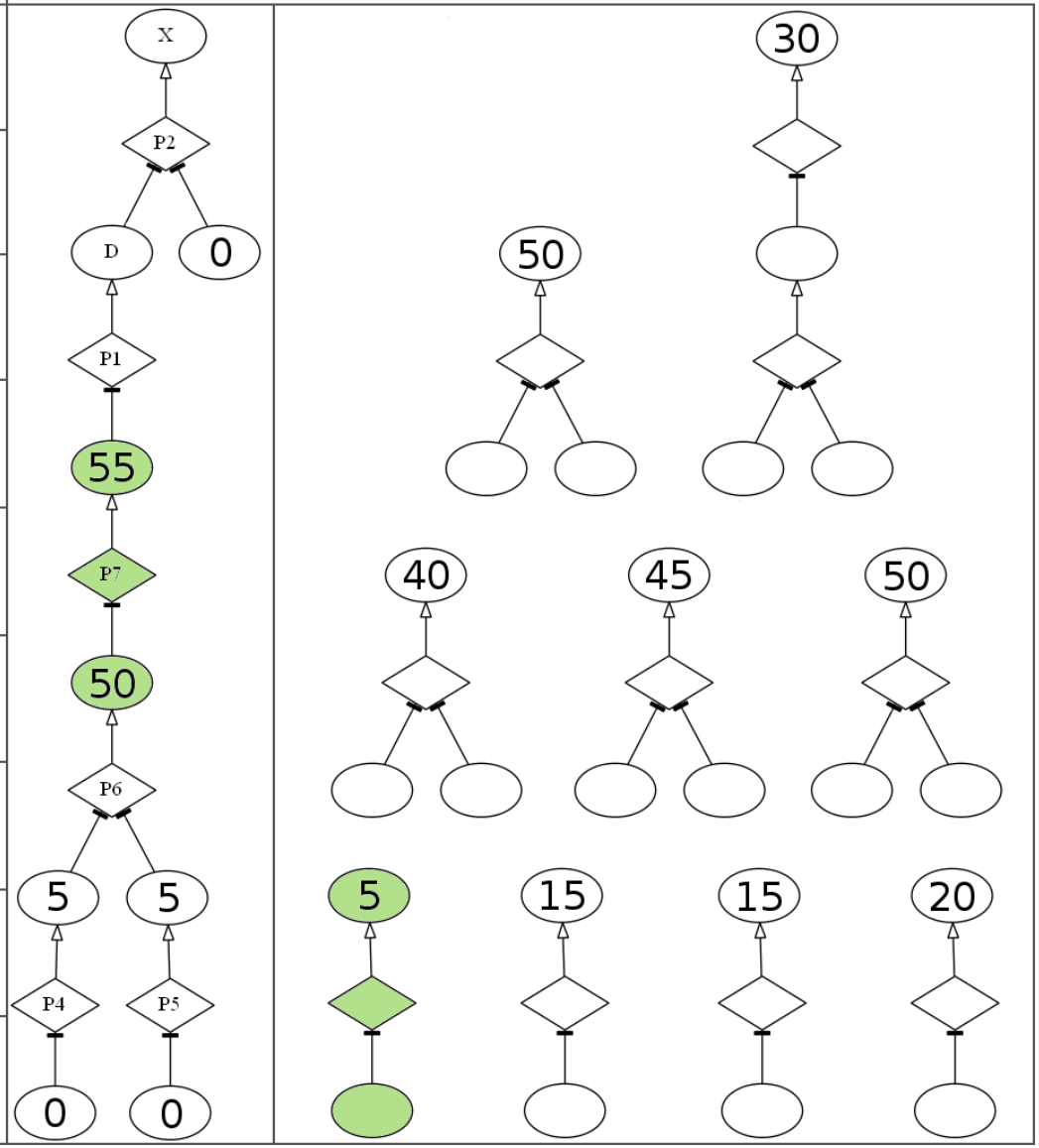
Matching

Node	Bound
X	
D	
O	
C	50
M	5
N	5
A	0
B	0
E	0



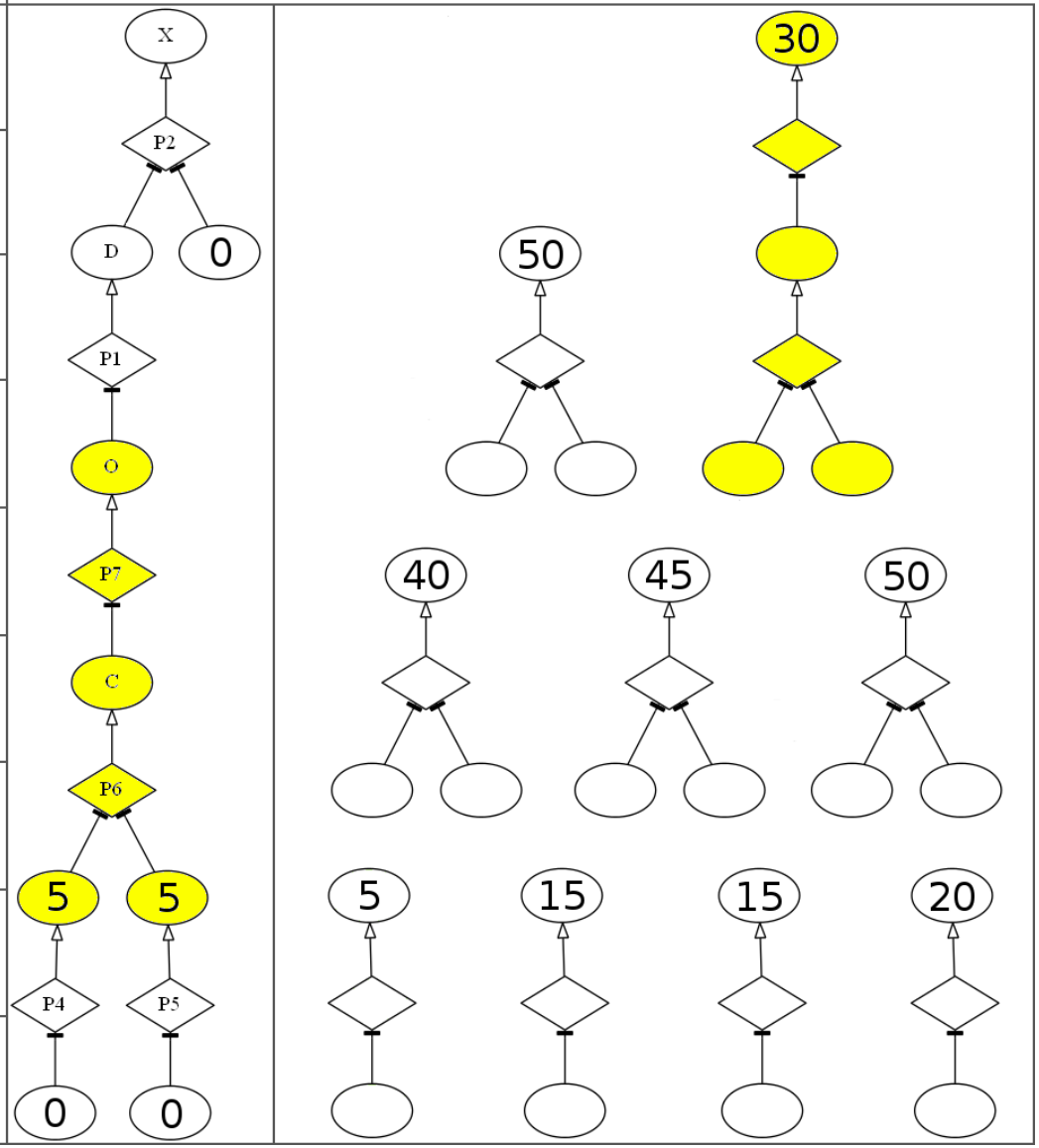
Matching

Node	Bound
X	
D	
O	55
C	50
M	5
N	5
A	0
B	0
E	0



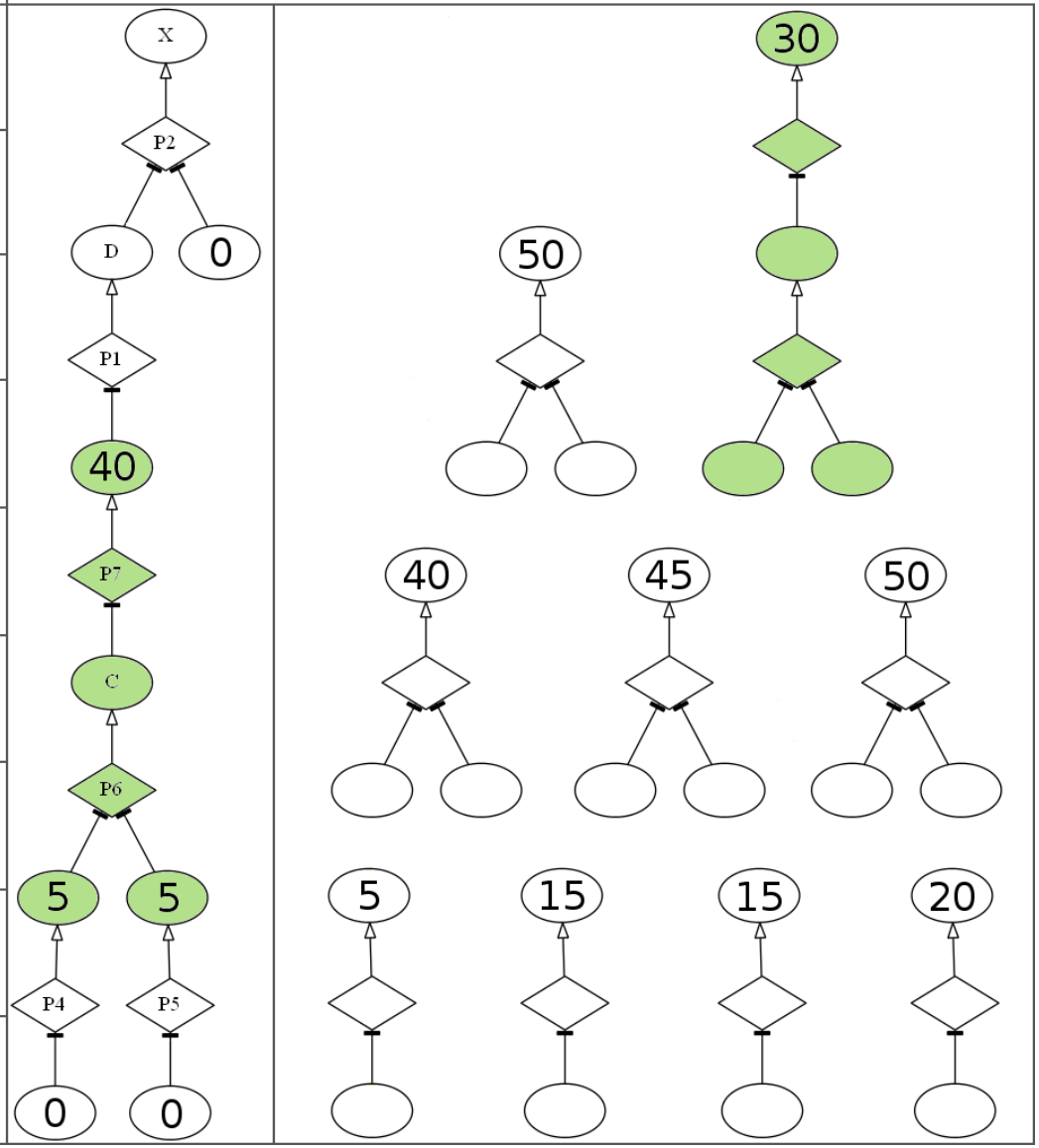
Matching

Node	Bound
X	
D	
O	55
C	50
M	5
N	5
A	0
B	0
E	0



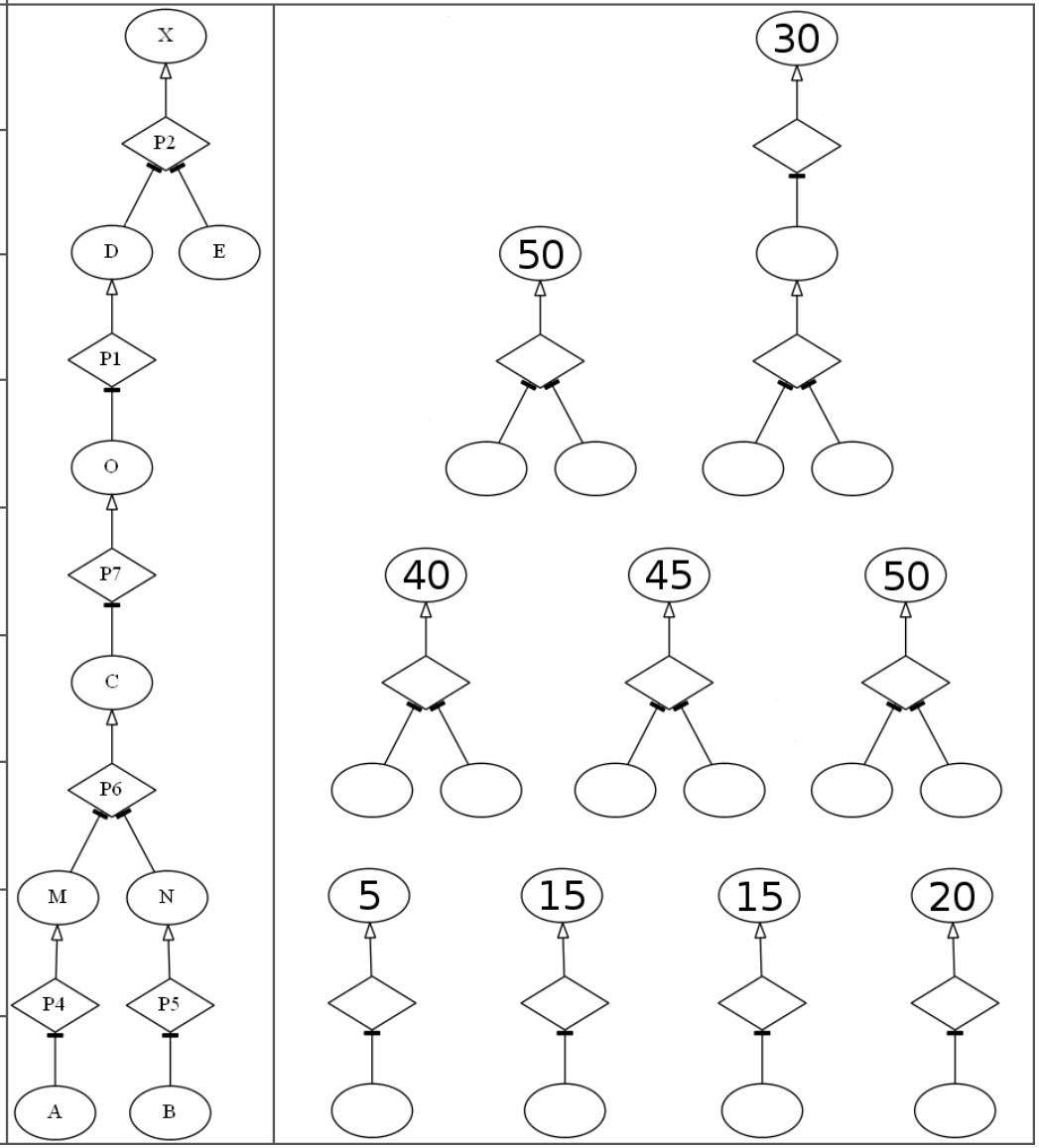
Matching

Node	Bound
X	
D	
O	40
C	50
M	5
N	5
A	0
B	0
E	0



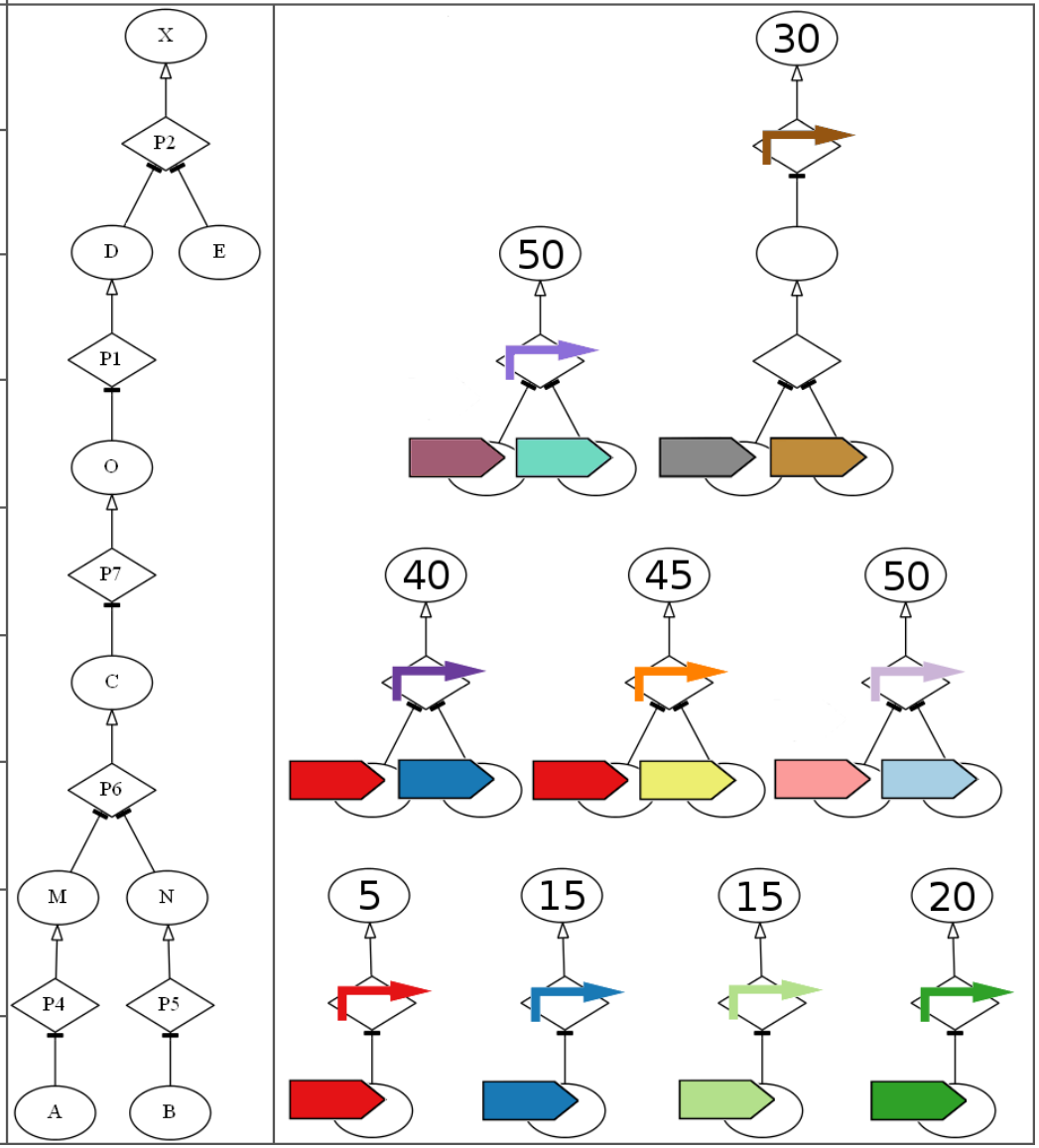
Matching

Node	Bound
X	85
D	45
O	40
C	50
M	5
N	5
A	0
B	0
E	0



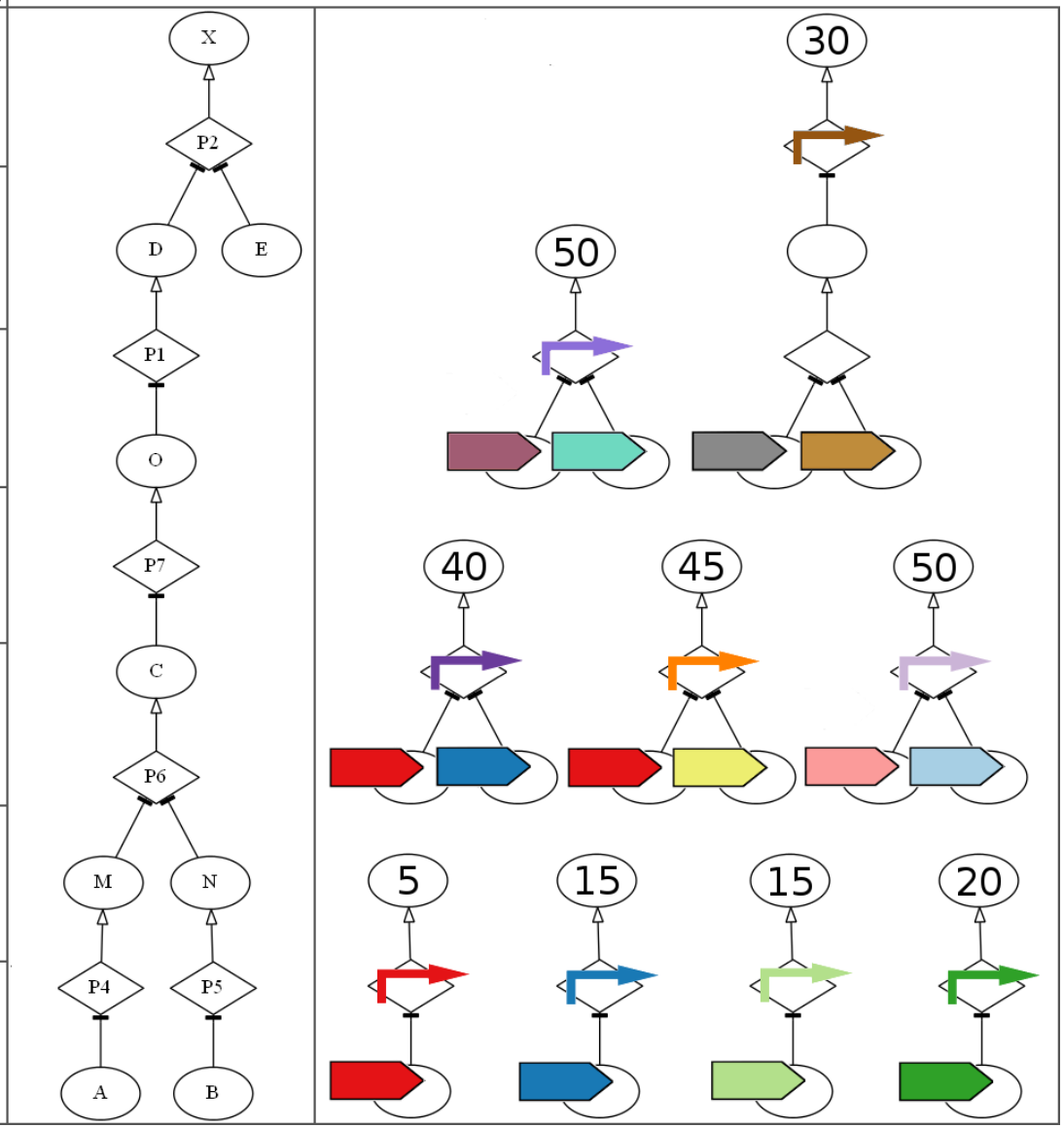
Covering

Node	Bound
X	85
D	45
O	40
C	50
M	5
N	5
A	0
B	0
E	0



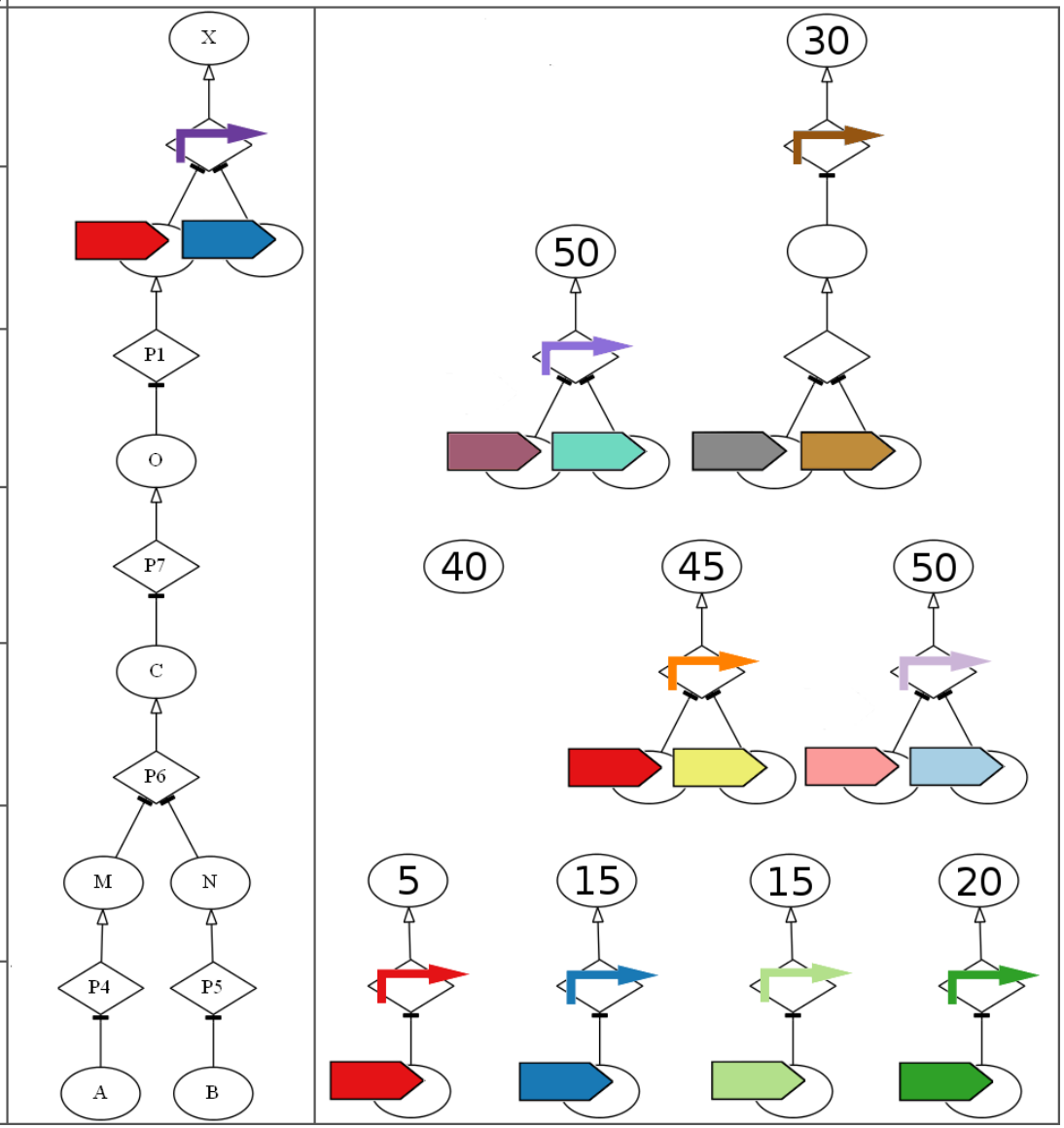
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E		45,0	
O		40	
C		50	
M,N		5,5	
A,N		0,5	
B		0	



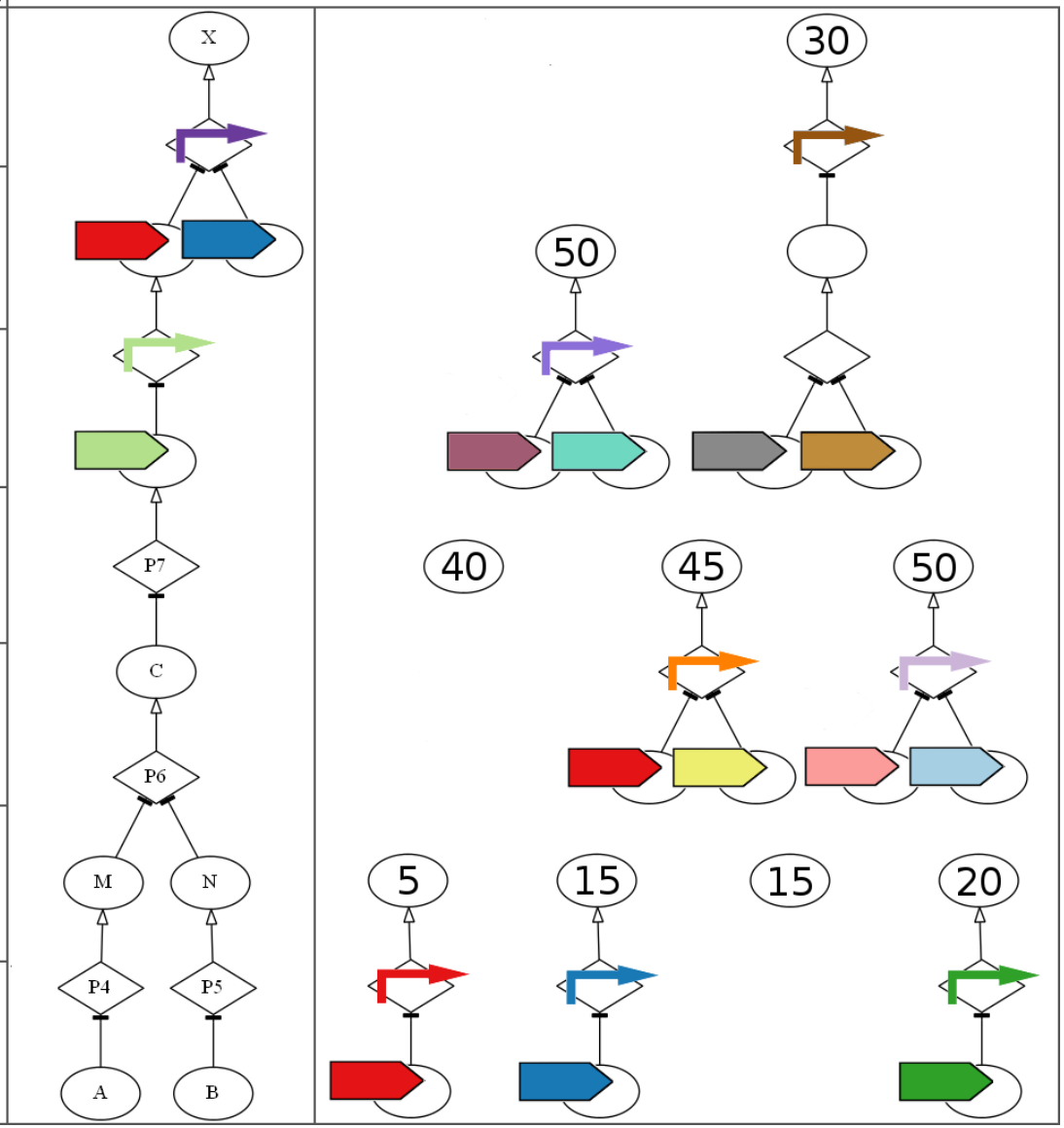
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	40	45,0	85
O		40	
C		50	
M,N		5,5	
A,N		0,5	
B		0	



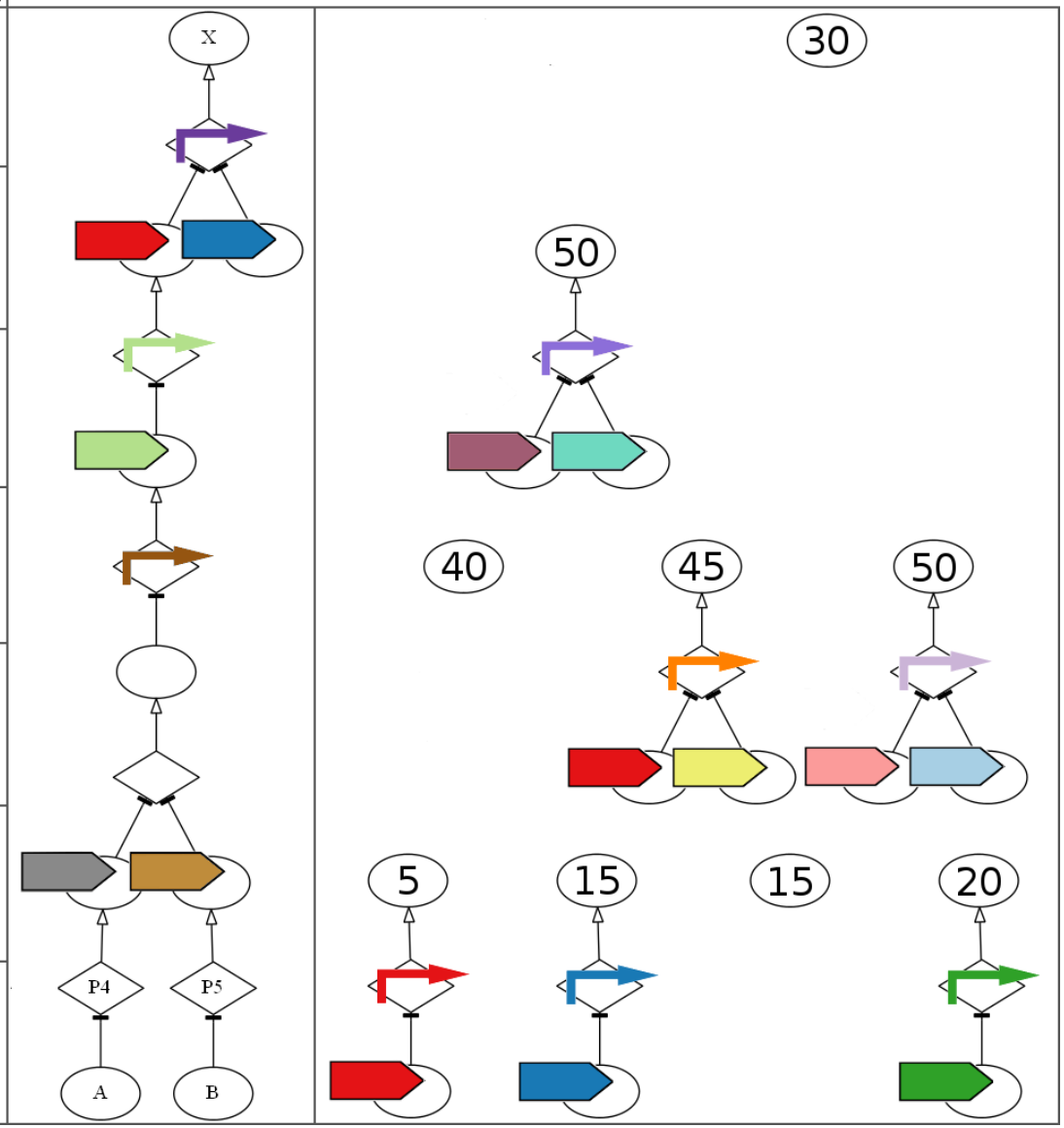
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	40	45,0	85
O	55	40	95
C		50	
M,N		5,5	
A,N		0,5	
B		0	



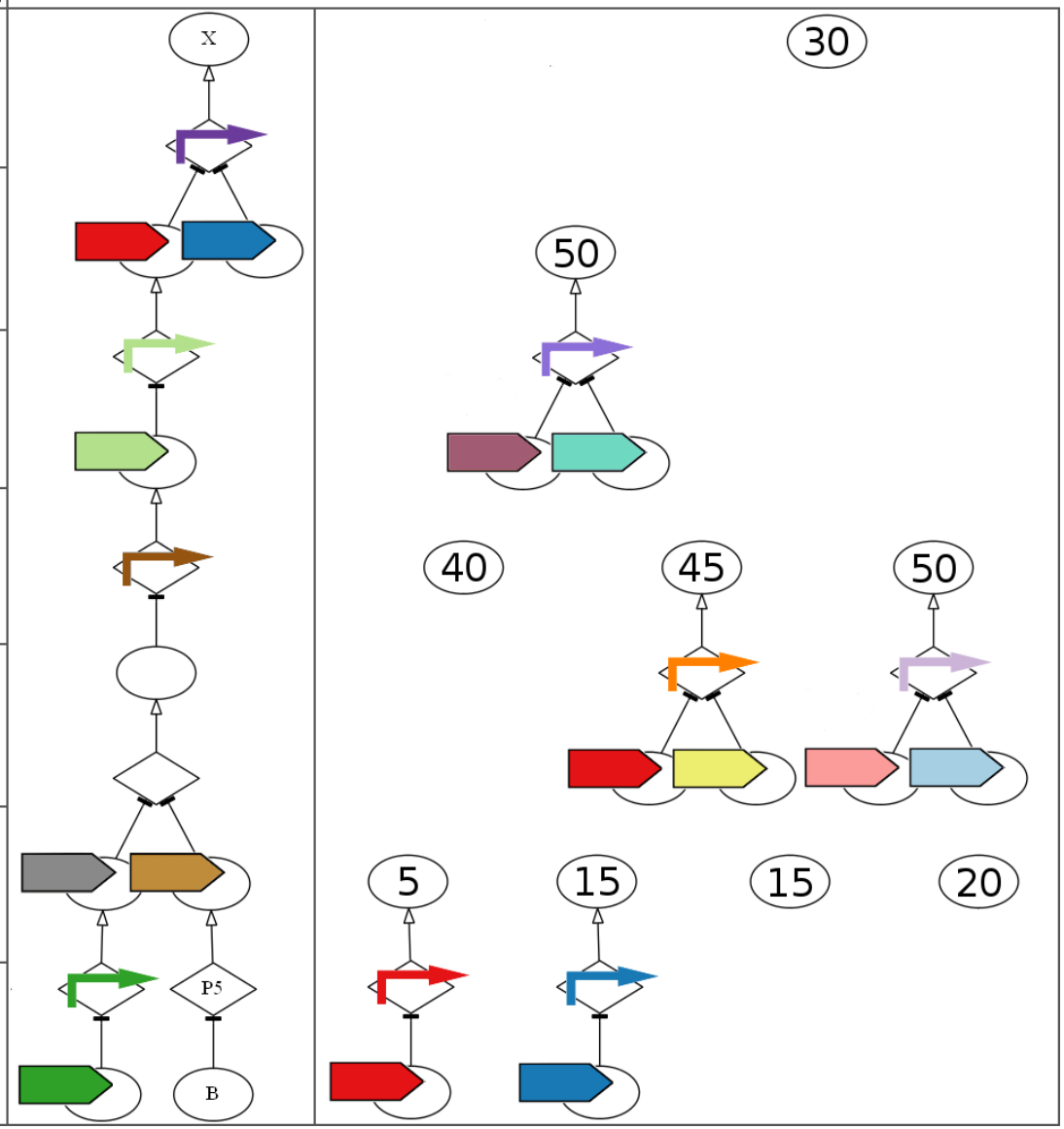
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	40	45,0	85
O	55	40	95
C	–	50	–
M,N	85	5,5	95
A,N		0,5	
B		0	



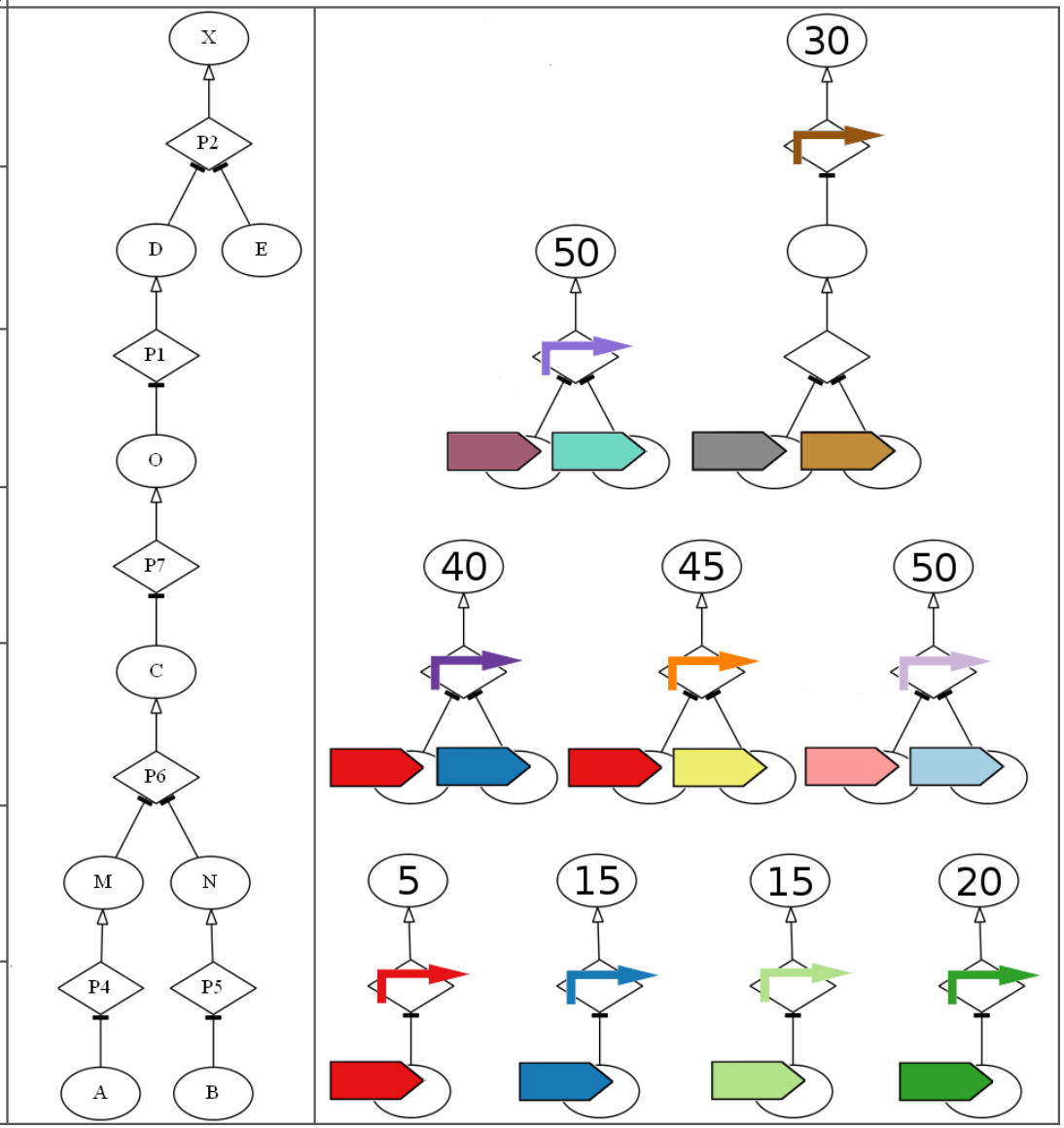
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	40	45,0	85
O	55	40	95
C	–	50	–
M,N	85	5,5	95
A,N	105	0,5	110
B		0	



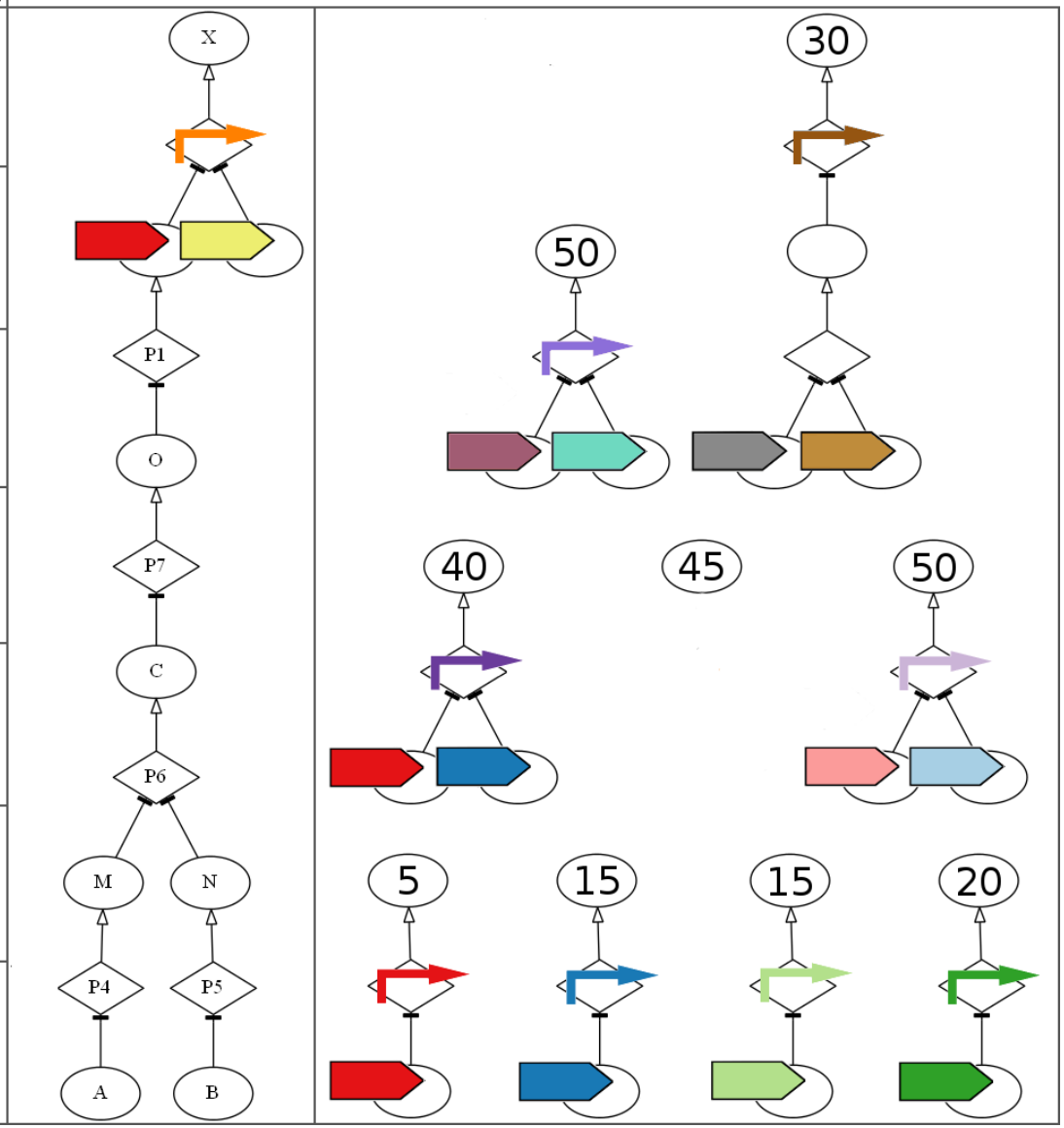
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E		45,0	
O		40	
C		50	
M,N		5,5	
A,N		0,5	
B		0	



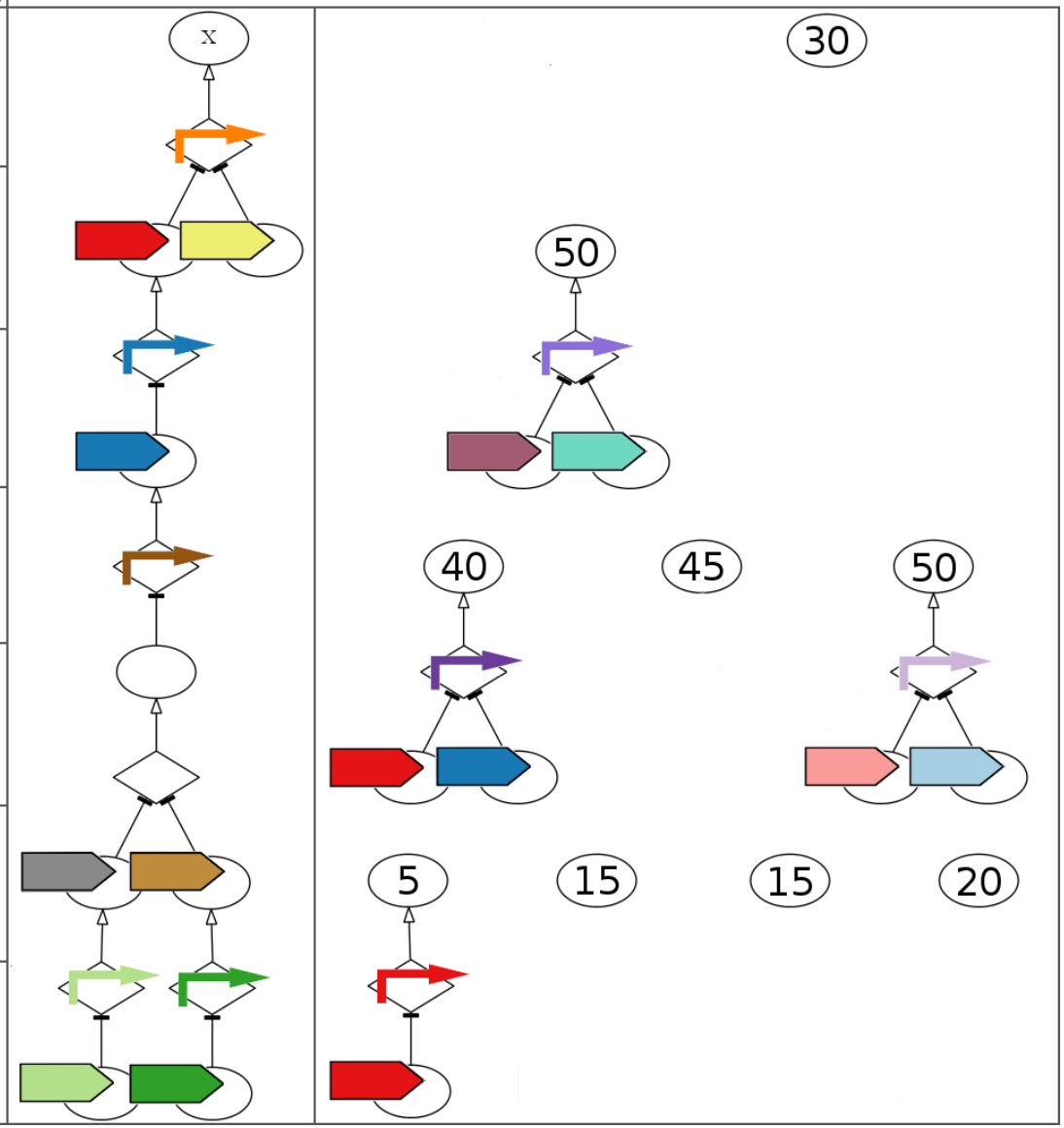
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	45	45,0	90
O		40	
C		50	
M,N		5,5	
A,N		0,5	
B		0	



Covering

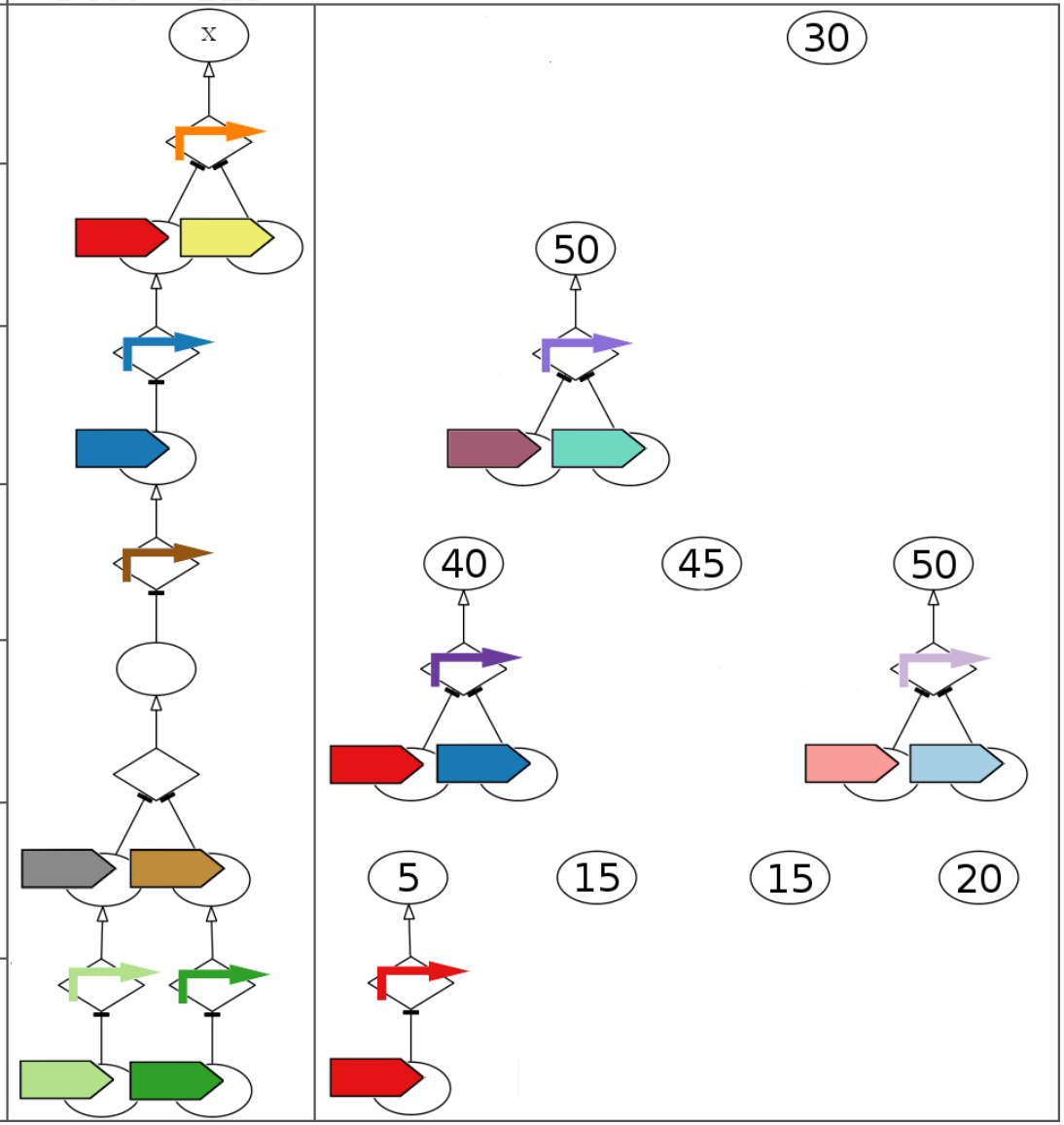
Node	Current	Bound	Estimate
X	0	85	85
D,E	45	45,0	90
O	60	40	100
C	–	50	–
M,N	90	5,5	100
A,N	105	0,5	110
B	125	0	125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	45	45,0	90
O	60	40	100
C	–	50	–
M,N	90	5,5	100
A,N	105	0,5	110
B	125	0	125

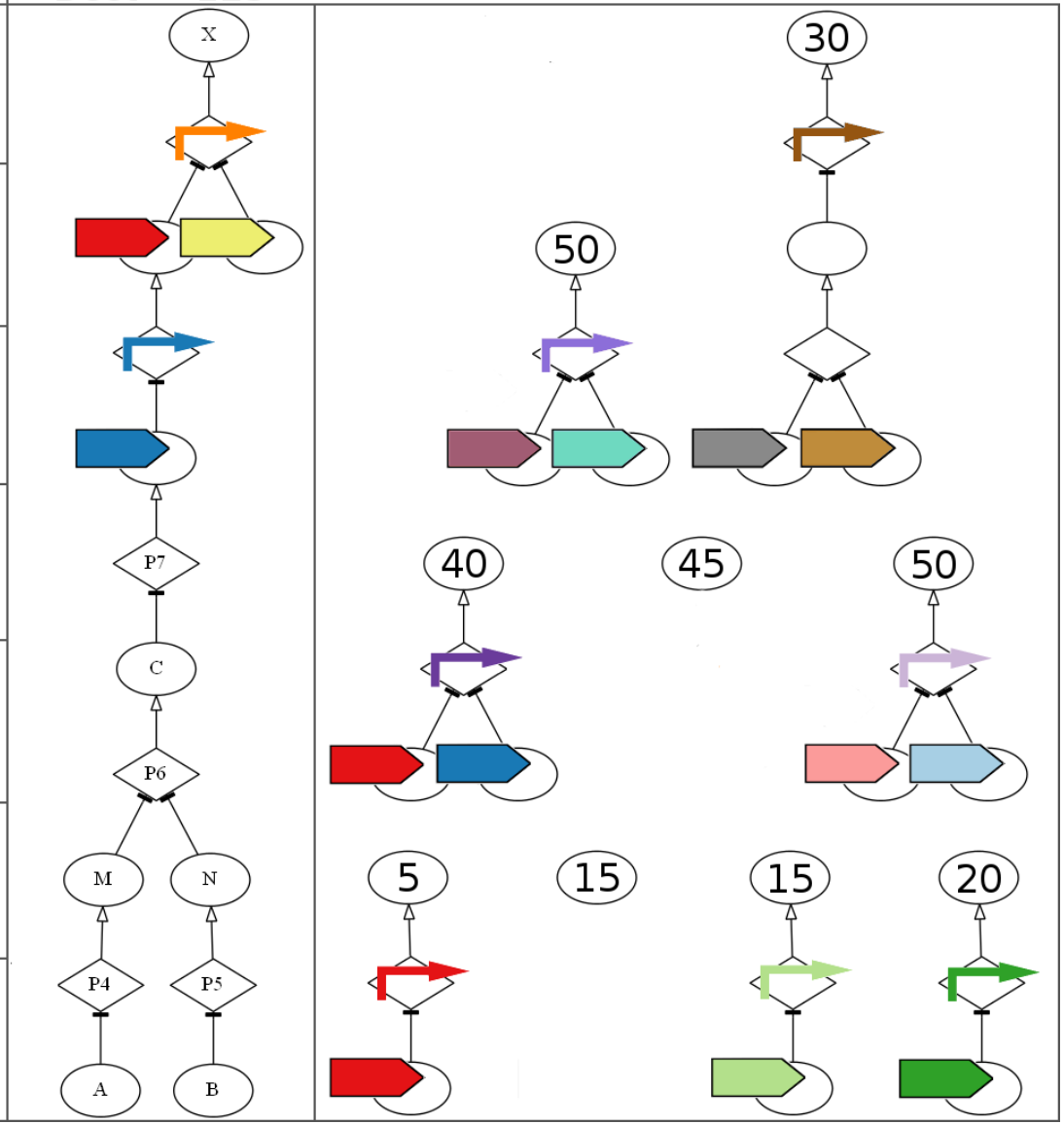
Best = 125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	45	45,0	90
O	60	40	100
C		50	
M,N		5,5	
A,N		0,5	
B		0	

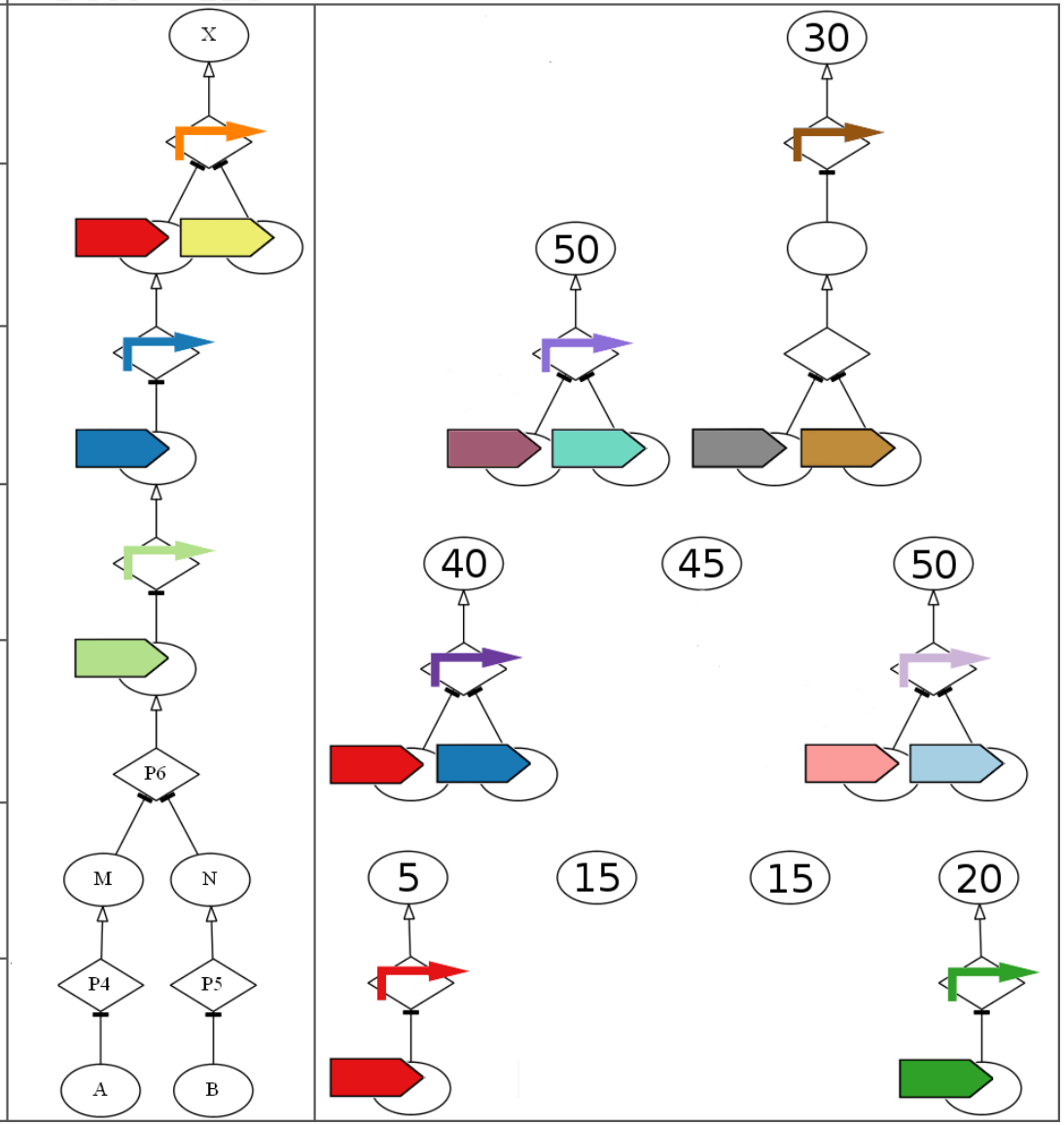
Best = 125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	45	45,0	90
O	60	40	100
C	75	50	125
M,N		5,5	
A,N		0,5	
B		0	

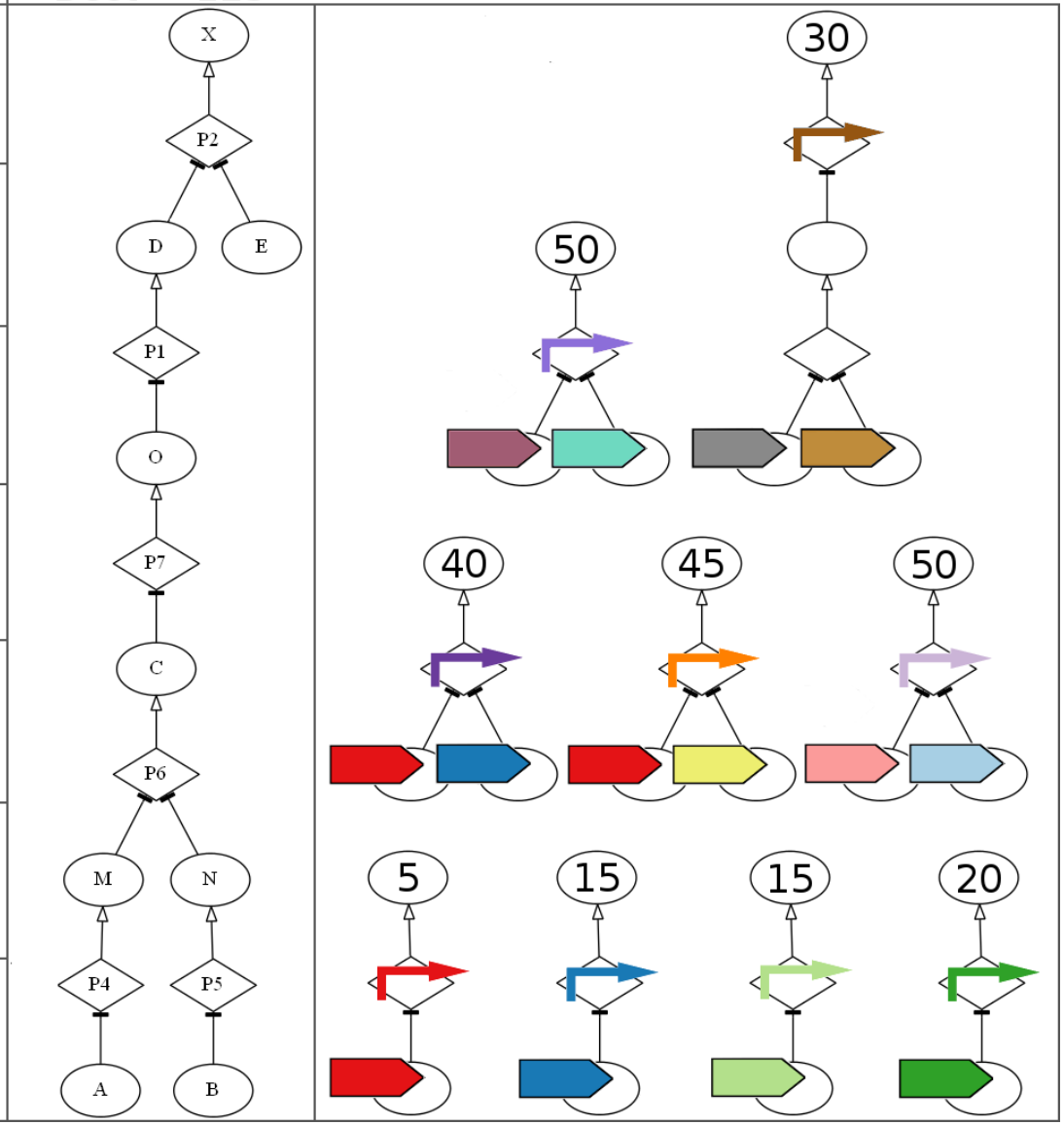
Best = 125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E		45,0	
O		40	
C		50	
M,N		5,5	
A,N		0,5	
B		0	

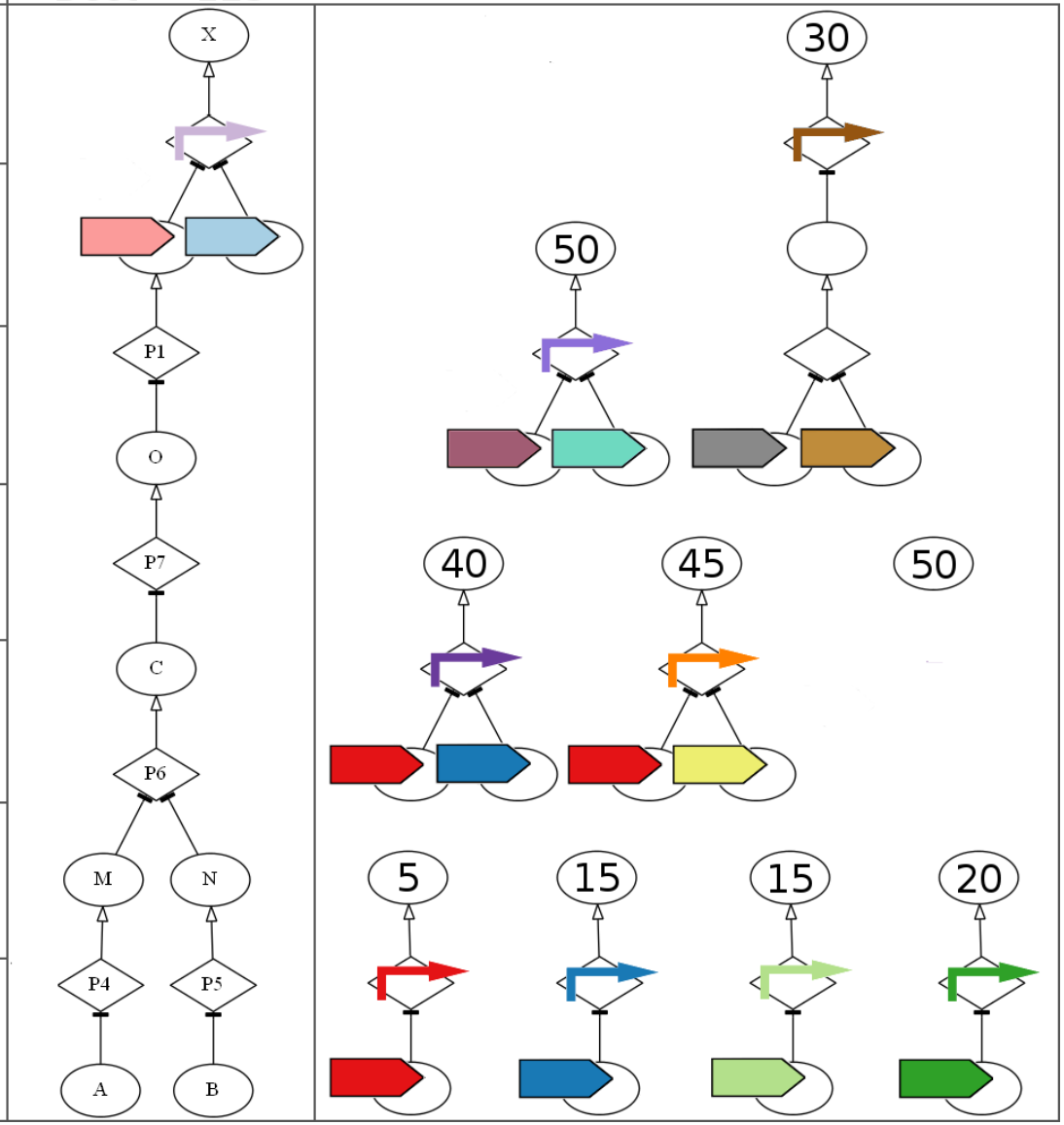
Best = 125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	50	45,0	95
O		40	
C		50	
M,N		5,5	
A,N		0,5	
B		0	

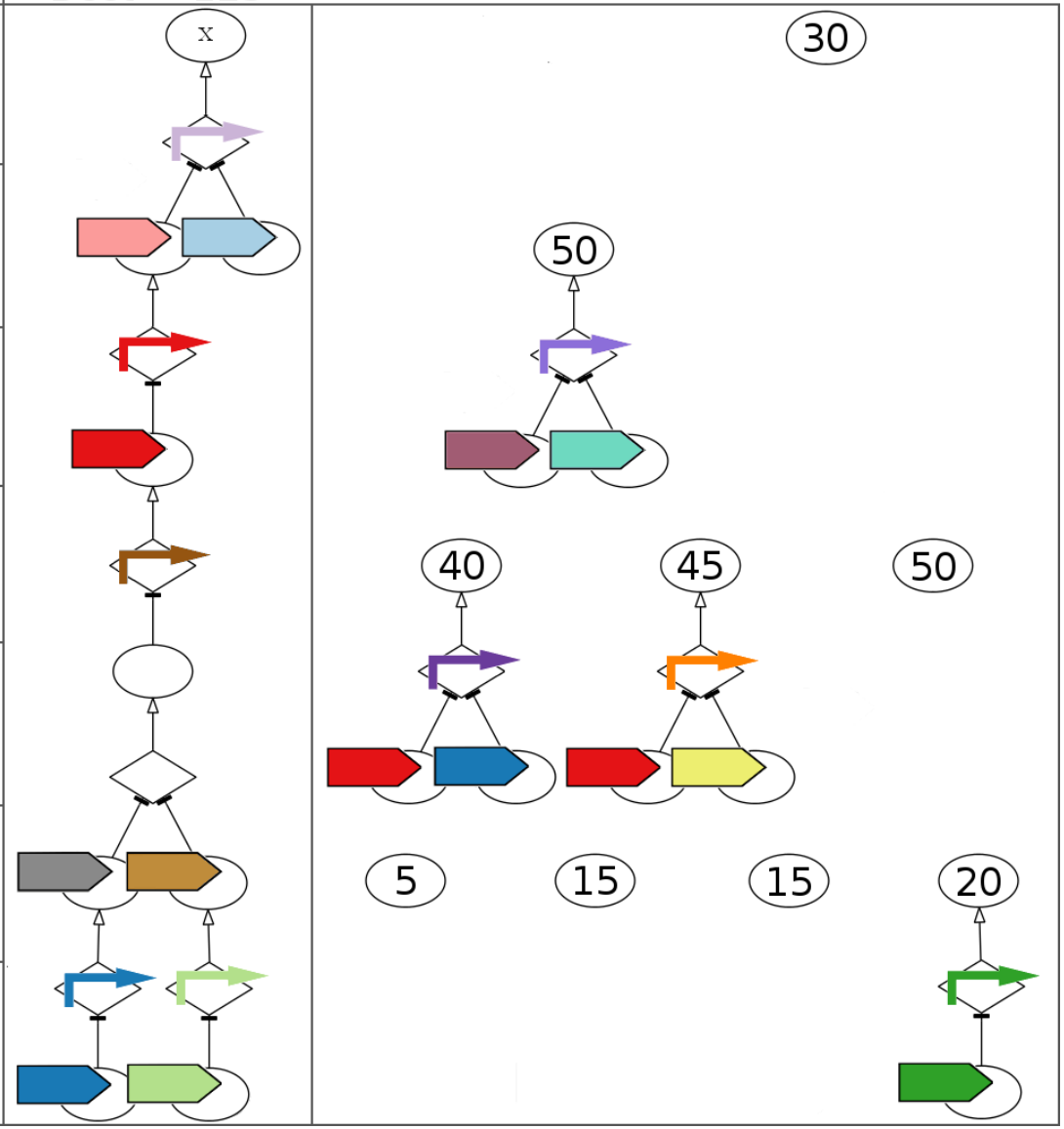
Best = 125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	50	45,0	95
O	55	40	95
C	–	50	–
M,N	85	5,5	95
A,N	100	0,5	105
B	115	0	115

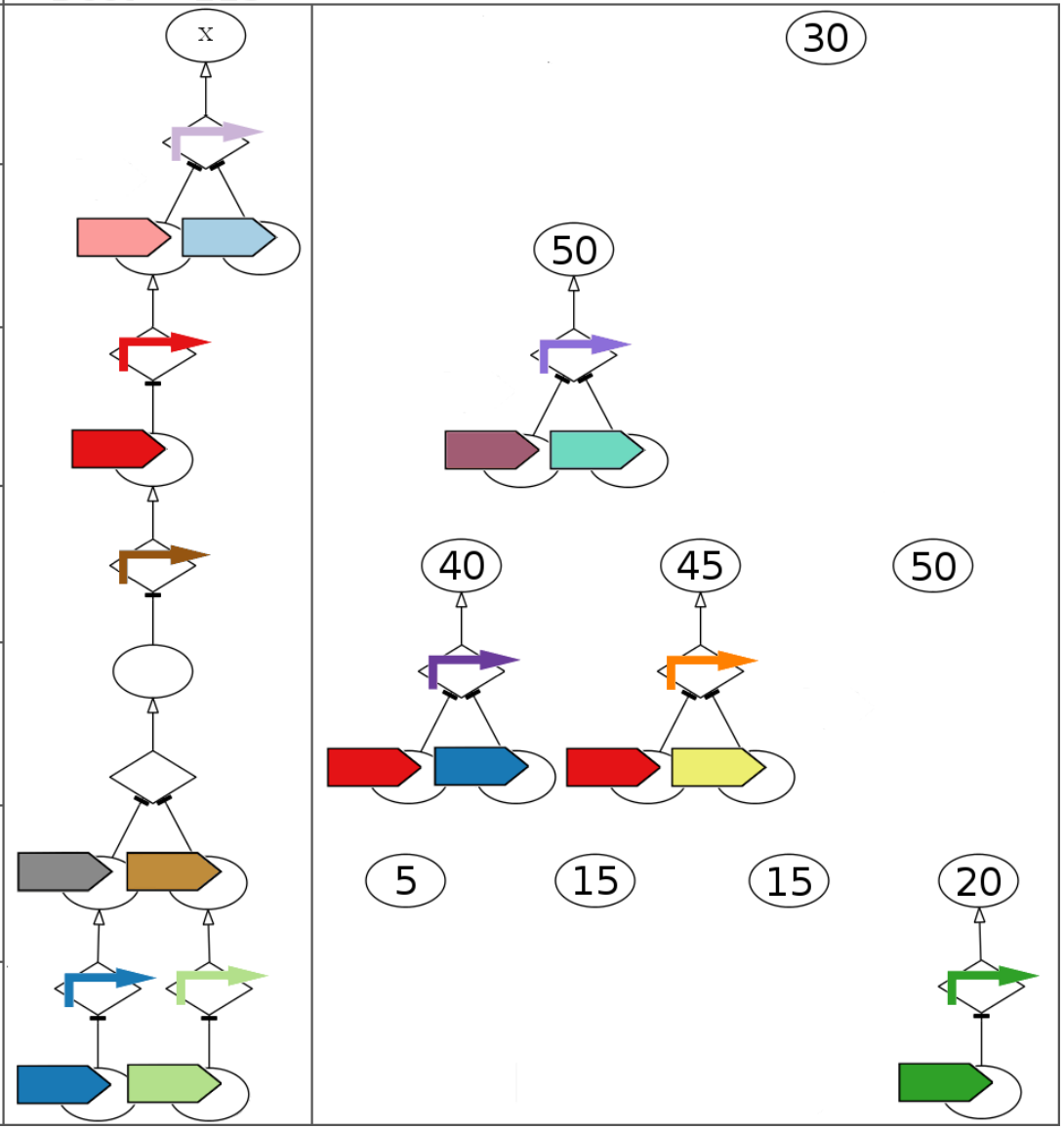
Best = 125



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	50	45,0	95
O	55	40	95
C	–	50	–
M,N	85	5,5	95
A,N	100	0,5	105
B	115	0	115

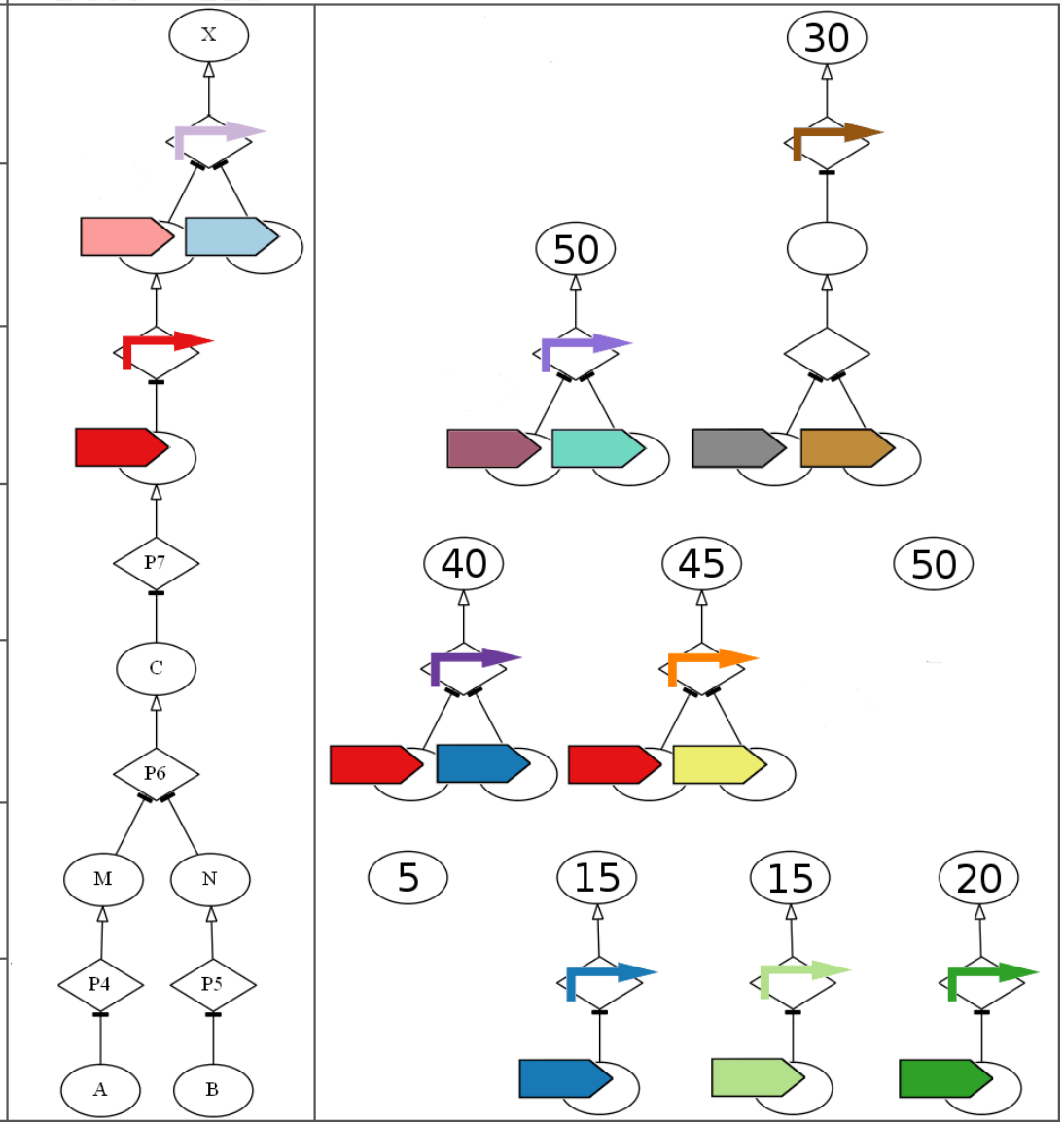
Best = 115



Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	50	45,0	95
O	55	40	95
C		50	
M,N		5,5	
A,N		0,5	
B		0	

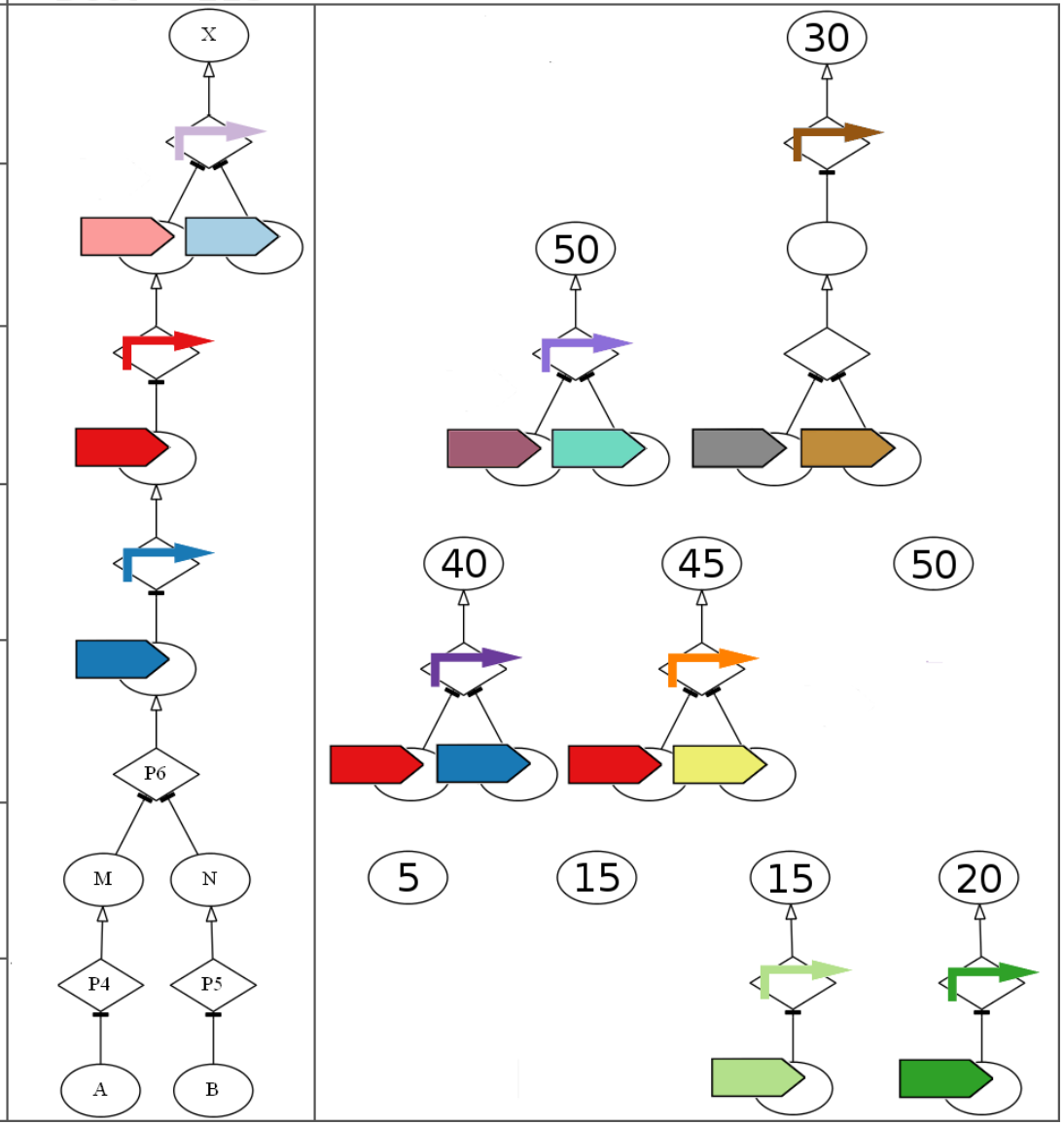
Best = 115



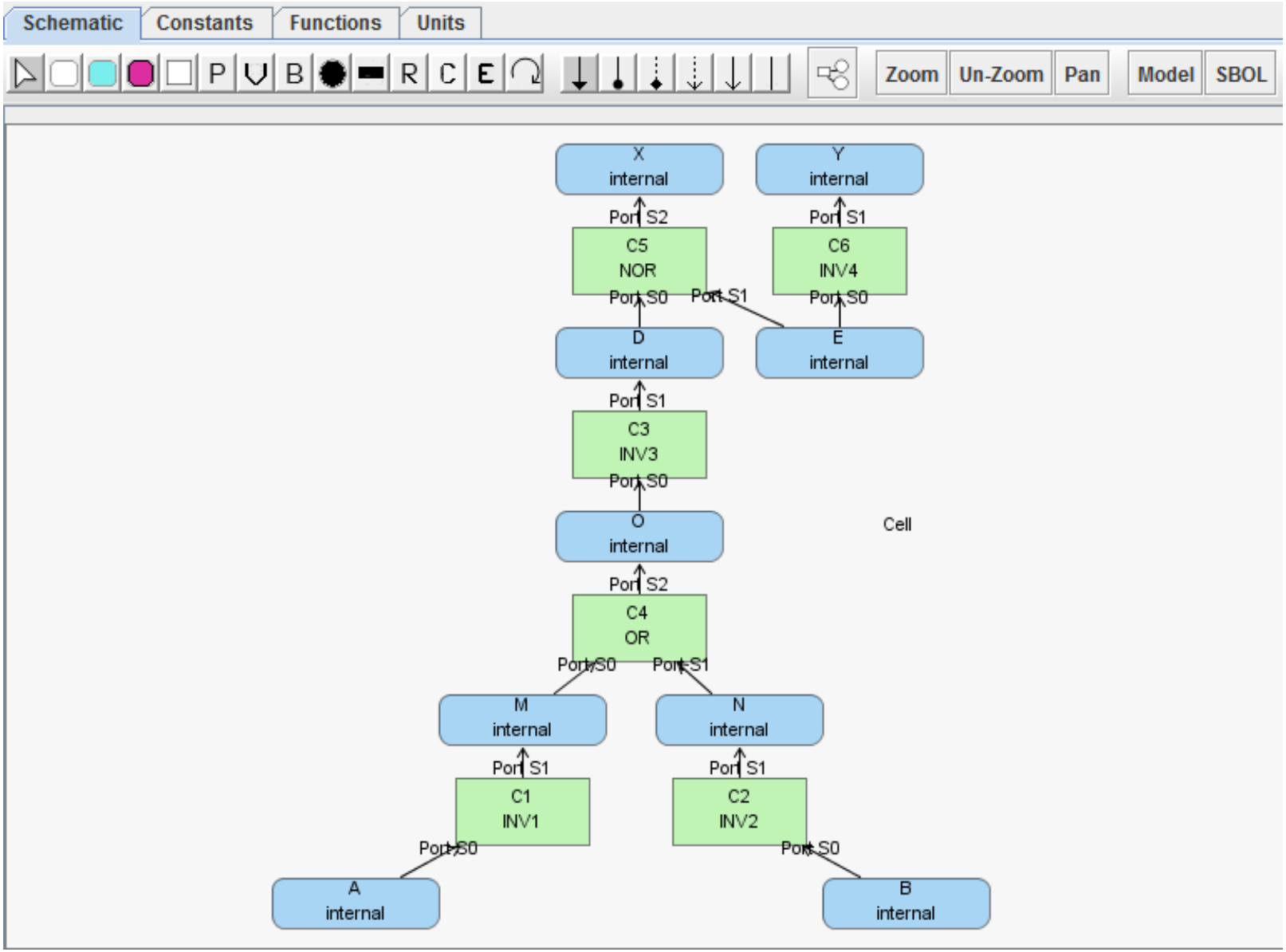
Covering

Node	Current	Bound	Estimate
X	0	85	85
D,E	50	45,0	95
O	55	40	95
C	70	50	120
M,N		5,5	
A,N		0,5	
B		0	

Best = 115



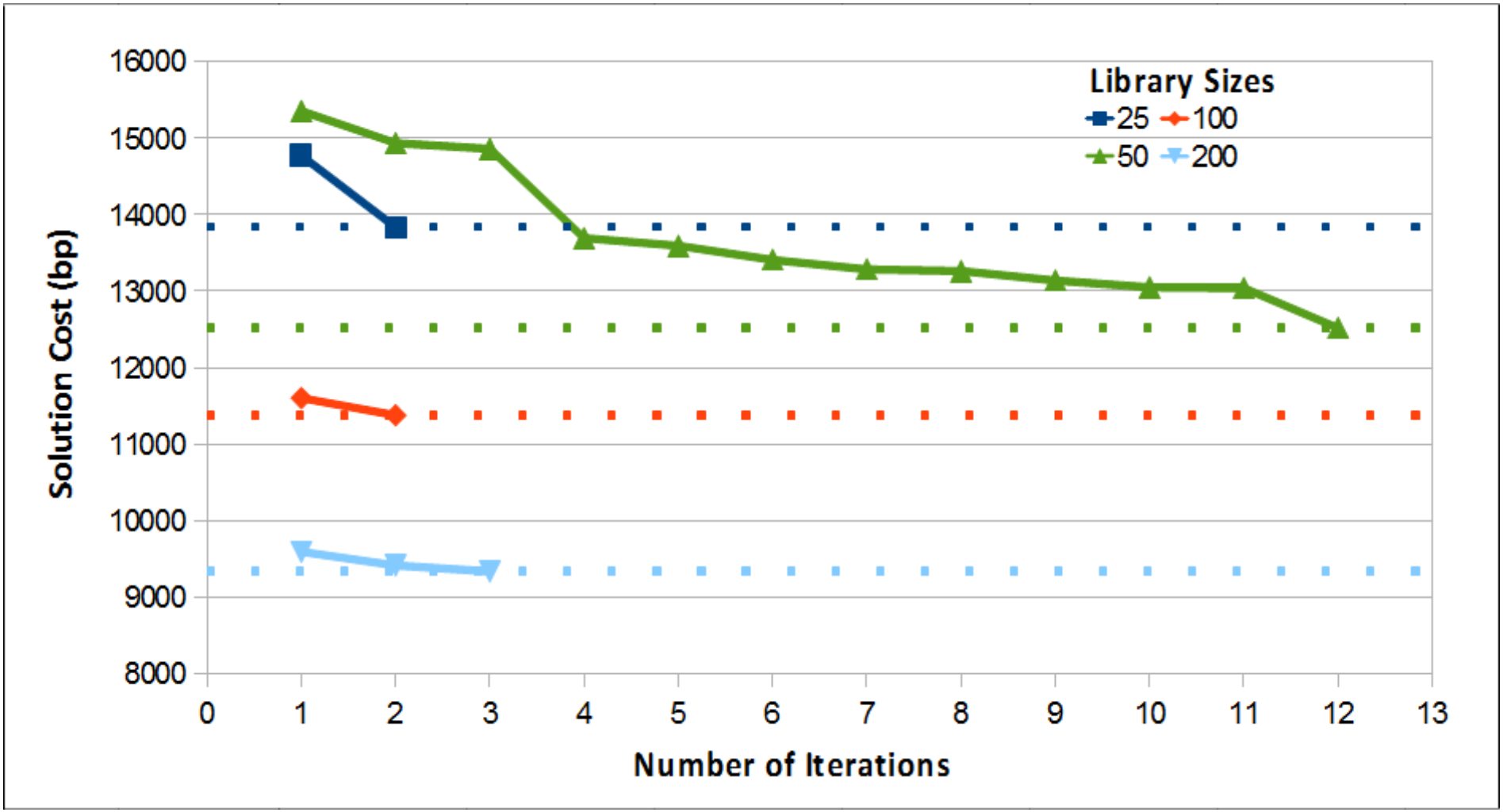
Output Design in iBioSim



Solution Cost Versus Iteration for OAI Circuit

AOI	Runtime (s)				Solution Cost (bp)			
Algorithm	**Library Size**				**Library Size**			
	25	**50**	**100**	**200**	**25**	**50**	**100**	**200**
Exhaustive	0.2	1	60	>1h	3662	2871	2946	n/a
Greedy	0.007	0.01	0.02	0.03	3662	2871	2946	2913
Branch/bound	0.008	0.01	0.02	0.03	3662	2871	2946	2913
NAND/NOR	**Runtime (s)**				**Solution Cost (bp)**			
Algorithm	**Library Size**				**Library Size**			
	25	**50**	**100**	**200**	**25**	**50**	**100**	**200**
Exhaustive	1	>1h	>1h	>1h	11178	n/a	n/a	n/a
Greedy	0.02	0.03	0.04	0.06	13219	10933	11107	8482
Branch/bound	0.2	1	0.7	1	11178	10931	10592	8270
OAI Cascade	**Runtime (s)**				**Solution Cost (bp)**			
Algorithm	**Library Size**				**Library Size**			
	25	**50**	**100**	**200**	**25**	**50**	**100**	**200**
Exhaustive	>1h	>1h	>1h	>1h	n/a	n/a	n/a	n/a
Greedy	2	0.03	0.04	0.07	14774	15357	11603	9592
Branch/bound	4	100	10	40	13836	12518	11377	9335

Solution Cost Versus Iteration for OAI Circuit

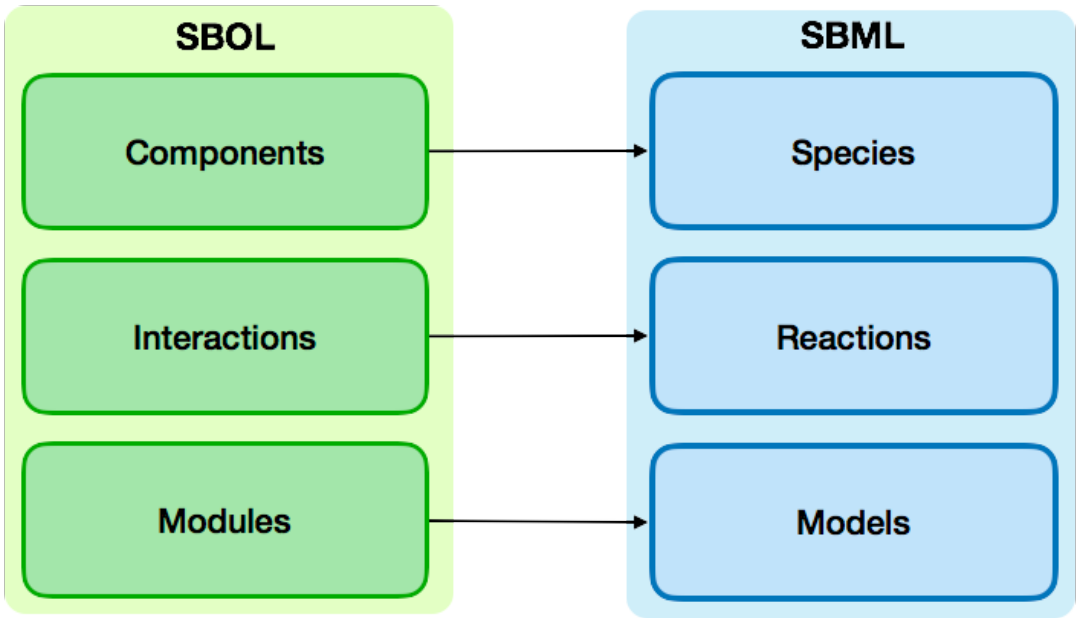


Verification

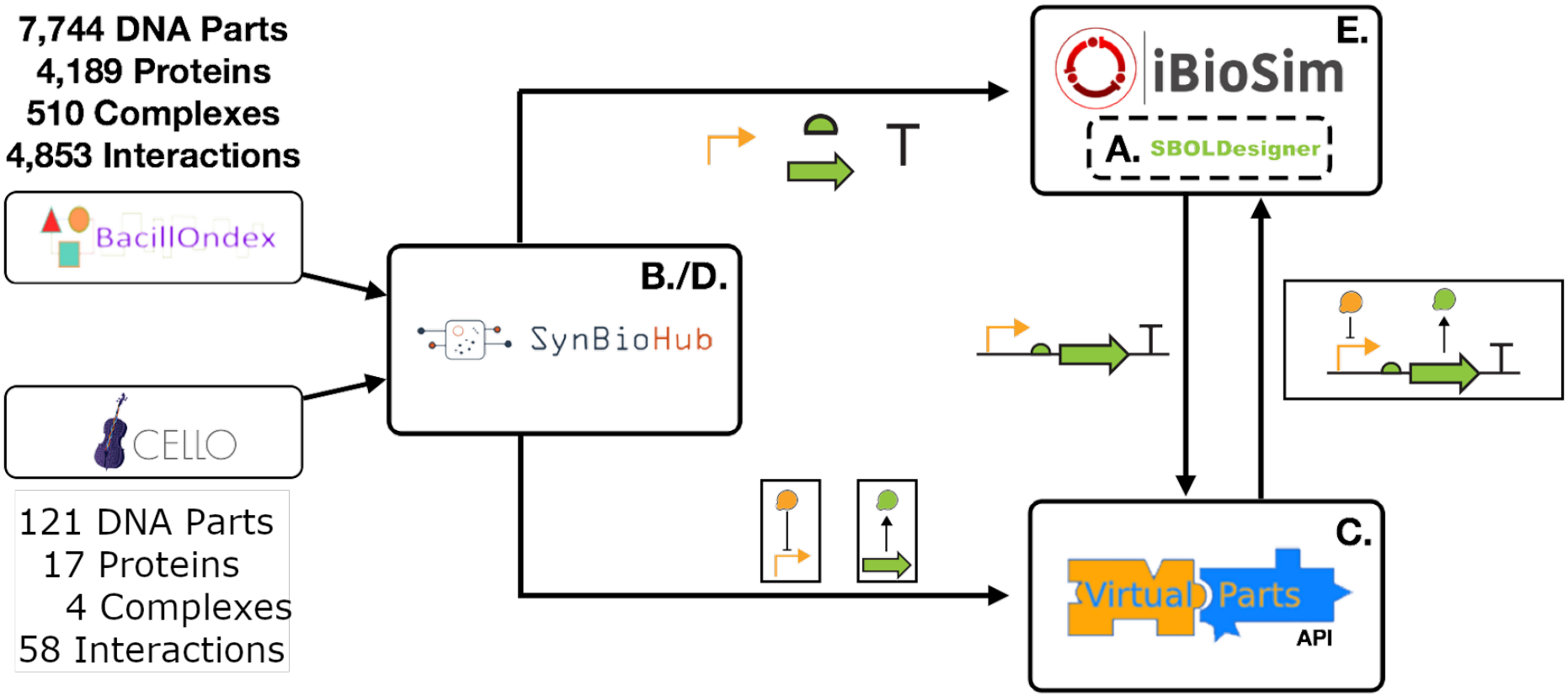
- Large amount of information is being produced about biological parts that can be used to build complex designs.
- Computational modeling *in silico* are created manually to predict the behavior of designs implemented *in vivo* or *in vitro*.
- Models in these designs have functional relationships and design constraints between parts that can be used for simulation.
- Computational models can be created by extracting knowledge about the DNA components and their interacting entities.
- These models can then be used to create simulations to verify performance of genetic circuit.

Deriving Dynamic Models

- SBOL to SBML conversion is applied to verify the behavior of a design.
- Ontology terms are used to translate components between standards.

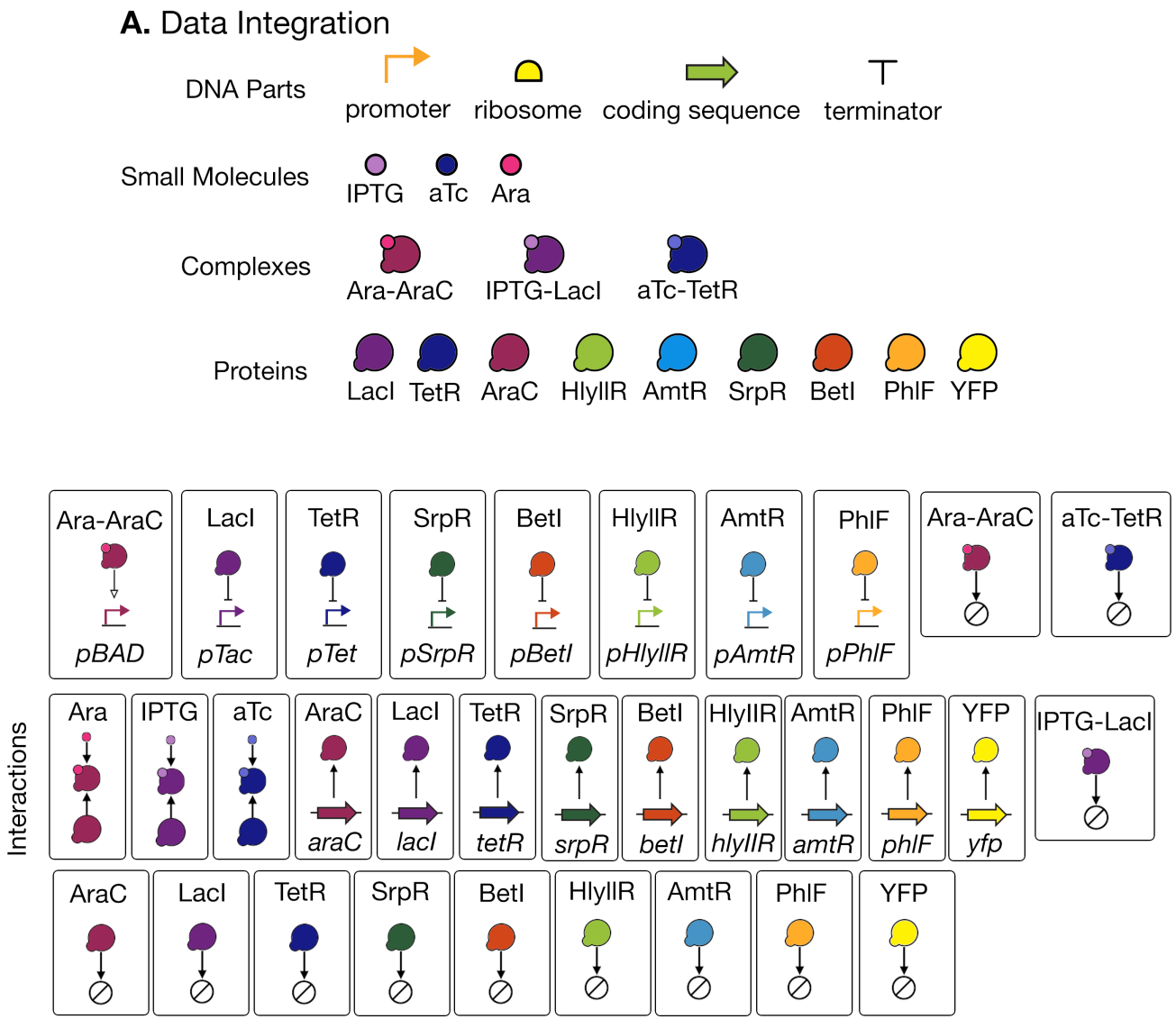


Model Generation Workflow

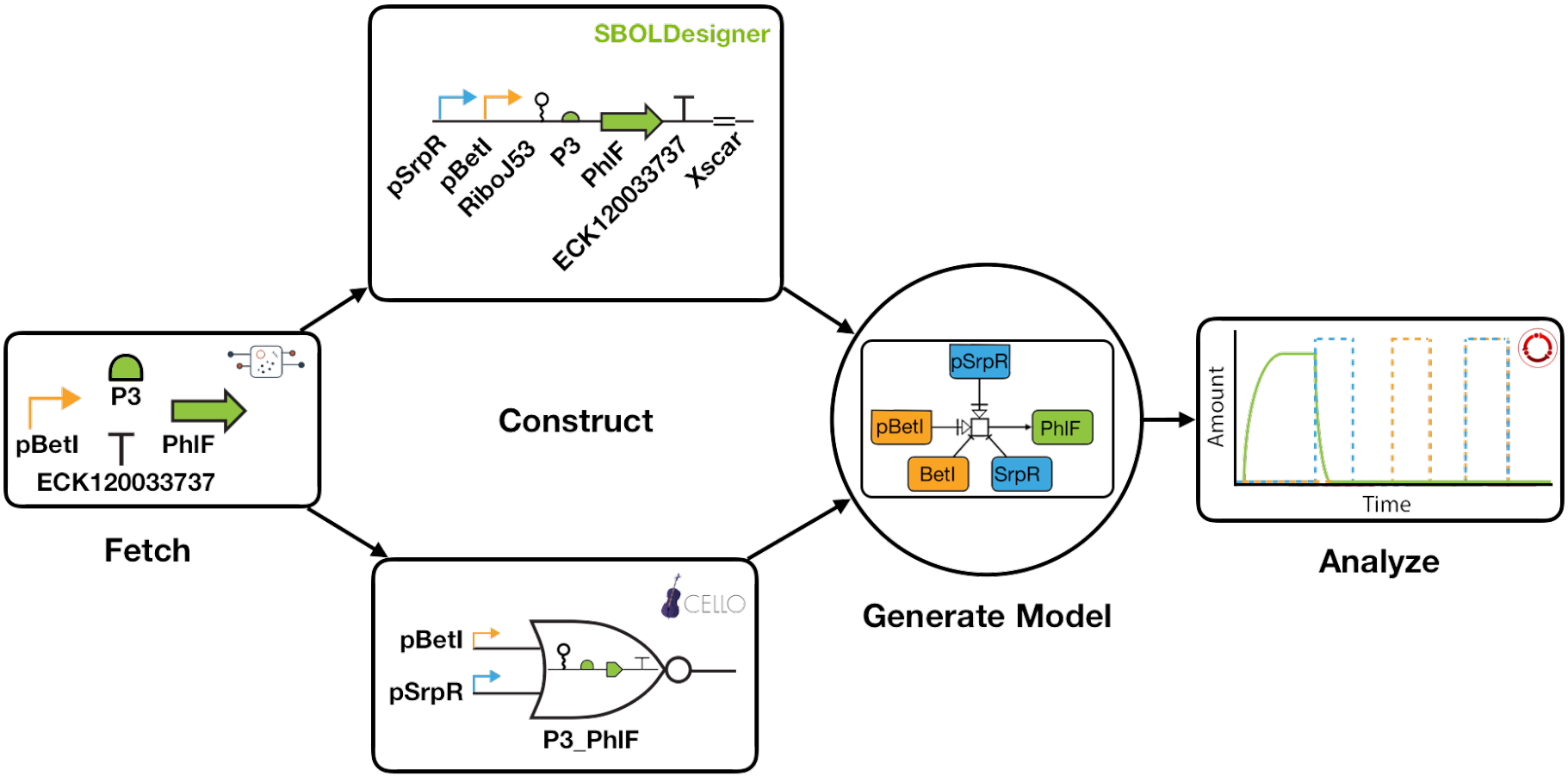


Mısırlı et al., *ACS Synthetic Biology* (2018).

Data Integration: Cello Part Library

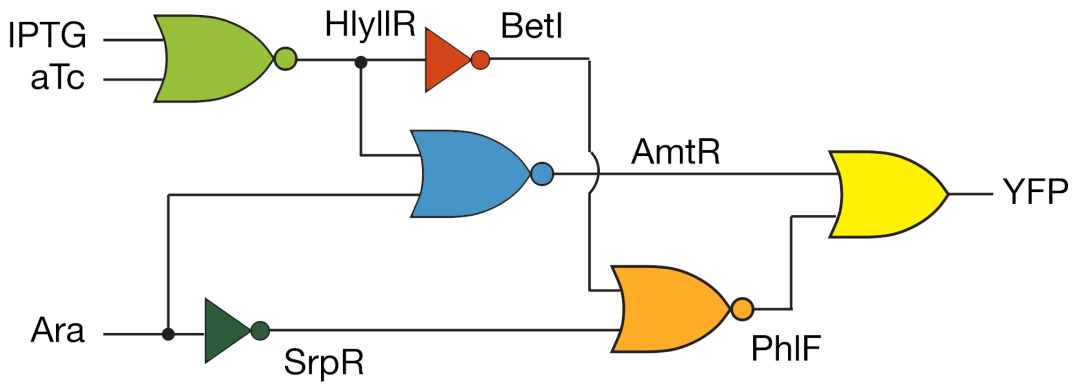


Genetic Circuit Construction

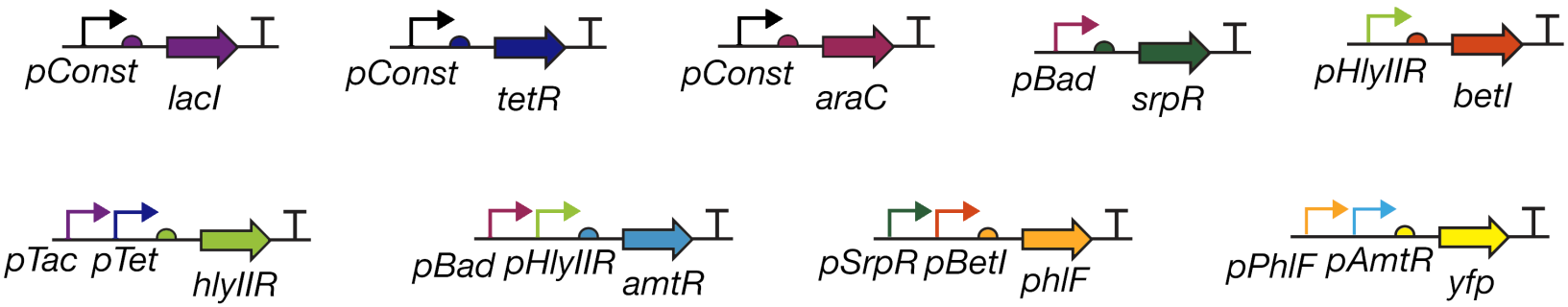


Genetic Circuit Construction: Rule 30 Example

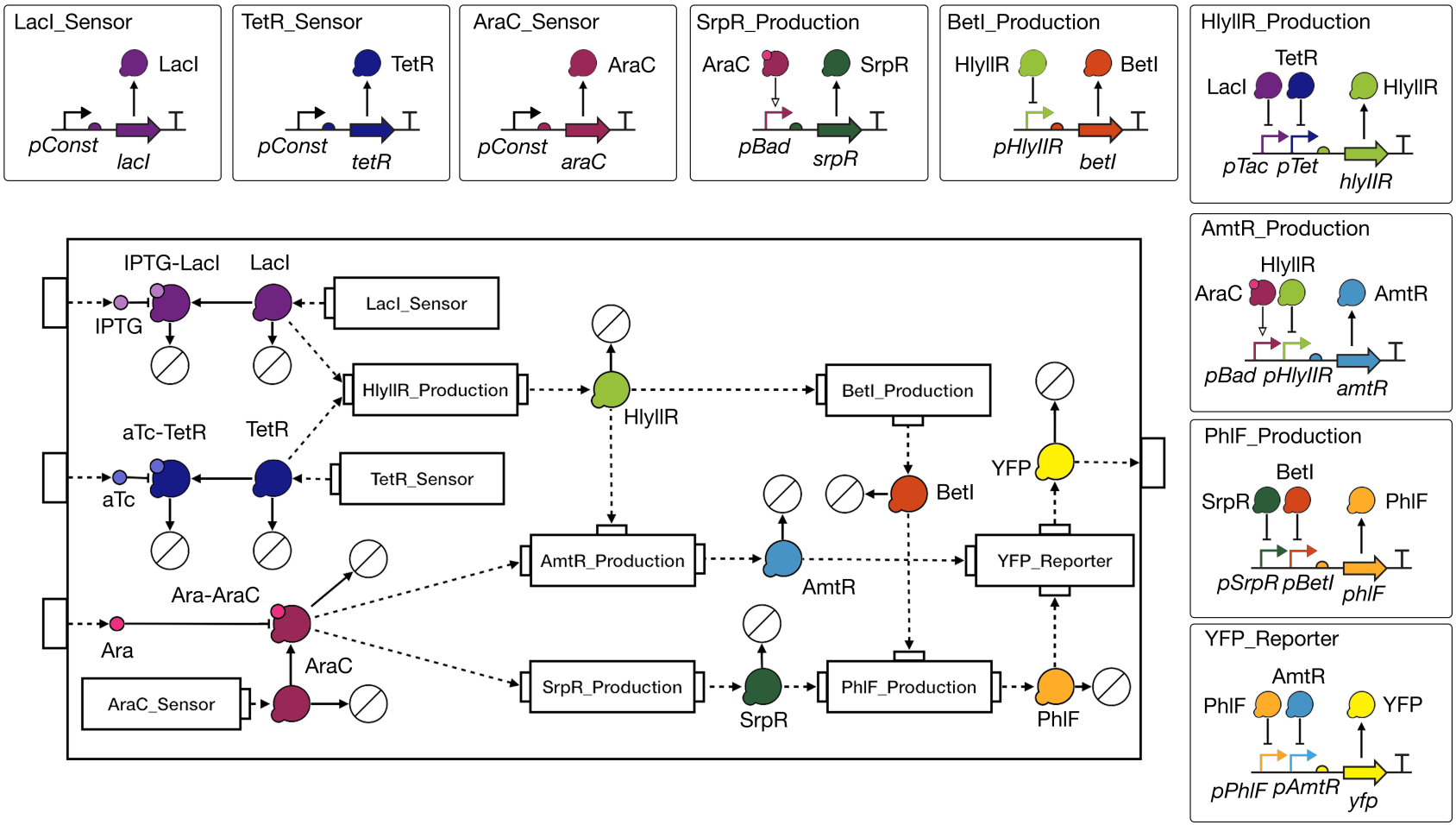
B. Rule 30



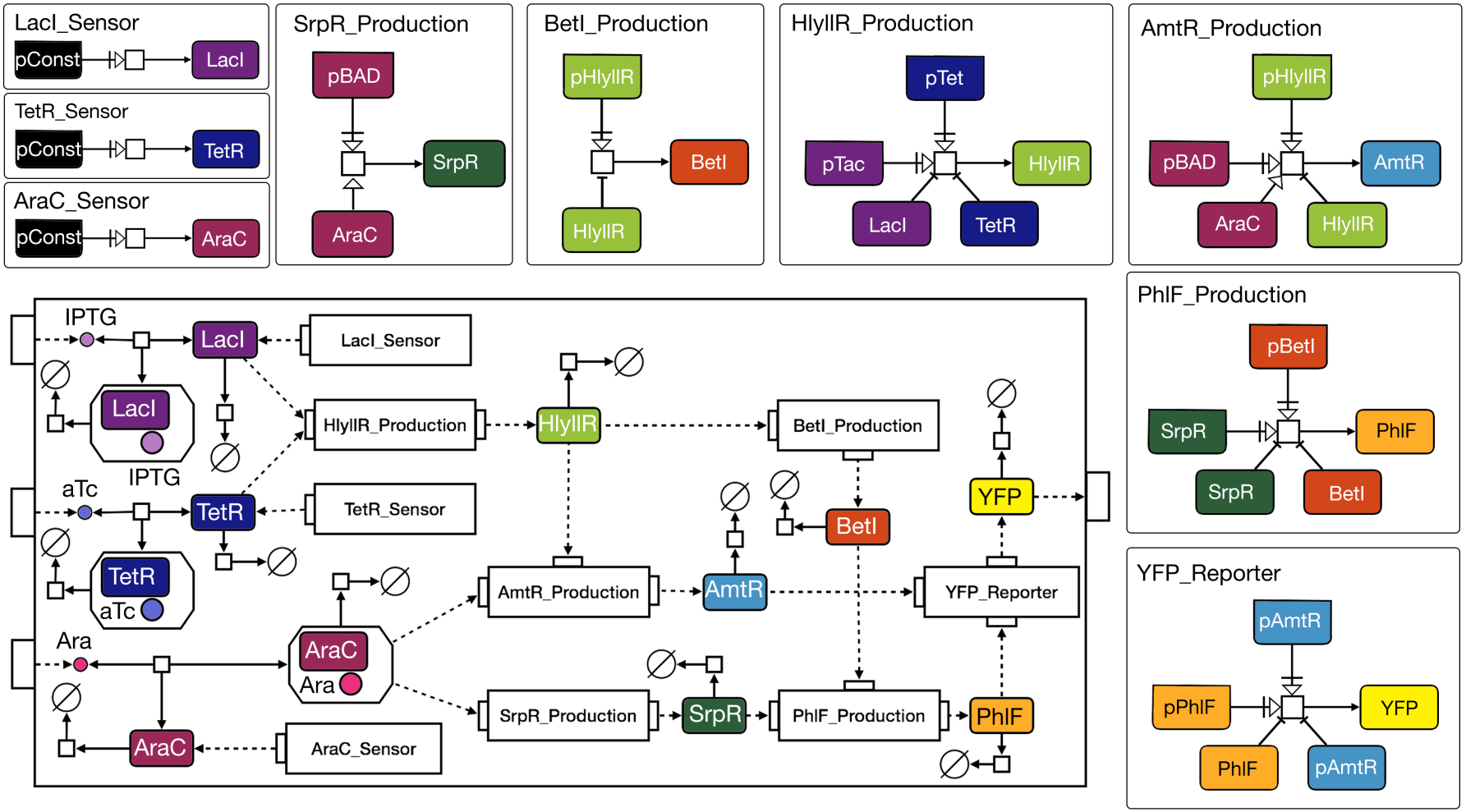
C. Genetic Circuit Construction



Enriched SBOL Representation: Rule 30 Example

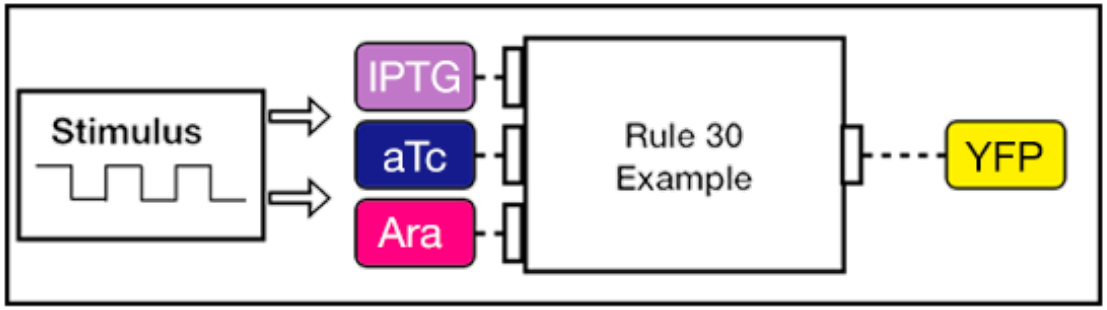


Dynamic SBML Model: Rule 30 Example

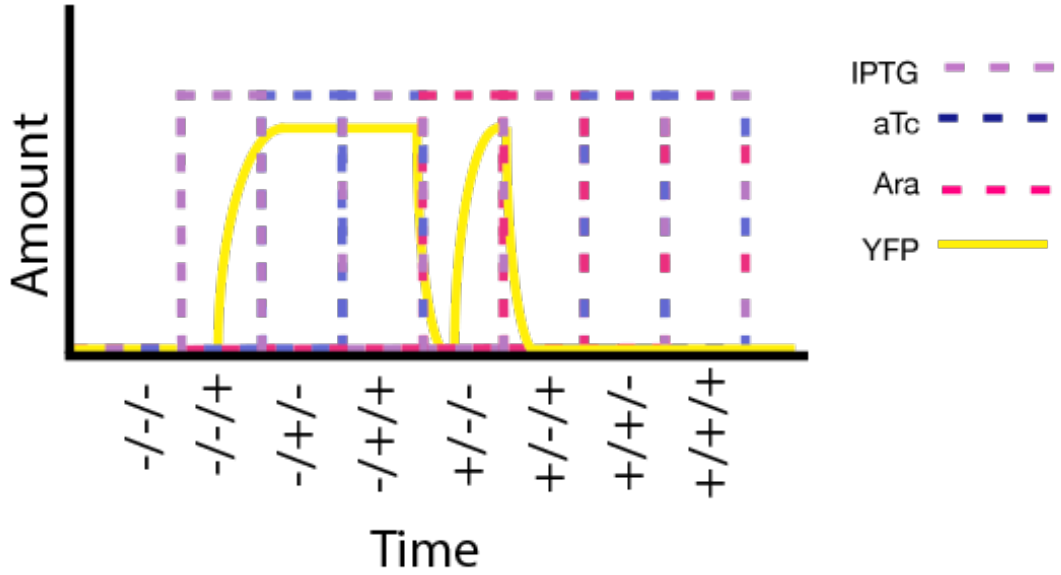


Simulation: Rule 30 Example

A. Testing Environment



B. Simulation



Sequential Genetic Circuits

- Existing frameworks are limited to combinational circuits.
- Memory circuits have been created but have not been introduced to technology mapping tools.
- The impact of designing genetic circuits sequentially needs to be explored on a deeper level.

Synchronous and Asynchronous Designs

- Sequential circuits can be designed synchronously or asynchronously.
- **Synchronous designs** use a global clock signal to order operations.
- **Asynchronous designs** order operations using handshaking protocols.
- Biological systems are naturally asynchronous.

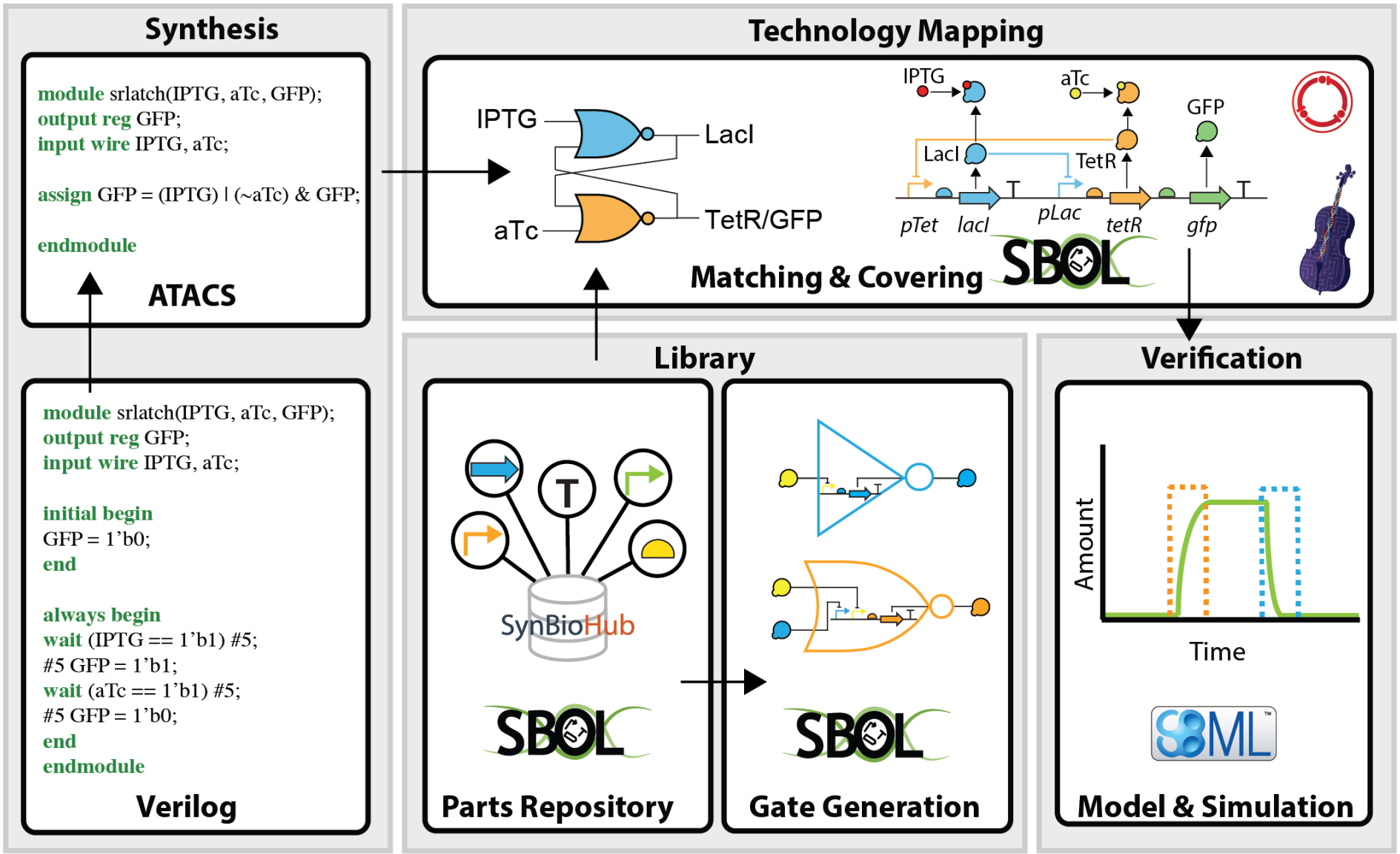
Synchronous Genetic Sensor Example

```
module sync_sensor(Clock, Sensor, Actuator);
  input Clock, Sensor;
  output reg Actuator;
initial begin
  Actuator = 1'b0;
end
always @(posedge Clock) begin
  Actuator = Sensor;
end
endmodule
```

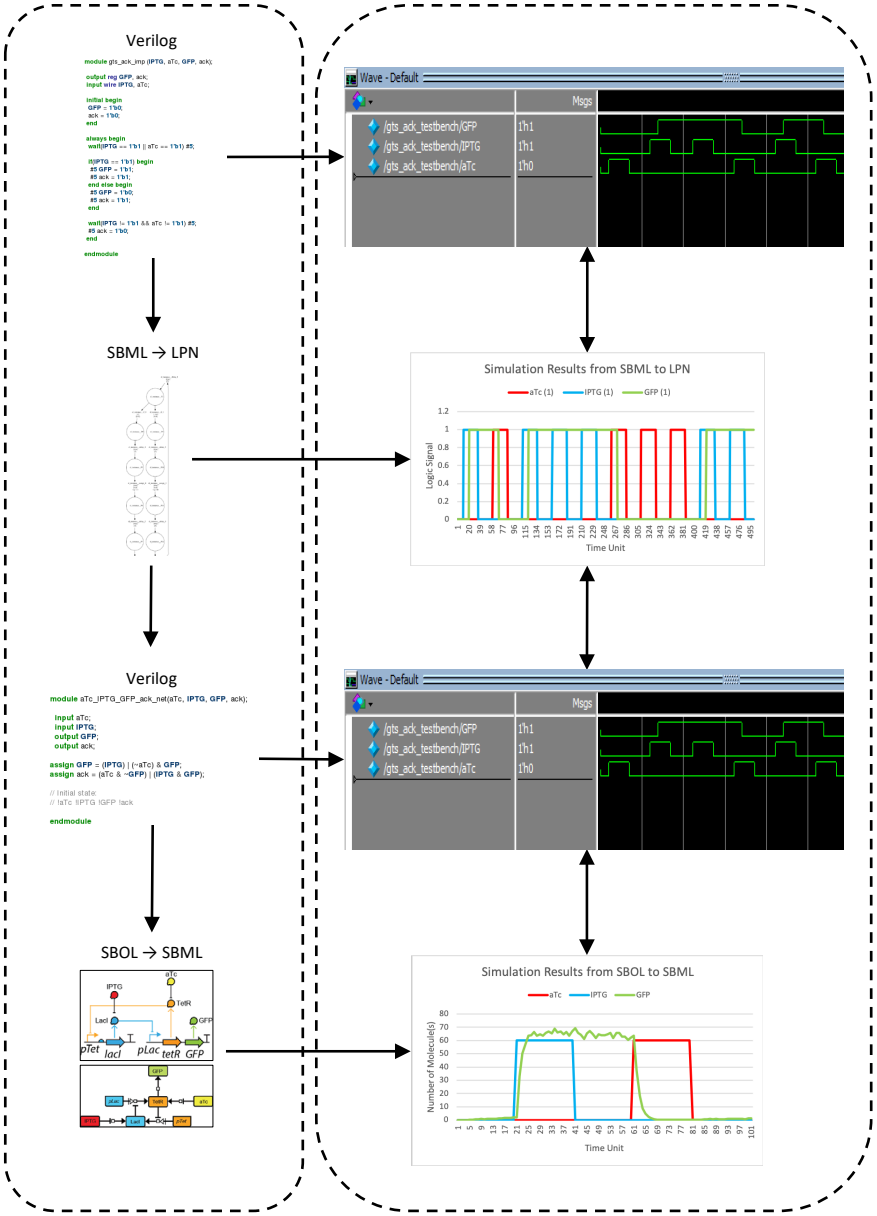
Asynchronous Genetic Sensor Example

```
module async_sensor(Start, Sensor, Actuator);
  input wire Start, Sensor;
  output reg Actuator;
initial begin
  Actuator = 1'b0;
end
always begin
  wait (Start == 1'b1 && Sensor == 1'b1);
  #5 Actuator = 1'b1;
  wait (Sensor == 1'b0);
  #5 Actuator = 1'b0;
end
endmodule
```

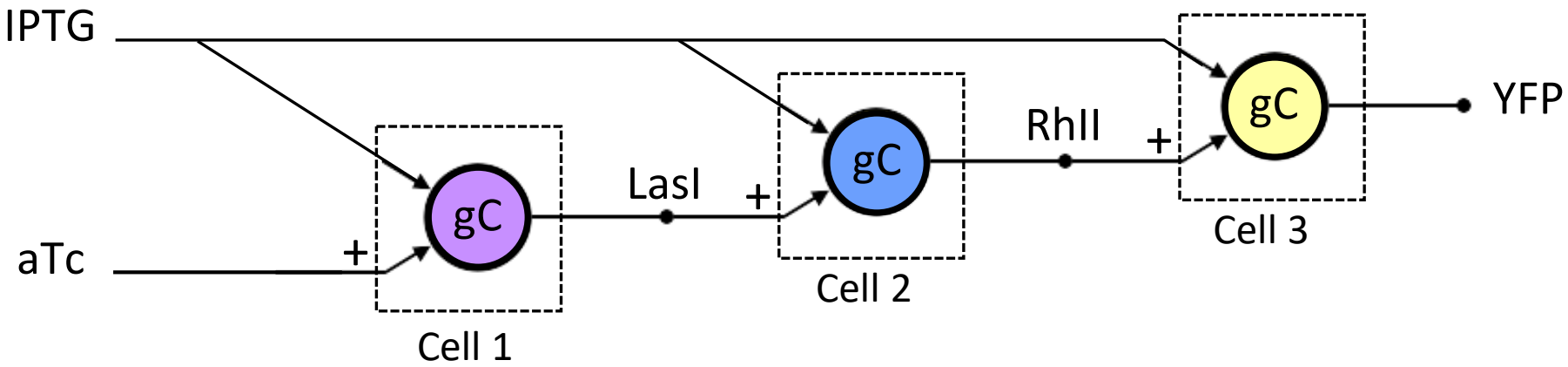
Asynchronous Genetic Design Workflow



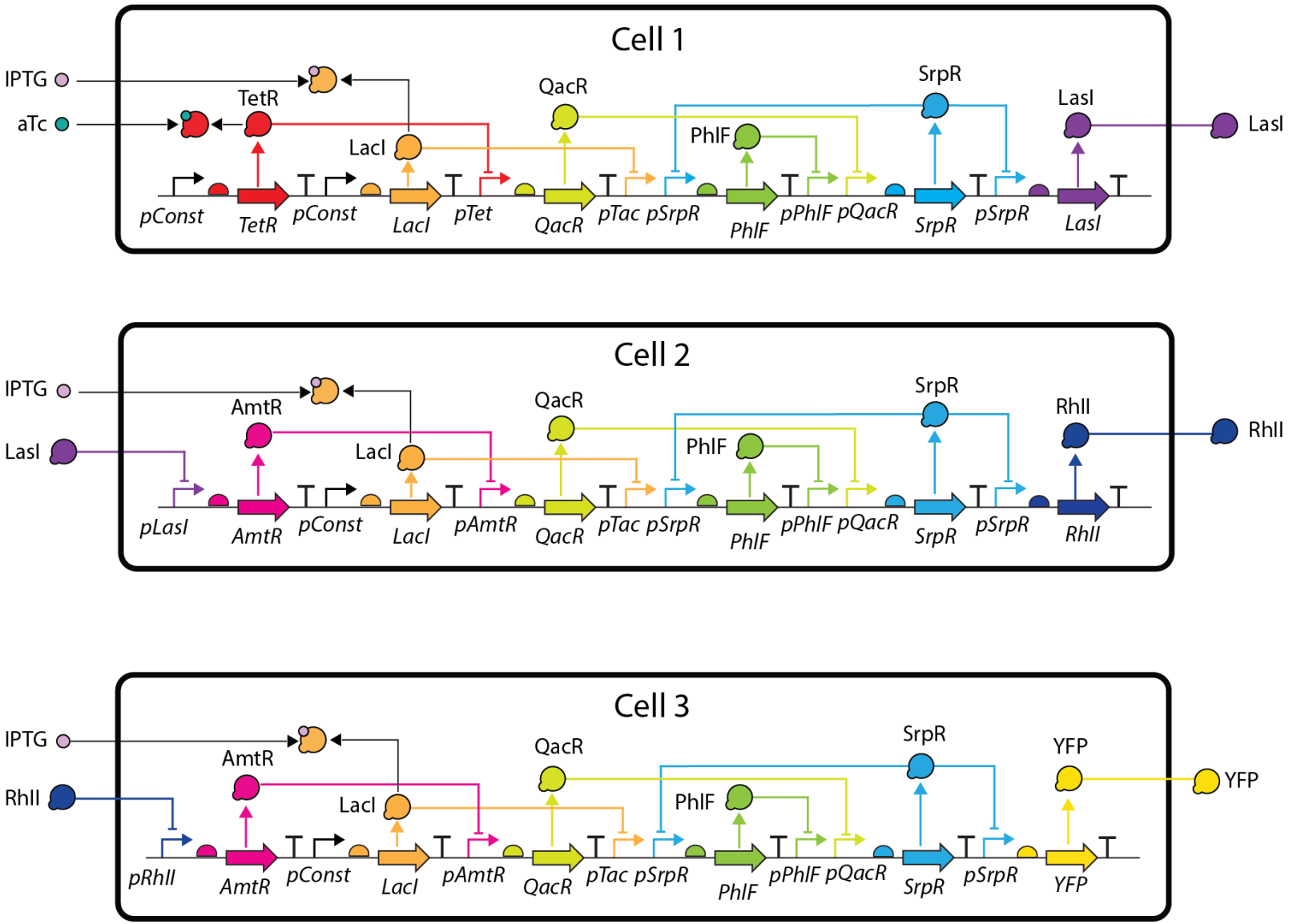
Verification Process



Genetic Sensor Design



Genetic Sensor Design



Future Work

- Asynchronous genetic circuit design workflow needs to be fully automated and seamless.
- Verilog templates should be developed to help designers who are not familiar with Verilog and/or asynchronous design.
- Hazards (i.e. unintended signal changes) in asynchronous genetic circuits should be more carefully considered and evaluated.
- Some asynchronous genetic designs produced by this workflow should be built and tested in the laboratory.