Utilizing Signal Temporal Logic to Characterize and Compose Modules in Synthetic Biology

Curtis Madsen, Prashant Vaidyanathan, Cristian-Ioan Vasile, Rachael Ivison, Junmin Wang, Calin Belta, and Douglas Densmore

IWBDA – August 16, 2016

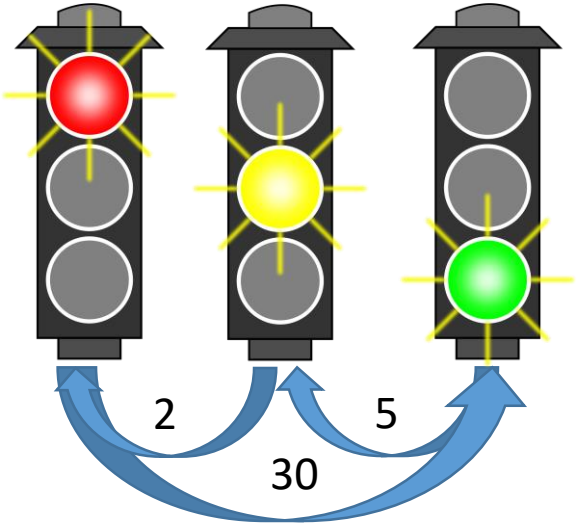
Introduction

- One of the fundamental goals in the field of synthetic biology is to reliably engineer biological systems to respond to environmental conditions according to a pre-determined genetic program.
- Using Boolean logic functions, synthetic biologists have successfully engineered living cells to perform certain functions[1].
- However, it has been difficult to realize the full potential of genetically encoded logic in practical applications without the ability to specify timing and performance of genetic circuits.

[1]A. A. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. A. Strychalski, D. Ross, D. Densmore, and C. A. Voigt, “Genetic circuit design automation,” Science.

Performance Specifications

- To remedy this issue, we propose using performance specifications.
- For example, we can give a performance specification for a traffic light:
  - A traffic light should remain green *until* a pedestrian requests a walk signal. *Within 5 seconds* of receiving the request, the traffic light should change to yellow for *2 seconds*, and then change to red for *30 seconds* before switching back to green.



Performance Specifications

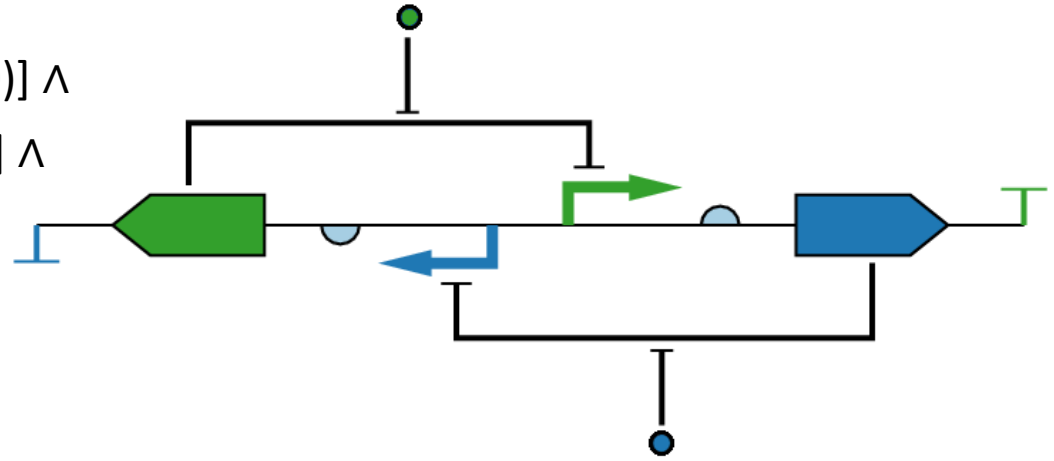
- Temporal logic can be used to write performance specifications as it allows for reasoning about behavior over time.
- In particular, we use Signal Temporal Logic (STL) as it allows for the specification of requirements on signals at specific times leading to a level of expressiveness necessary for genetic circuit design.

$[G\ [0,200)\ (aTc > 30 \wedge TetR > 30)] \wedge$

$[F\ [0,200)\ G\ [0,200)\ (TetR \leq 30)] \wedge$

$[G\ [400,600)\ (IPTG > 30)] \wedge$

$[F\ [400,800)\ (TetR > 30)]$



Temporal Logic Syntax

- Logical Operators:
  - Conjunction: $\phi \wedge \psi$
  - Disjunction: $\phi \vee \psi$
  - Implication: $\phi \rightarrow \psi$
  - Negation: $\neg\phi$
- Temporal Operators:
  - Until: $\phi \, U \, \psi$
  - Future (Eventually): $F \, \phi$ (or $\Diamond\phi$)
  - Globally: $G \, \phi$ (or $\Box\phi$)

Until Operator

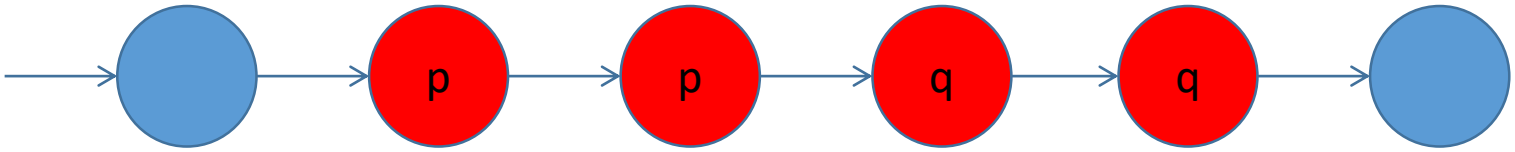
p U q

- q holds at the current or a future position, and p has to hold until that position. At that position p does not have to hold any more.

Until Operator

p U q

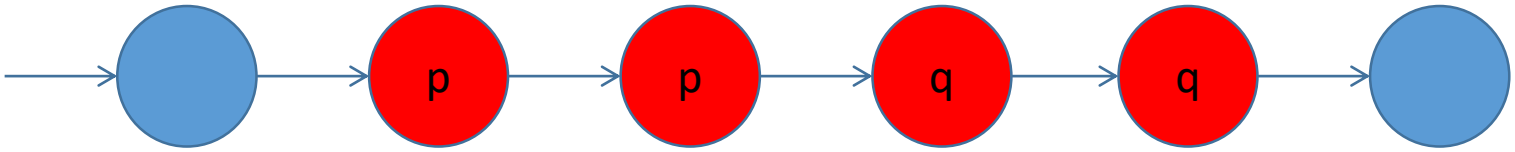
- q holds at the current or a future position, and p has to hold until that position. At that position p does not have to hold any more.

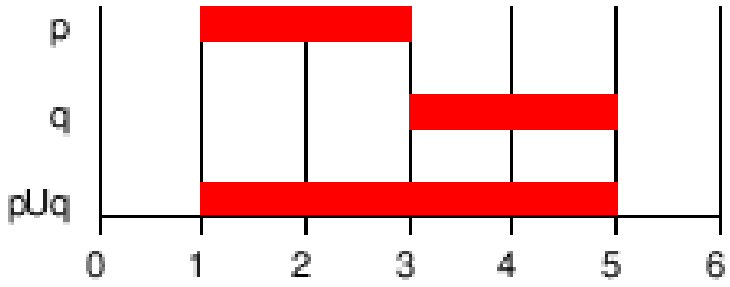


Until Operator

p U q

- q holds at the current or a future position, and p has to hold until that position. At that position p does not have to hold any more.





Future (Eventually) Operator

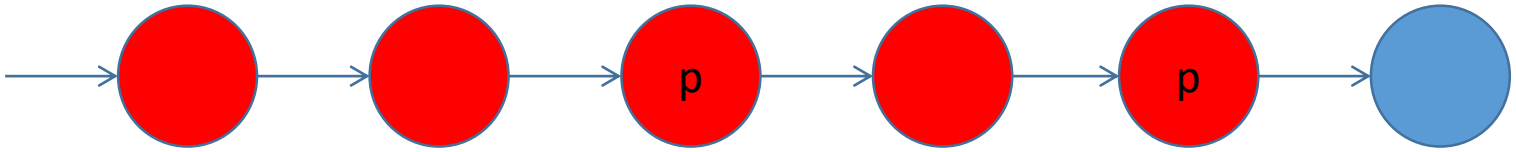
F p

- **F**uture: p eventually has to hold (somewhere on the subsequent path).

Future (Eventually) Operator

F p

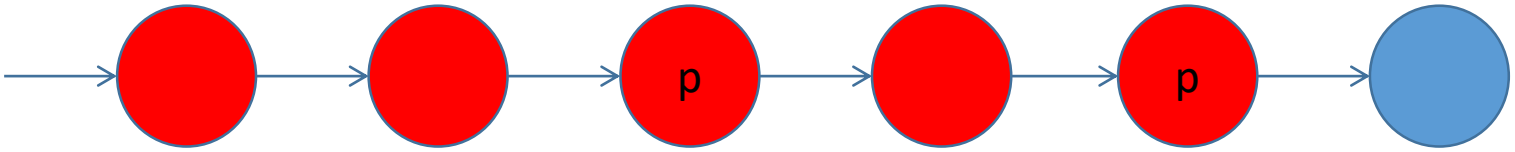
- **F**uture: p eventually has to hold (somewhere on the subsequent path).

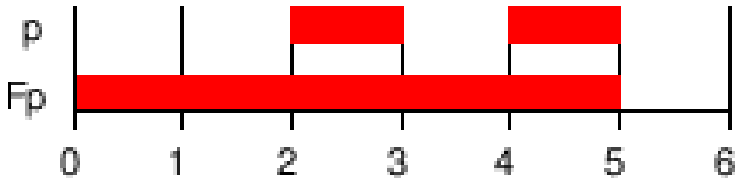


Future (Eventually) Operator

F p

- **F**uture: p eventually has to hold (somewhere on the subsequent path).





Globally Operator

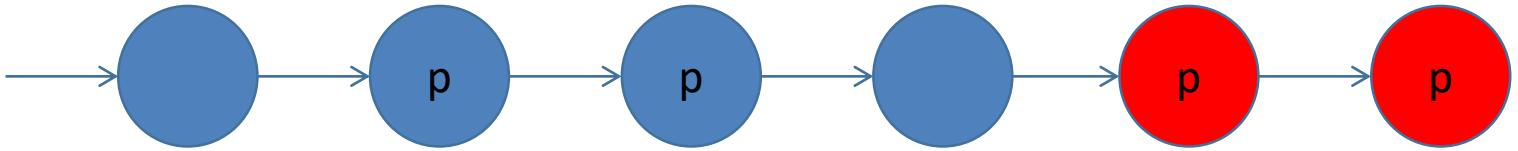
G p

- **G**lobally: p has to hold on the entire subsequent path.

Globally Operator

G p

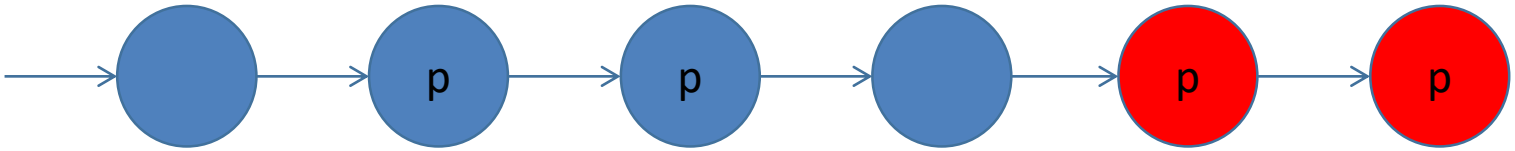
- **G**lobally: p has to hold on the entire subsequent path.

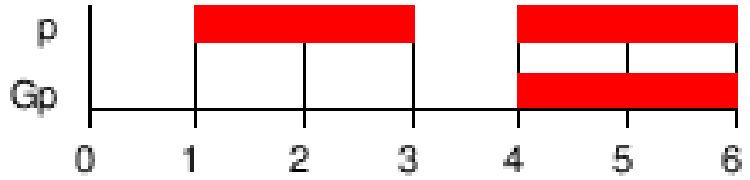


Globally Operator

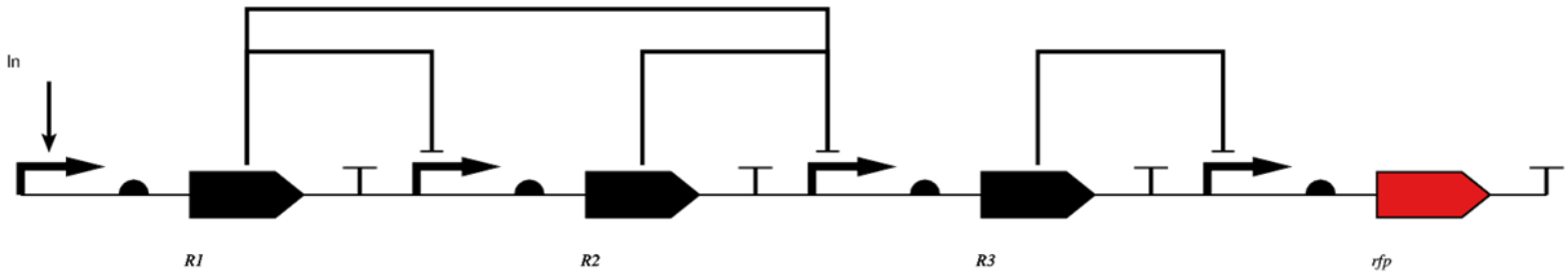
G p

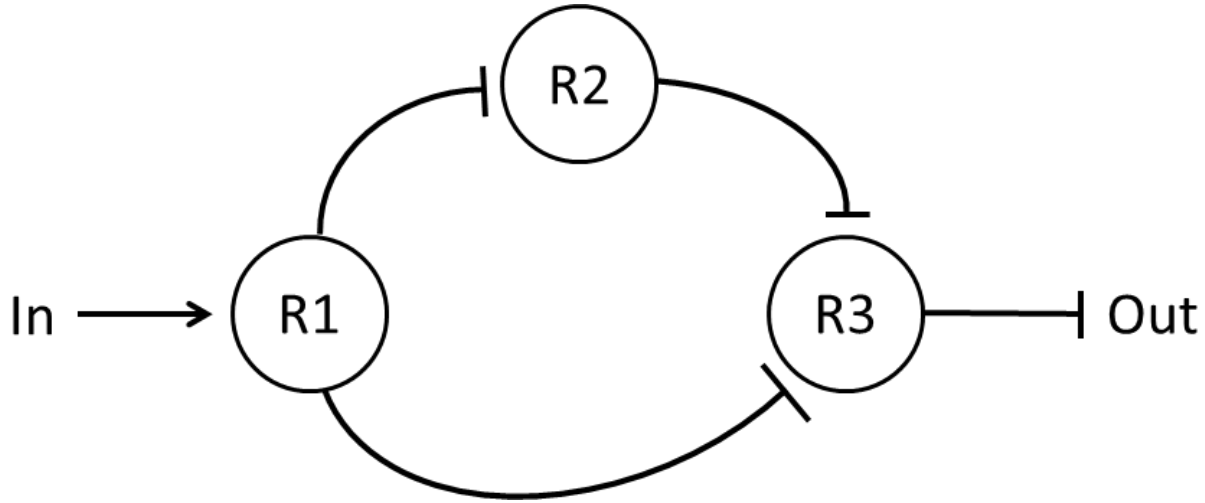
- **G**lobally: p has to hold on the entire subsequent path.



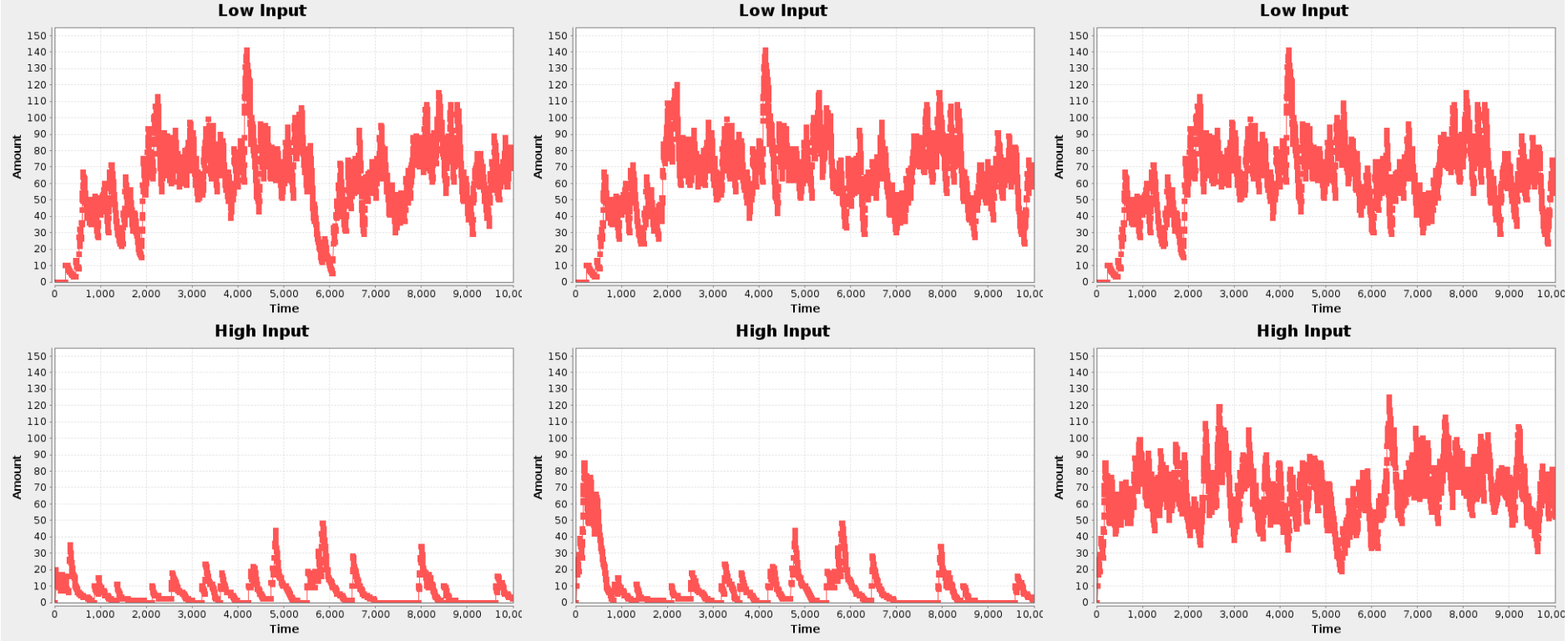


Repressor Incoherent FeedForward Loop (RIFFL)





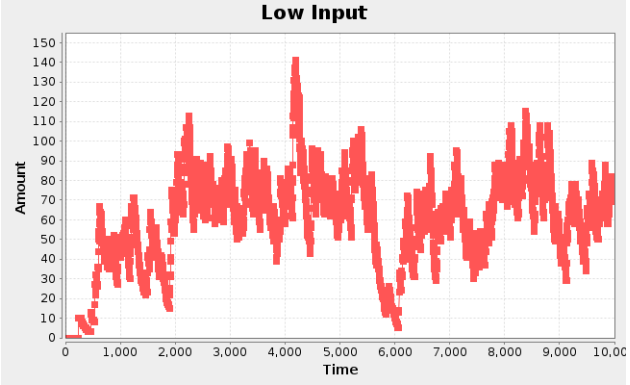
Potential Behaviors

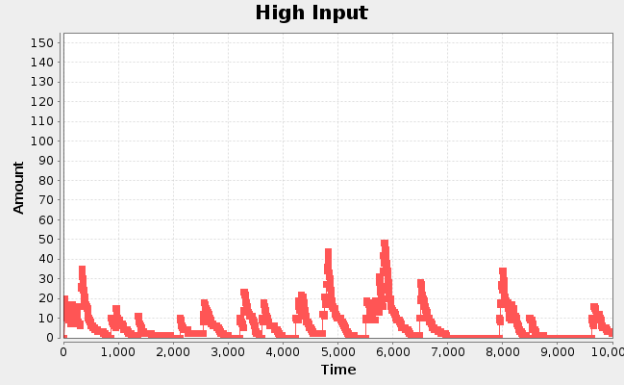


- Depending on which repressor modules are used, different behaviors can be achieved.

Signal Temporal Logic (STL)

- These behaviors must be encoded in STL.
- For instance, when “In” is greater than 100, “Out” always eventually rises above 50 within 1000 time units.
- Also, when “In” is 0, “Out” always remains below 50.



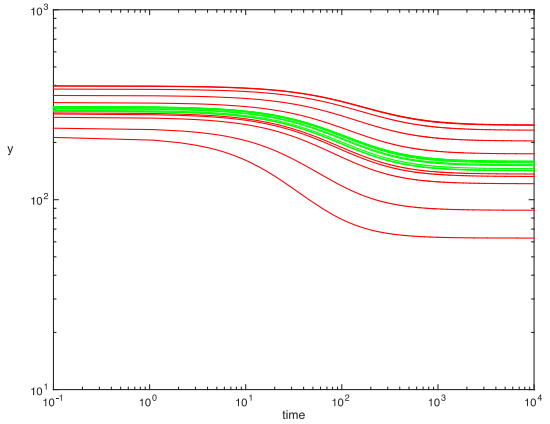


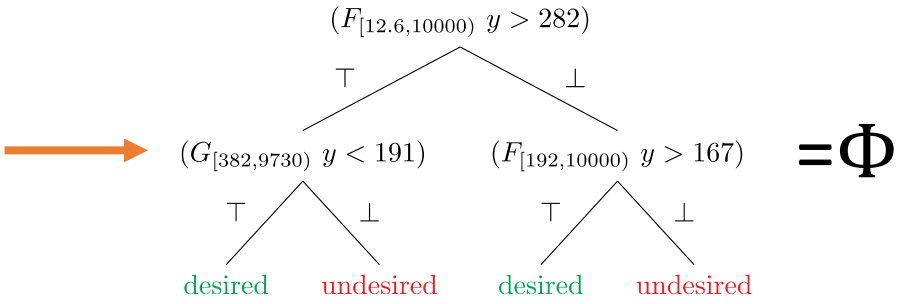
- This corresponds to the following STL:

$[G\ [0,10000)\ (In > 100)] \rightarrow [G\ [0,10000)\ F\ [0,1000)\ (Out > 50)] \wedge$
$[G\ [0,10000)\ (In \leq 0)] \rightarrow [G\ [0,10000)\ (Out \leq 50)]$

Characterization – Temporal Logic Inference (TLI)

- Using supervised learning, TLI[2] can be used to "learn" an STL formula from time series data.
- The current implementation of TLI works by finding optimal STL primitives and parameters:
  - $G_{[t_0,t_1]}\, x_i < k, \quad F_{[t_0,t_1]}\, x_i > k,$
  - $G_{[t_0,t_1]}\, x_i > k, \quad F_{[t_0,t_1]}\, x_i < k, \quad \ldots$
  - t_0, t_1, k found by simulated annealing

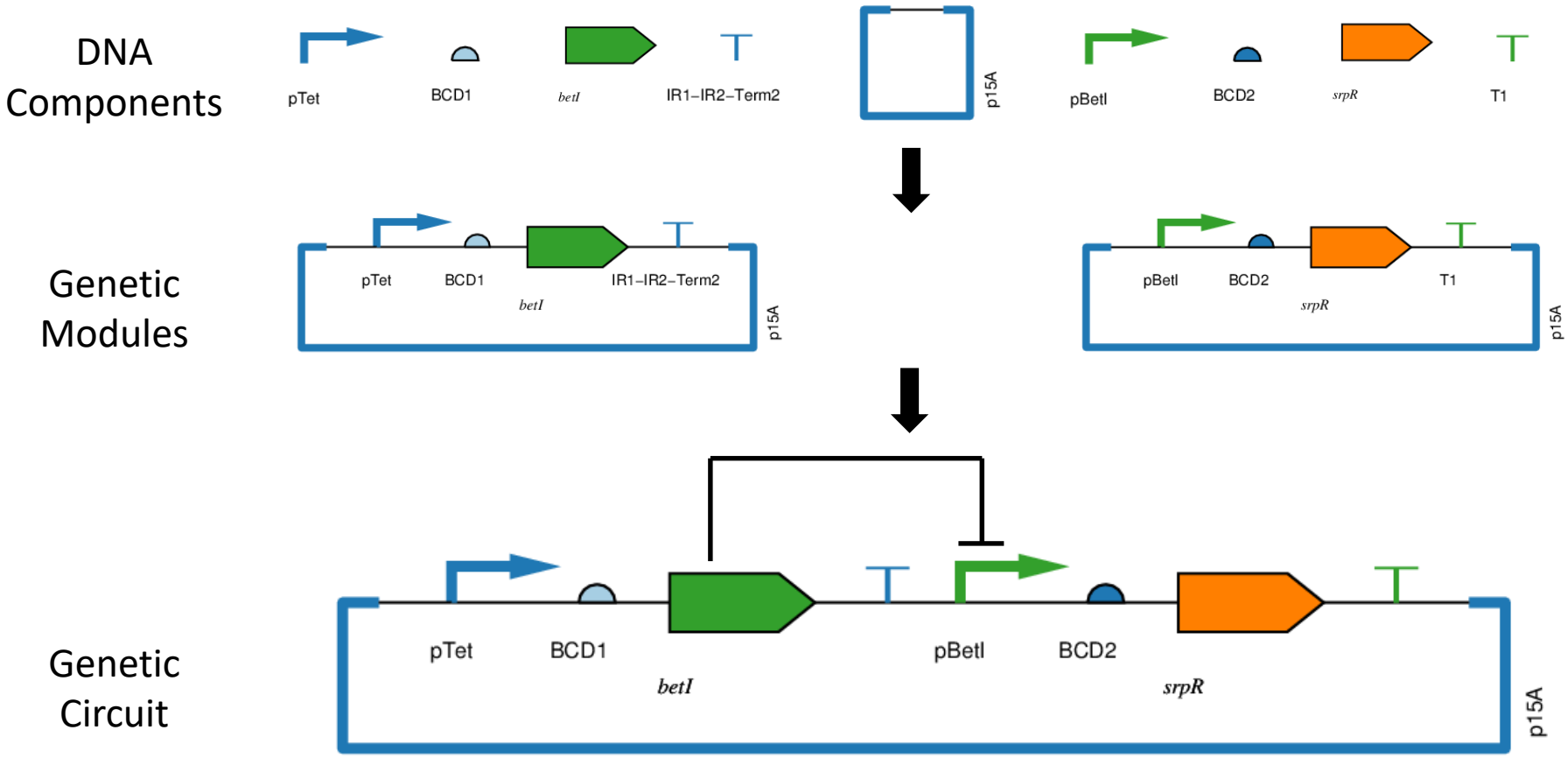




[2]G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A Decision Tree Approach to Data Classification using Signal Temporal Logic," presented at the Proceedings of the 19th International Conference on Hybrid systems: Computation and Control, New York, NY, USA, 2016, pp. 1–10.

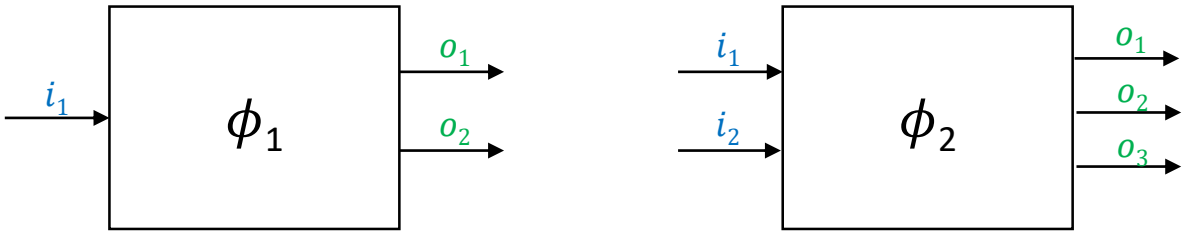
Composability in Synthetic Biology

- DNA segments representing genetic parts and modules can be composed to create genetic circuits.



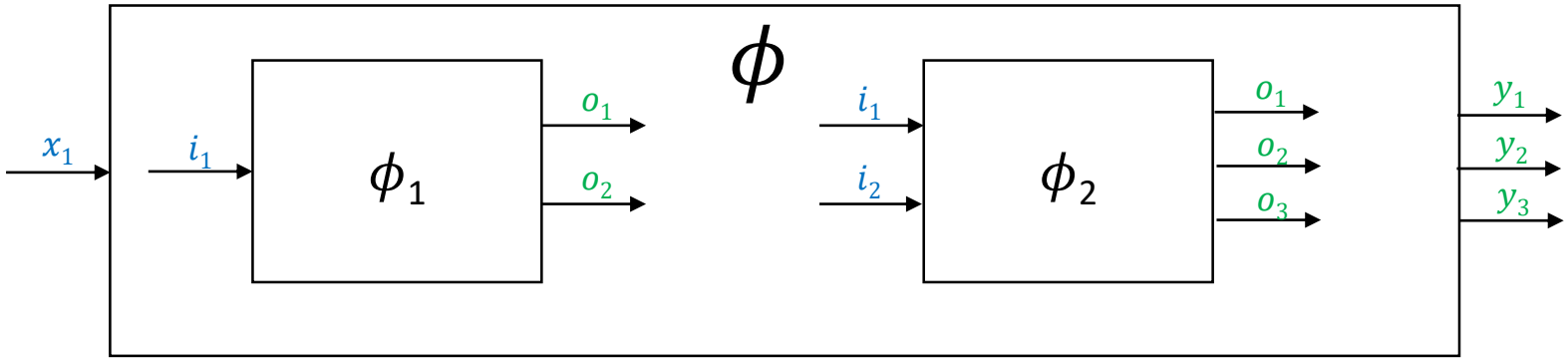
STLb

- STL with added functionality:
  - Concatenation (•) – allows one STL formula to connect to another in sequence.
  - Inputs – signals that are annotated as drivers of the formula.
  - Outputs – signals that are annotated as being produced by the formula.
  - Mapping – a collection of assignments among the inputs and outputs of STL formulae.
- For instance, ϕ is composed of ϕ_1 and ϕ_2:



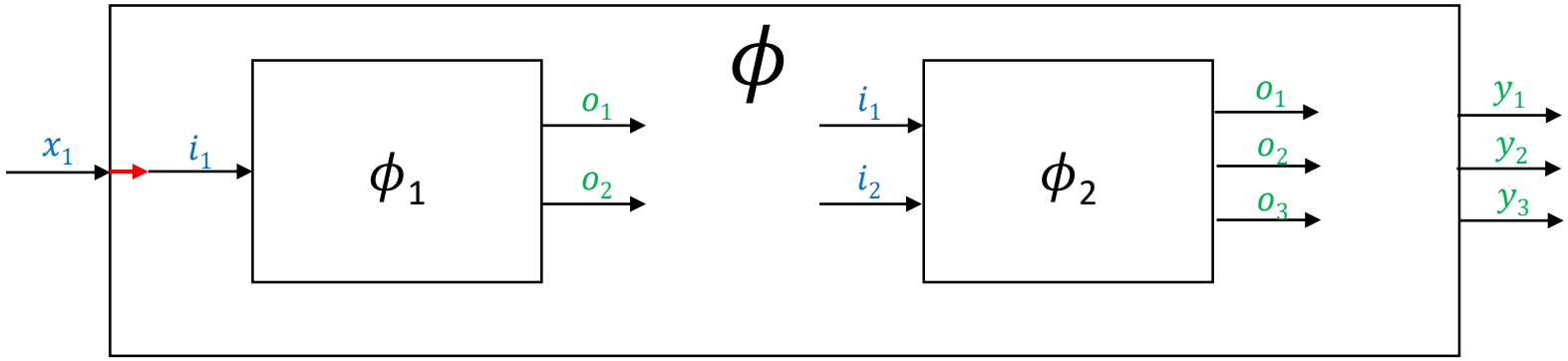
STLb

- STL with added functionality:
  - Concatenation (•) – allows one STL formula to connect to another in sequence.
  - Inputs – signals that are annotated as drivers of the formula.
  - Outputs – signals that are annotated as being produced by the formula.
  - Mapping – a collection of assignments among the inputs and outputs of STL formulae.
- For instance, ϕ is composed of ϕ_1 and ϕ_2:
  - Concatenation – $\phi(x_1, y_1, y_2, y_3) = \phi_1(i_1, o_1, o_2) \bullet \phi_2(i_1, i_2, o_1, o_2, o_3)$.



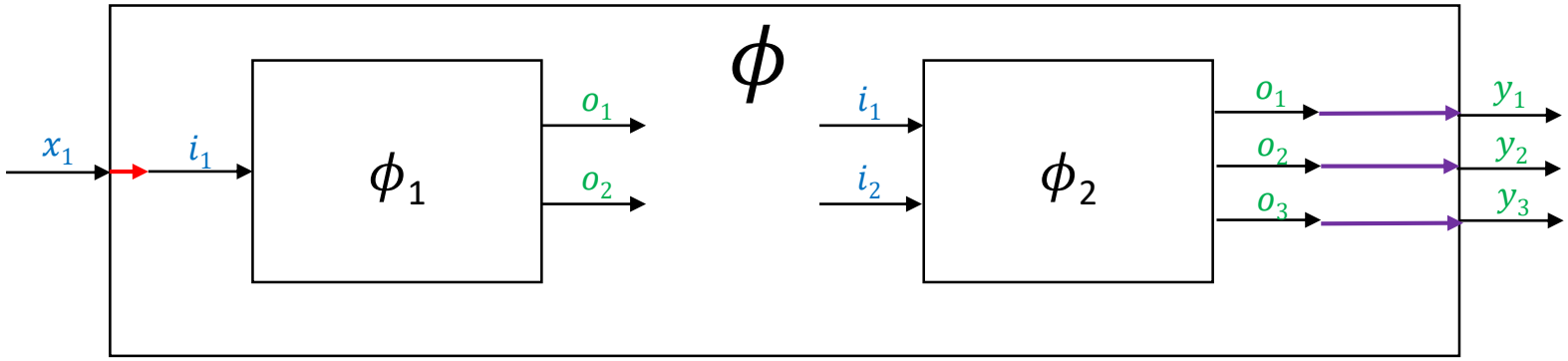
STLb

- STL with added functionality:
  - Concatenation (•) – allows one STL formula to connect to another in sequence.
  - Inputs – signals that are annotated as drivers of the formula.
  - Outputs – signals that are annotated as being produced by the formula.
  - Mapping – a collection of assignments among the inputs and outputs of STL formulae.
- For instance, ϕ is composed of ϕ_1 and ϕ_2:
  - Concatenation – $\phi(x_1, y_1, y_2, y_3) = \phi_1(i_1, o_1, o_2) \bullet \phi_2(i_1, i_2, o_1, o_2, o_3)$.
  - Input mapping – $(\phi: x_1 = \phi_1: i_1)$.



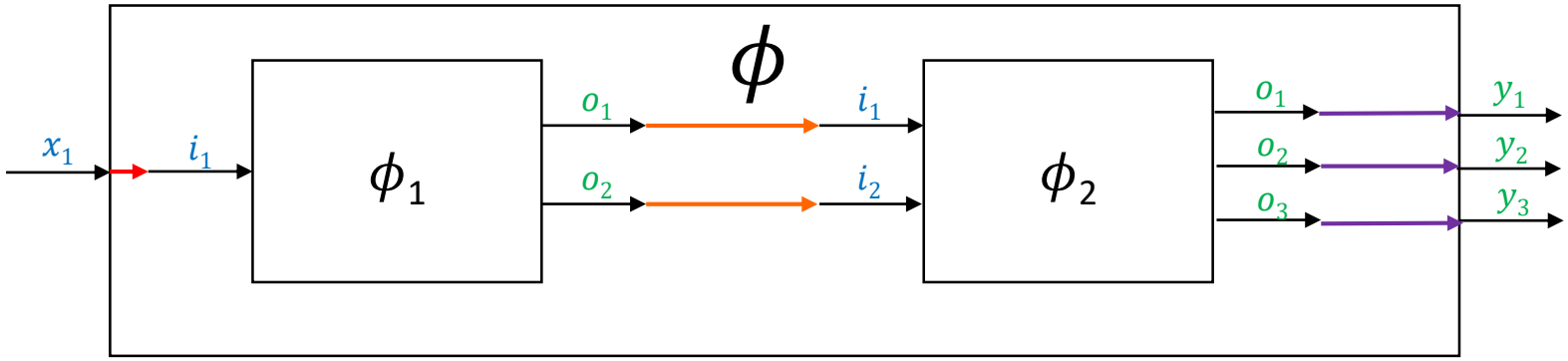
STLb

- STL with added functionality:
  - Concatenation (•) – allows one STL formula to connect to another in sequence.
  - Inputs – signals that are annotated as drivers of the formula.
  - Outputs – signals that are annotated as being produced by the formula.
  - Mapping – a collection of assignments among the inputs and outputs of STL formulae.
- For instance, ϕ is composed of ϕ_1 and ϕ_2:
  - Concatenation – $\phi(x_1, y_1, y_2, y_3) = \phi_1(i_1, o_1, o_2) \bullet \phi_2(i_1, i_2, o_1, o_2, o_3)$.
  - Input mapping – $(\phi: x_1 = \phi_1: i_1)$.
  - Output mapping – $(\phi: y_1 = \phi_2: o_1) \wedge (\phi: y_2 = \phi_2: o_2) \wedge (\phi: y_3 = \phi_2: o_3)$.



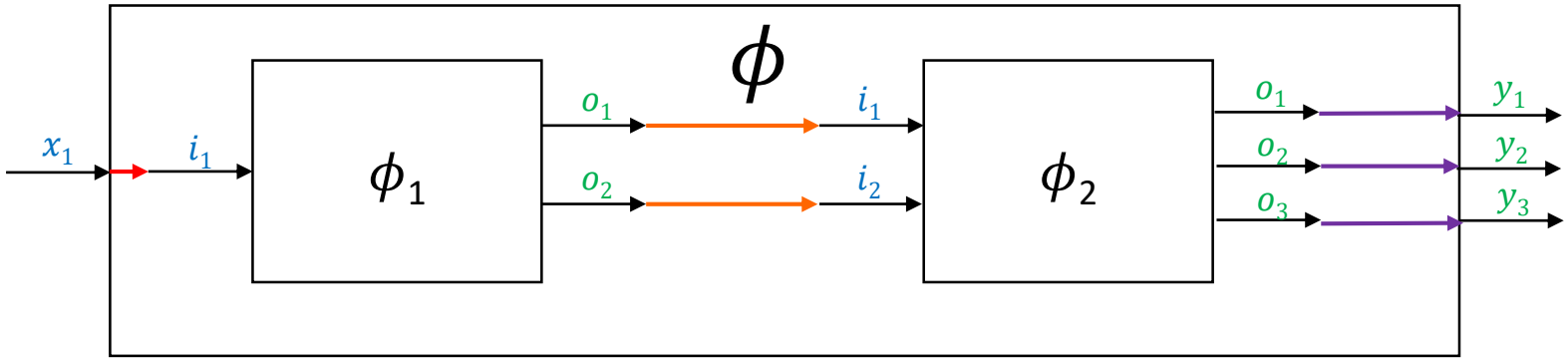
STLb

- STL with added functionality:
  - Concatenation (•) – allows one STL formula to connect to another in sequence.
  - Inputs – signals that are annotated as drivers of the formula.
  - Outputs – signals that are annotated as being produced by the formula.
  - Mapping – a collection of assignments among the inputs and outputs of STL formulae.
- For instance, ϕ is composed of ϕ_1 and ϕ_2:
  - Concatenation – $\phi(x_1, y_1, y_2, y_3) = \phi_1(i_1, o_1, o_2) \bullet \phi_2(i_1, i_2, o_1, o_2, o_3)$.
  - Input mapping – $(\phi: x_1 = \phi_1: i_1)$.
  - Output mapping – $(\phi: y_1 = \phi_2: o_1) \wedge (\phi: y_2 = \phi_2: o_2) \wedge (\phi: y_3 = \phi_2: o_3)$.
  - Internal mapping – $(\phi_1: o_1 = \phi_2: i_1) \wedge (\phi_1: o_2 = \phi_2: i_2)$.



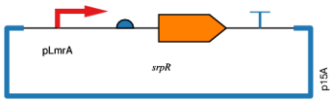
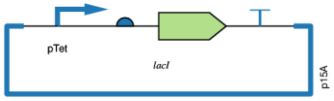
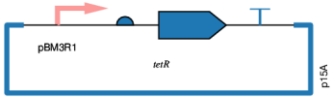
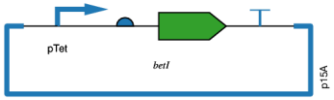
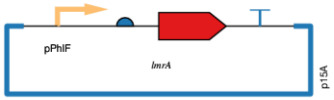
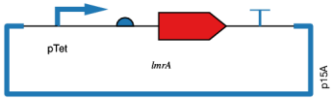
STLb

- STL with added functionality:
  - Concatenation (•) – allows one STL formula to connect to another in sequence.
  - Inputs – signals that are annotated as drivers of the formula.
  - Outputs – signals that are annotated as being produced by the formula.
  - Mapping – a collection of assignments among the inputs and outputs of STL formulae.
- For instance, ϕ is composed of ϕ_1 and ϕ_2:
  - Concatenation – $\phi(x_1, y_1, y_2, y_3) = \phi_1(i_1, o_1, o_2) \bullet \phi_2(i_1, i_2, o_1, o_2, o_3)$.
  - Input mapping – $(\phi: x_1 = \phi_1: i_1)$.
  - Output mapping – $(\phi: y_1 = \phi_2: o_1) \wedge (\phi: y_2 = \phi_2: o_2) \wedge (\phi: y_3 = \phi_2: o_3)$.
  - Internal mapping – $(\phi_1: o_1 = \phi_2: i_1) \wedge (\phi_1: o_2 = \phi_2: i_2)$.

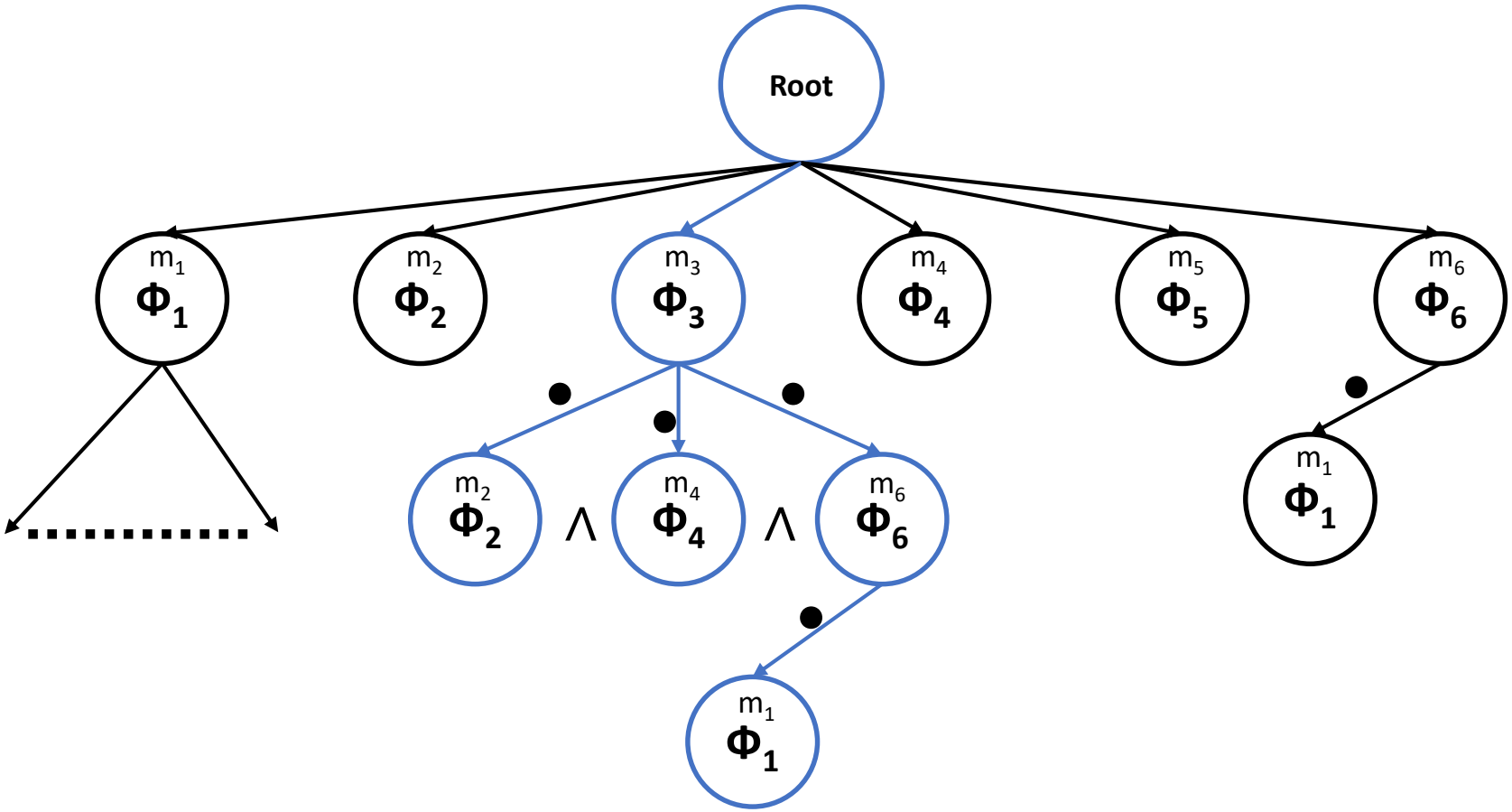


Note: The mapping can be applied to other STL operators, not just concatenation.

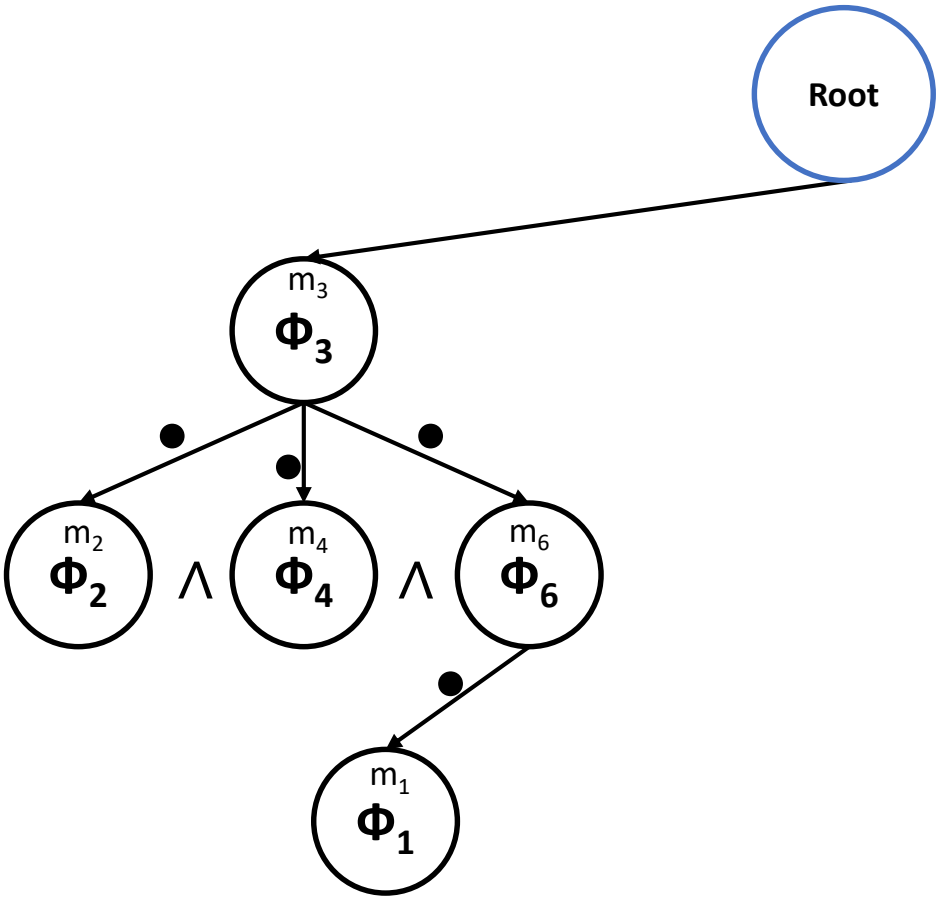
Library

Genetic Modules	Name	STL Formula
	m_1	Φ_1
	m_2	Φ_2
	m_3	Φ_3
	m_4	Φ_4
	m_5	Φ_5
	m_6	Φ_6

Design Space Exploration



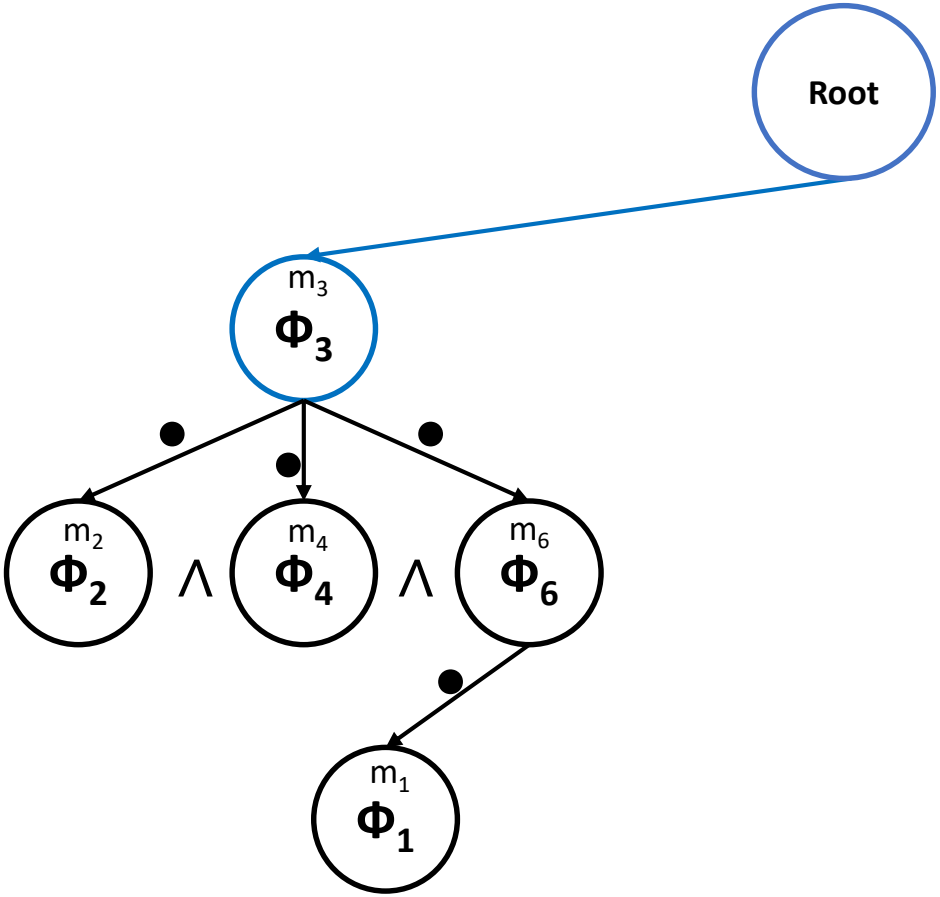
Design Space Exploration



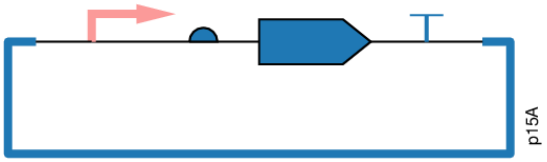
Design

STL Formula

Design Space Exploration



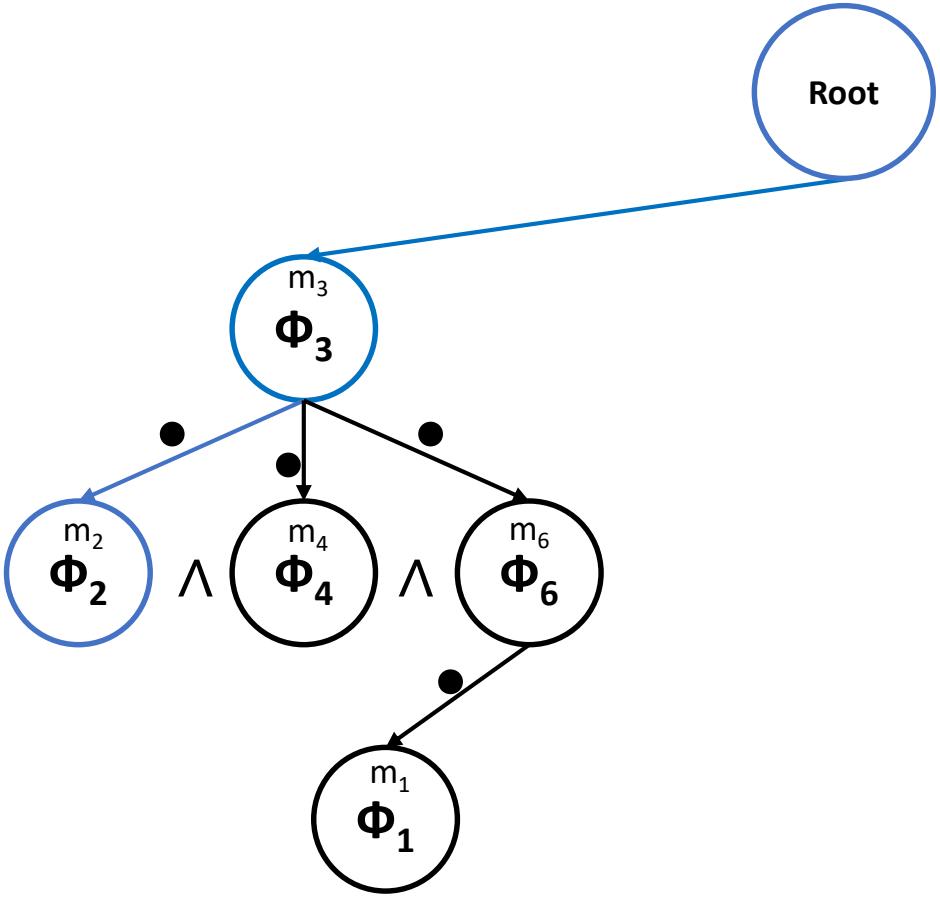
Design



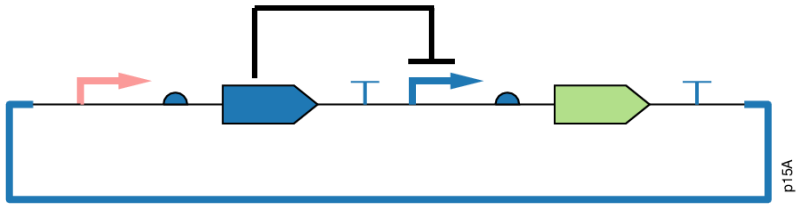
STL Formula

Φ_3

Design Space Exploration



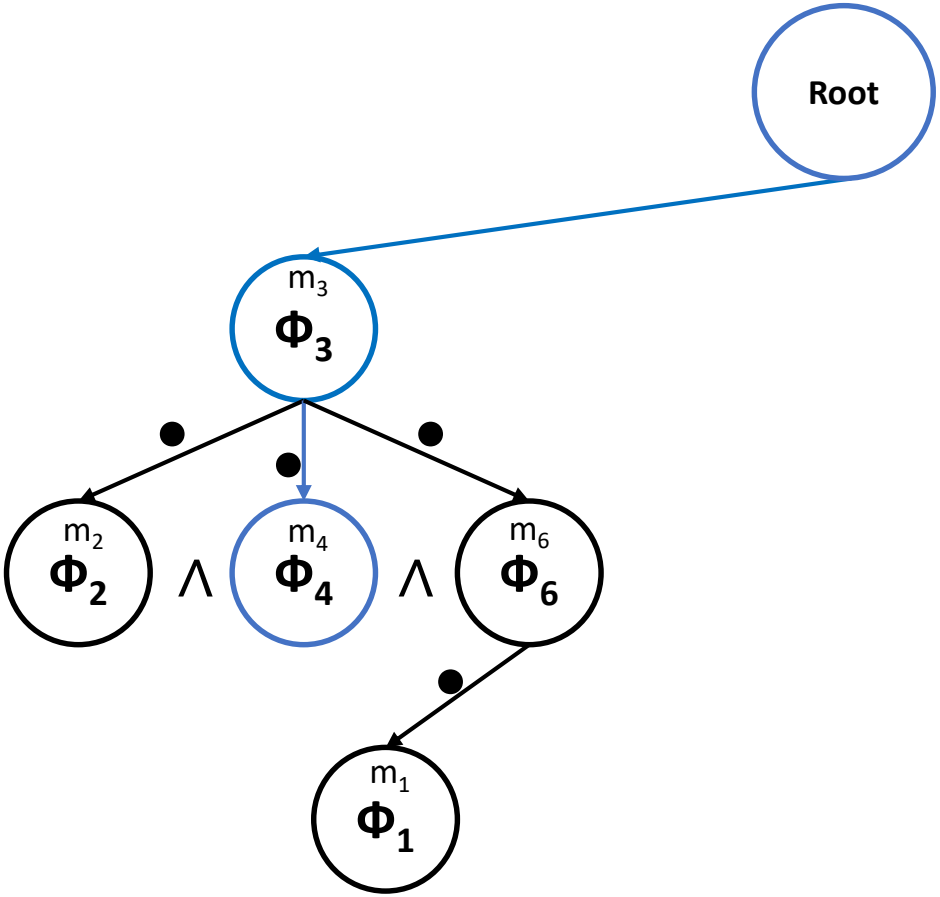
Design



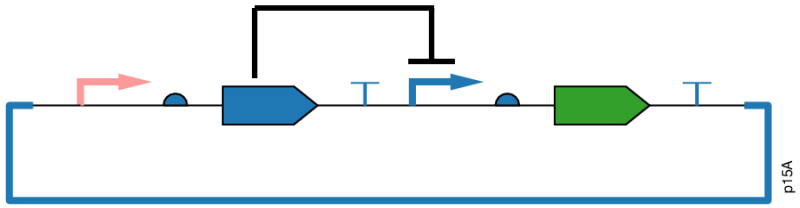
STL Formula

$\Phi_3 \bullet \Phi_2$

Design Space Exploration



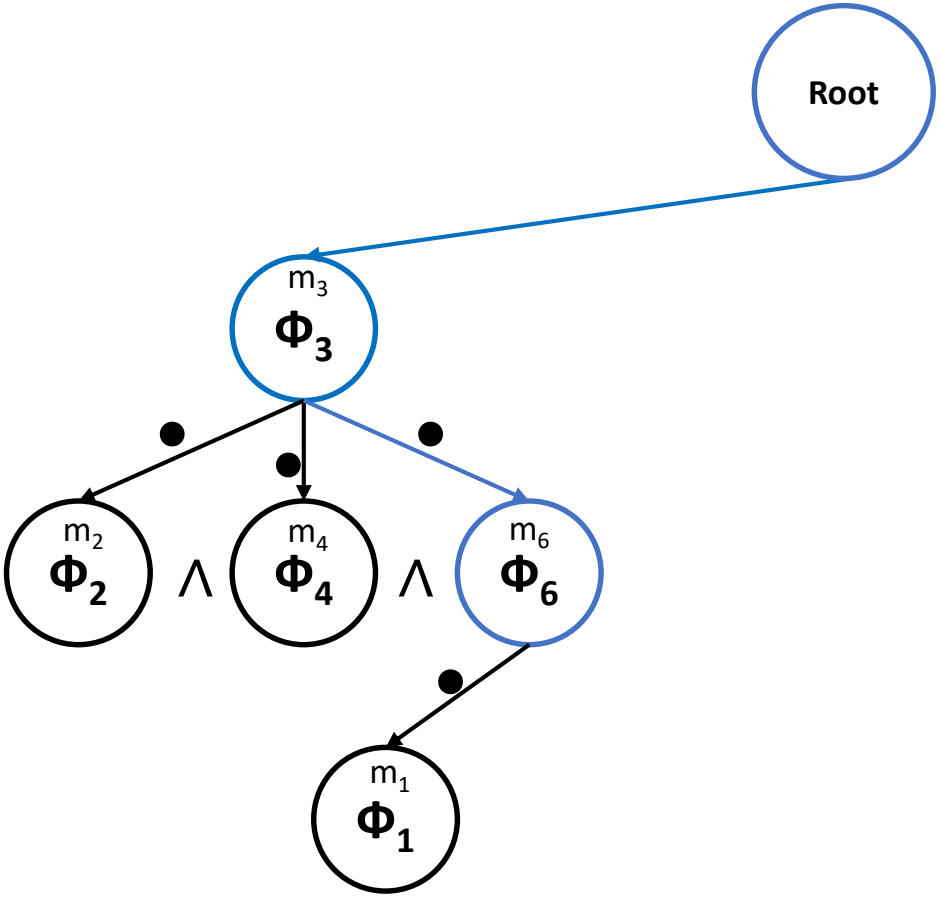
Design



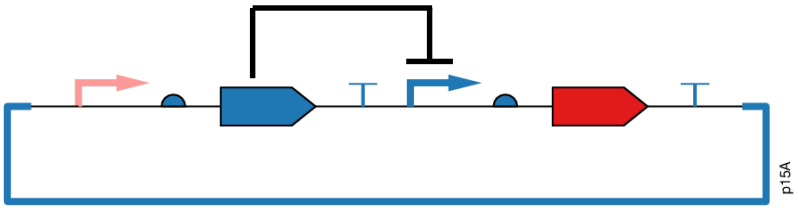
STL Formula

$\Phi_3 \bullet \Phi_4$

Design Space Exploration



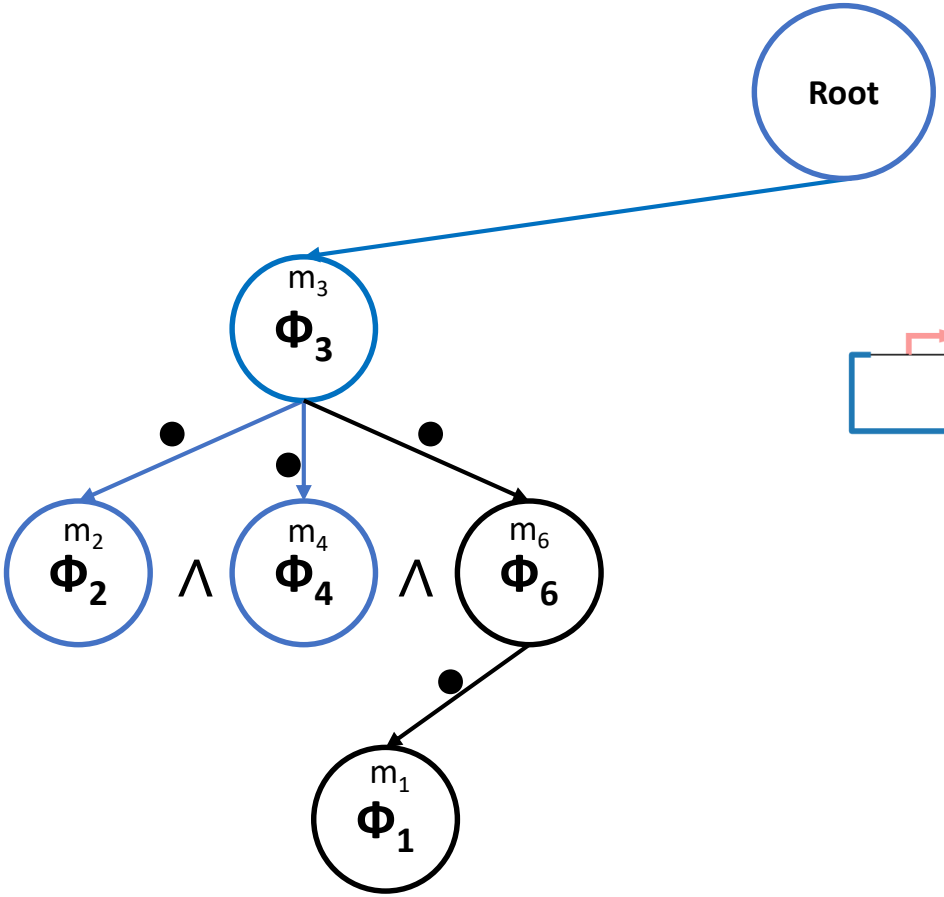
Design



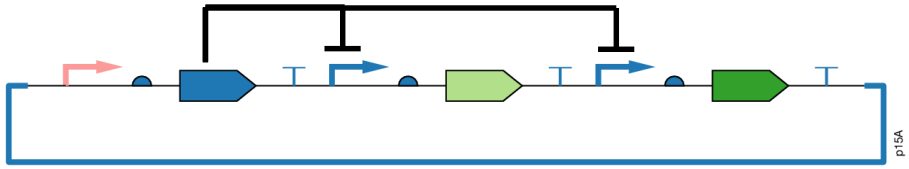
STL Formula

$\Phi_3 \bullet \Phi_6$

Design Space Exploration



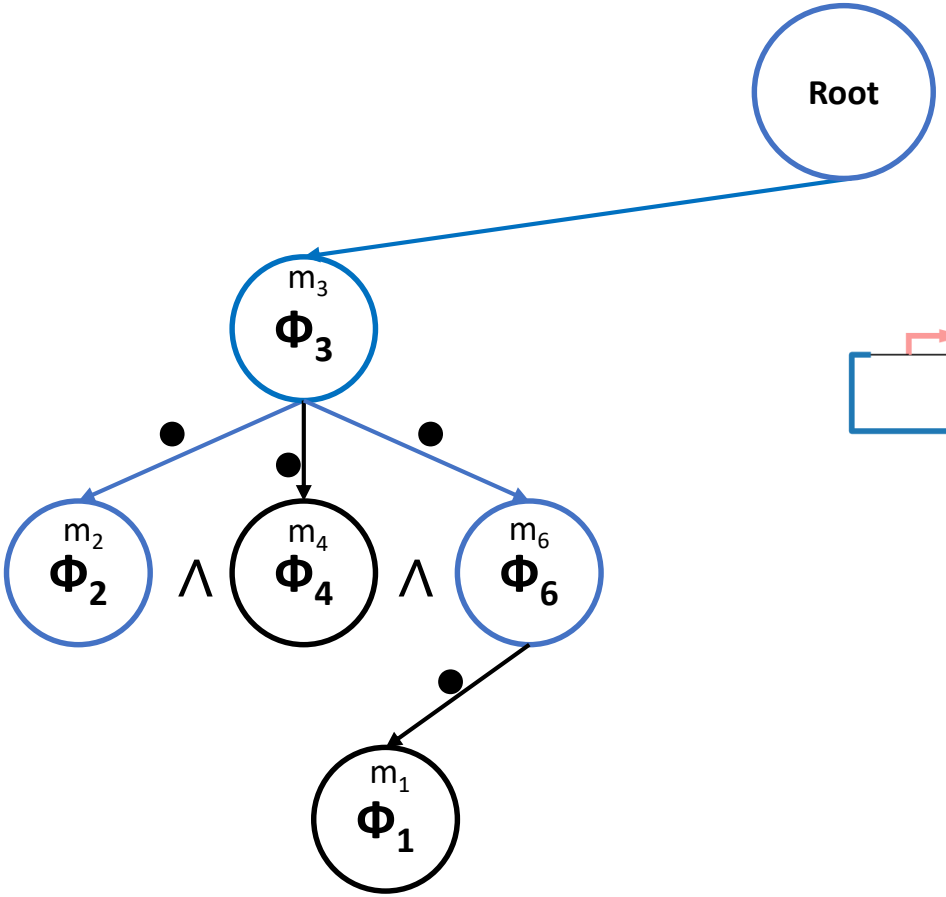
Design



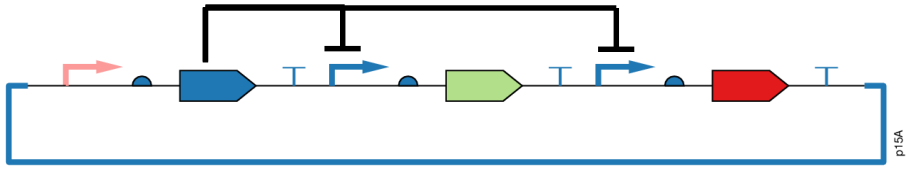
STL Formula

$$\Phi_3 \bullet (\Phi_2 \wedge \Phi_4)$$

Design Space Exploration



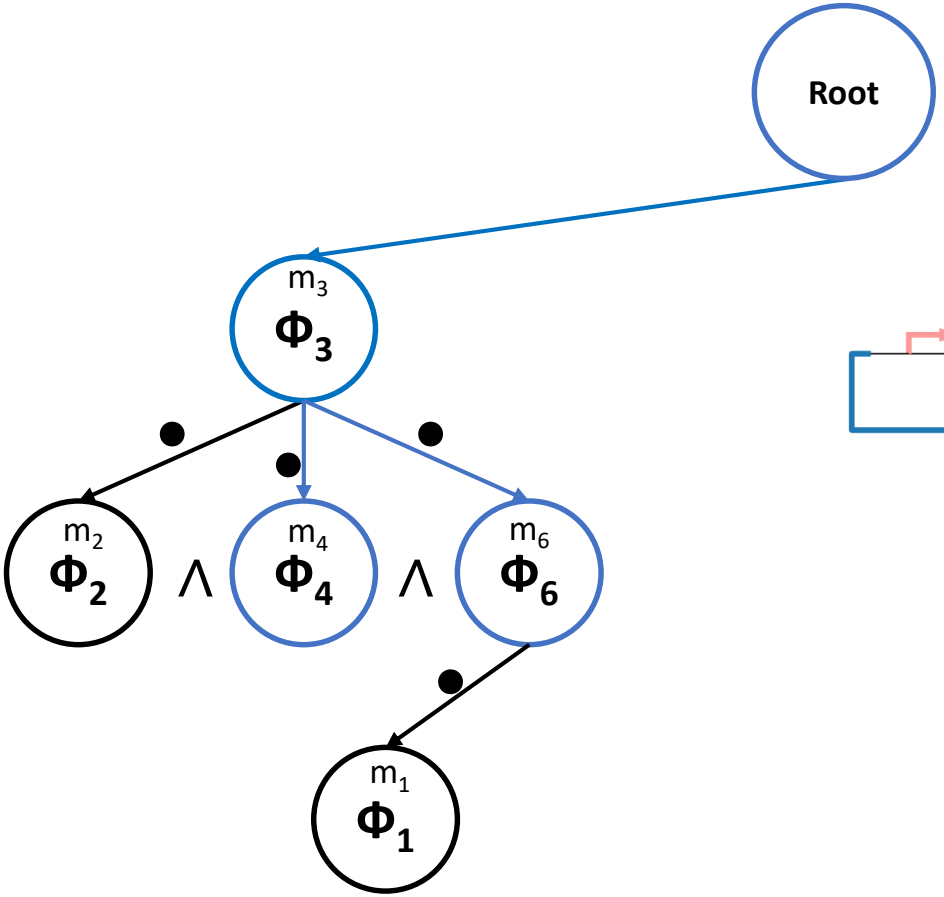
Design



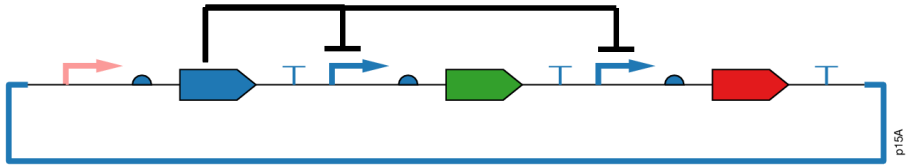
STL Formula

$$\Phi_3 \bullet (\Phi_2 \wedge \Phi_6)$$

Design Space Exploration



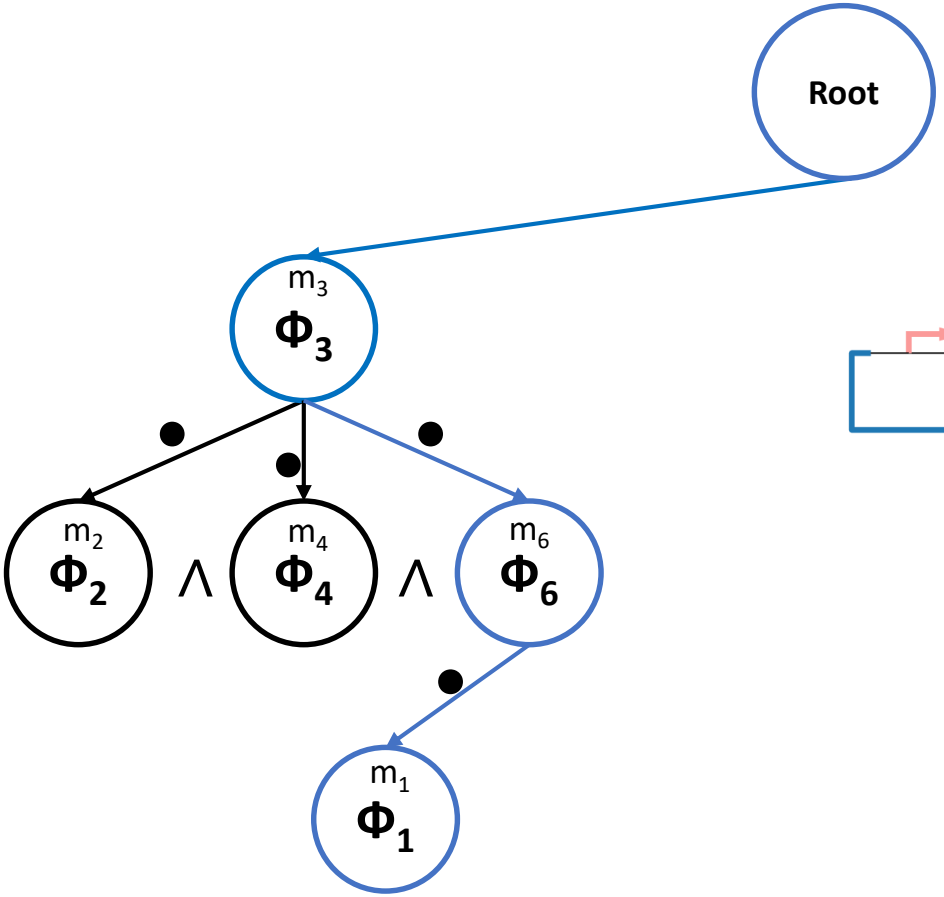
Design



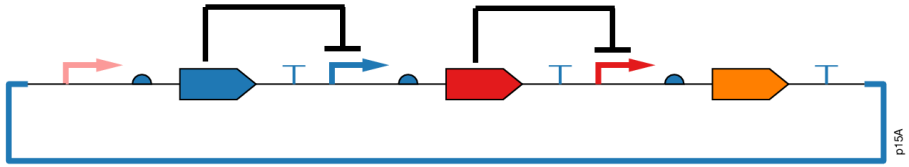
STL Formula

$$\Phi_3 \bullet (\Phi_4 \wedge \Phi_6)$$

Design Space Exploration



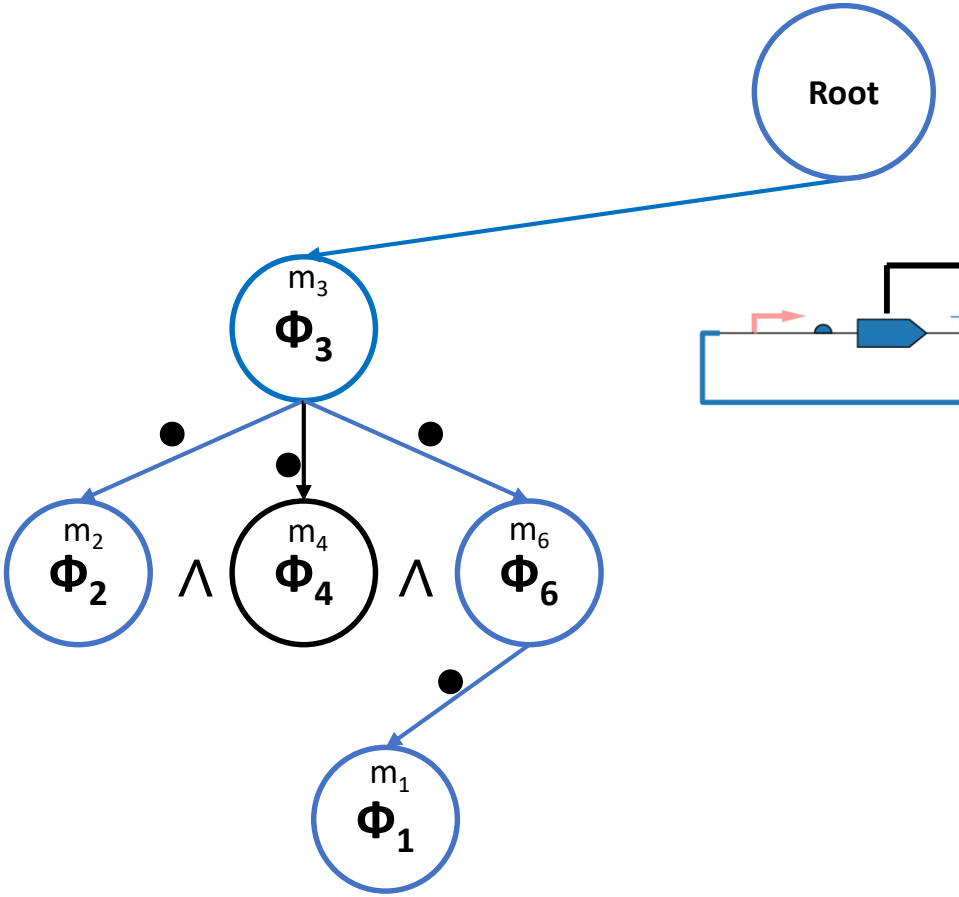
Design



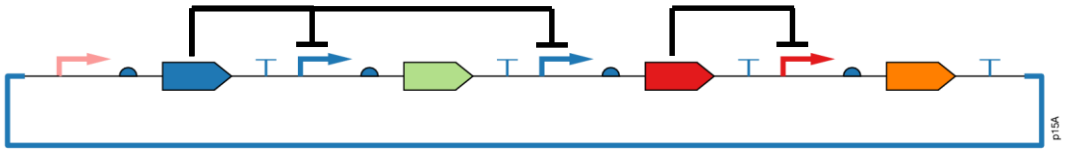
STL Formula

$$\Phi_3 \bullet (\Phi_6 \bullet \Phi_1)$$

Design Space Exploration



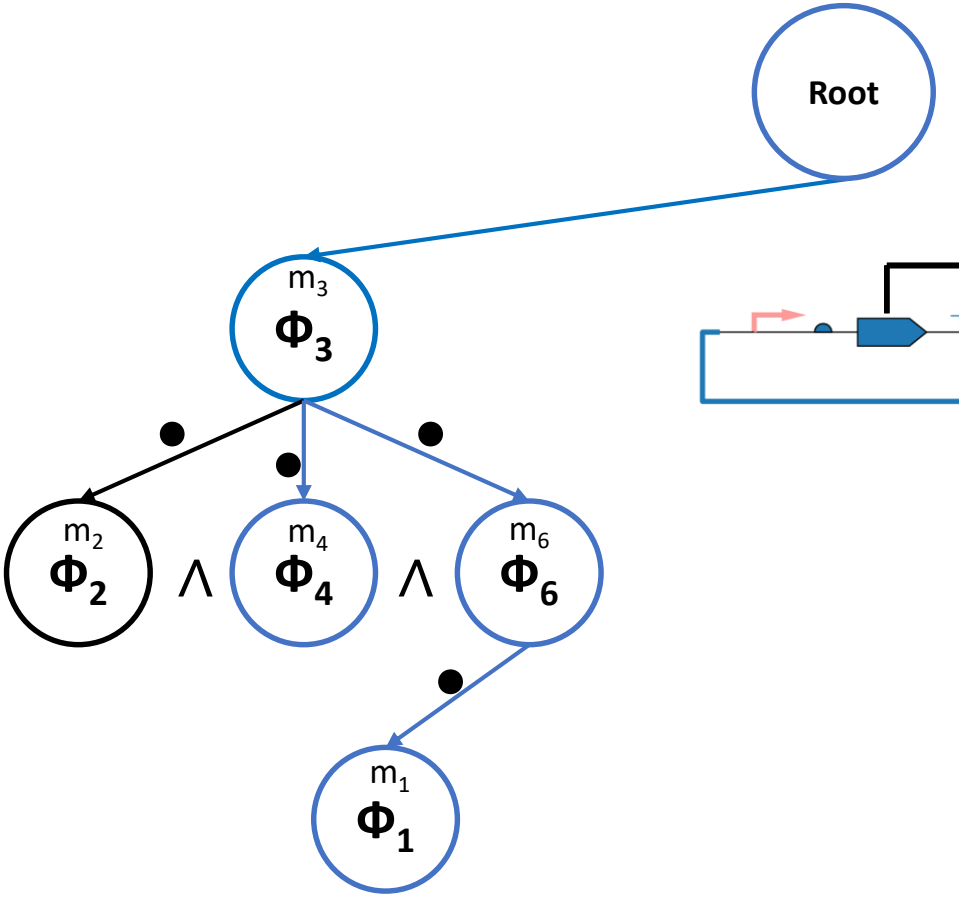
Design



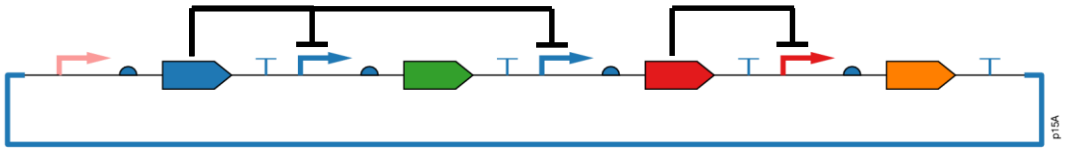
STL Formula

$$\Phi_3 \bullet (\Phi_2 \wedge (\Phi_6 \bullet \Phi_1))$$

Design Space Exploration



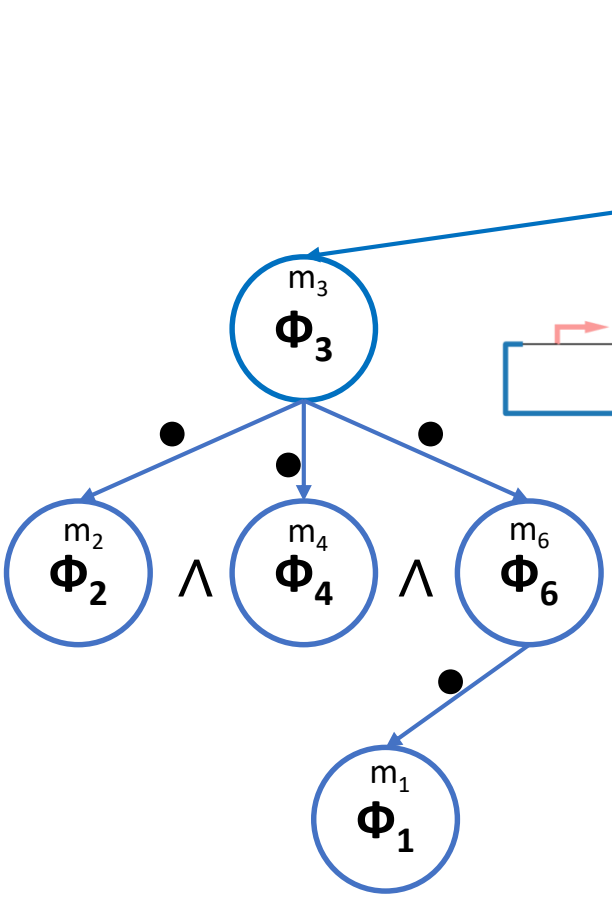
Design



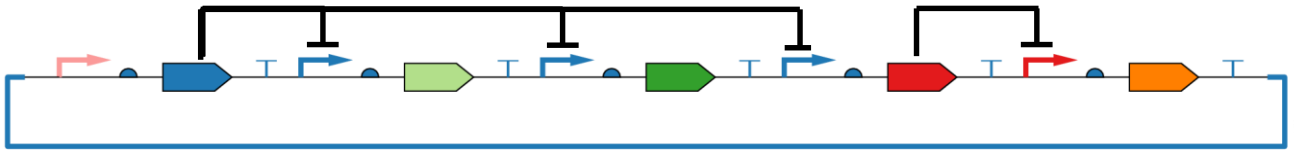
STL Formula

$$\Phi_3 \bullet (\Phi_4 \wedge (\Phi_6 \bullet \Phi_1))$$

Design Space Exploration



Design

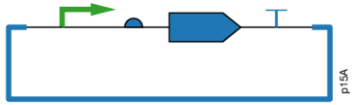


STL Formula

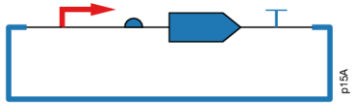
$$\Phi_3 \bullet (\Phi_2 \wedge \Phi_4 \wedge (\Phi_6 \bullet \Phi_1))$$

Constraint Pruning

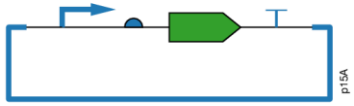
Library



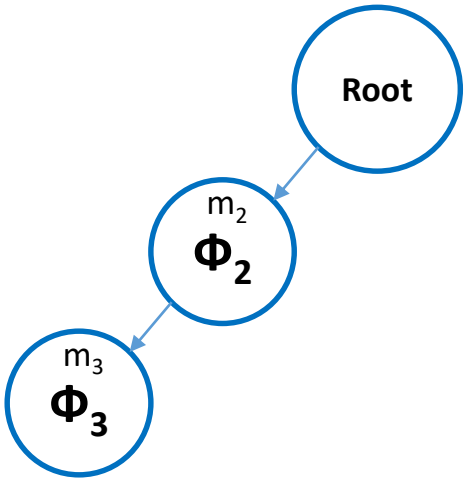
m_1 Φ_1

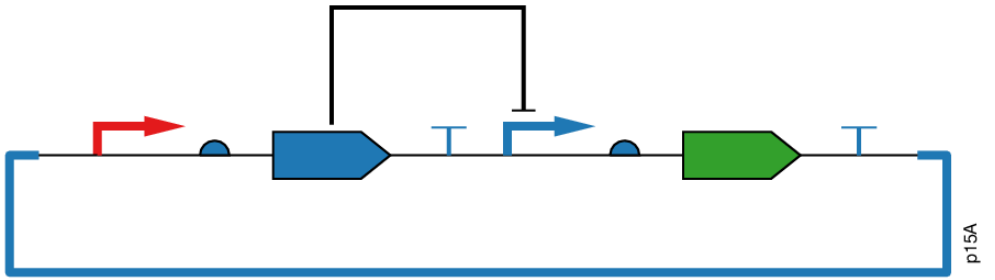


m_2 Φ_2



m_3 Φ_3

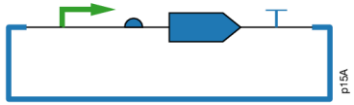




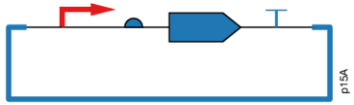
$\Phi_2 \bullet \Phi_3$

Constraint Pruning

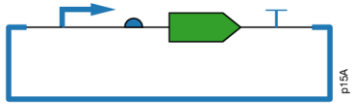
Library



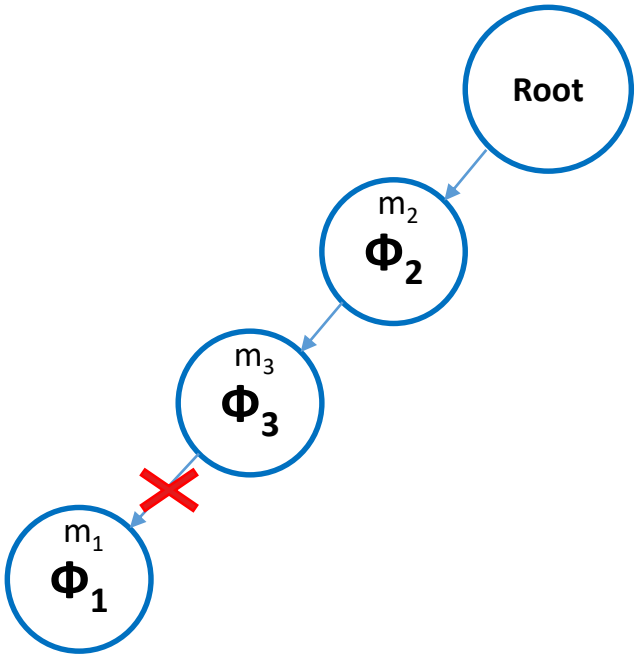
m_1 Φ_1



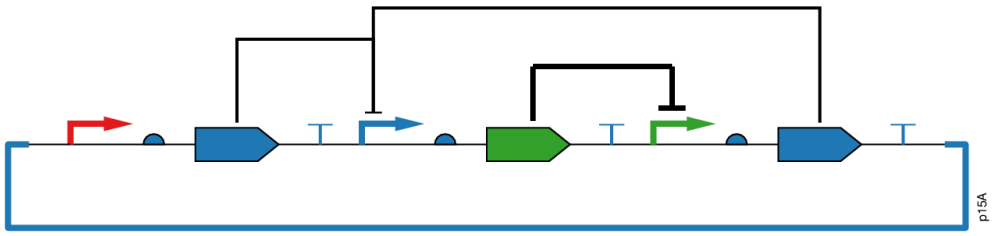
m_2 Φ_2



m_3 Φ_3

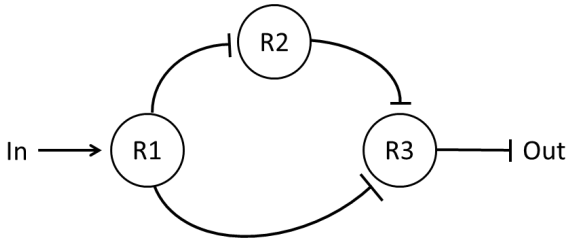


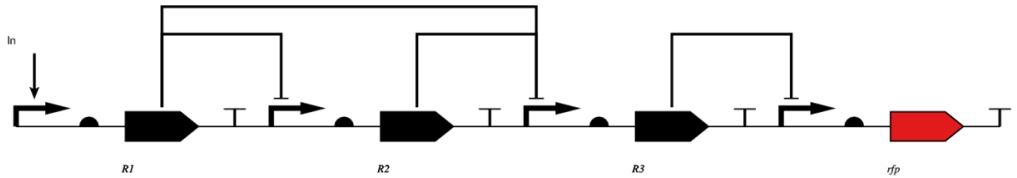
Cross Talk

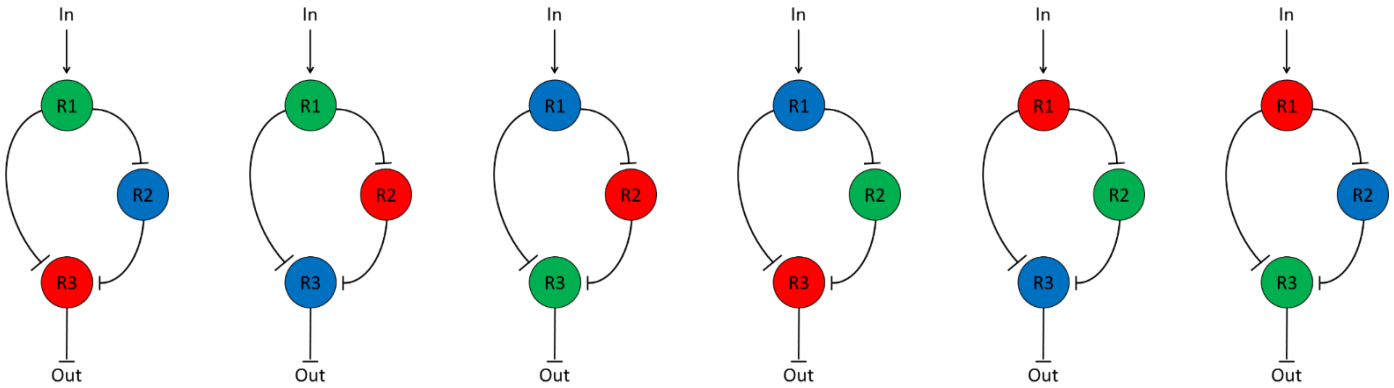


$\Phi_2 \bullet \Phi_3 \bullet \Phi_3$

Possible RIFFL Circuit Designs







Future Work

- Currently, TLI requires both desirable and undesirable traces.
  - We are working on a method that only requires desirable traces.
- We are adding constraints to help prune the potential design space of the composed genetic circuits.
- We are currently testing these methods on mammalian and bacterial synthetic biology examples.

Acknowledgements

Prashant Vaidyanathan

Cristian-Ioan Vasile

Rachael Ivison

Junmin Wang

Calin Belta

Douglas Densmore

This work is supported by the National Science Foundation under grant CPS Frontier 1446607.