Eugene's Enriched Set of Features to Design Synthetic Biological Devices

Haiyao Huang, Ernst Oberortner,
Douglas Densmore and Allan Kuchinsky.

IWBDA

3 June 2012

Overview

- Eugene: specification language for synthetic biology
- Inputs: imported parts, devices, and specification/constraints
- Output: devices satisfying specifications
- Highlights:
  - Human readable rules based language
  - Integrated with Clotho and SBOL

Eugene's Features

Eugene allows users to:

- Specify synthetic biological components at various abstraction levels
- Assign properties to components
- Define constraints (i.e., rules) on the components' compositions

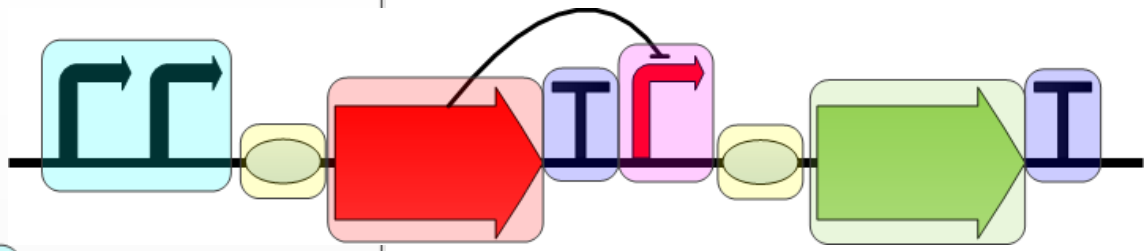
- Create and invoke user-defined and reusable functions
- Manage the control-flow of design synthesis with loops and conditional statements
- Generate novel biological devices automatically via built-in functions

Case Studies

- NOR gate
  - Highlight new design exploration features in Eugene with built-in functions.
    - permute()
    - product()
- Toggle switch
  - Highlights using loops and conditionals, and defining functions in Eugene
    - FOR/WHILE/DO-WHILE
    - IF-ELSE
    - Function definitions

NOR_GATE.EUG

```
Property name(txt);
Property sequence(txt);
Property represses(txt);
Property repressedBy(txt);

Part InduciblePromoter(name, sequence);
Part RBS(name, sequence);
Part Repressor(name, sequence, represses);
Part RepressiblePromoter(name, sequence, repressedBy);
Part Reporter(name, sequence);
Part Terminator(name, sequence);

Device AbstractNOR( InduciblePromoter, InduciblePromoter,
RBS, Repressor, Terminator, RepressiblePromoter, RBS,
Reporter, Terminator);
```



Declaring Part Types

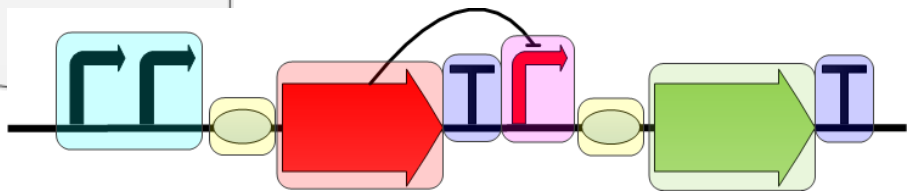
Declaring device

NOR_GATE.EUG

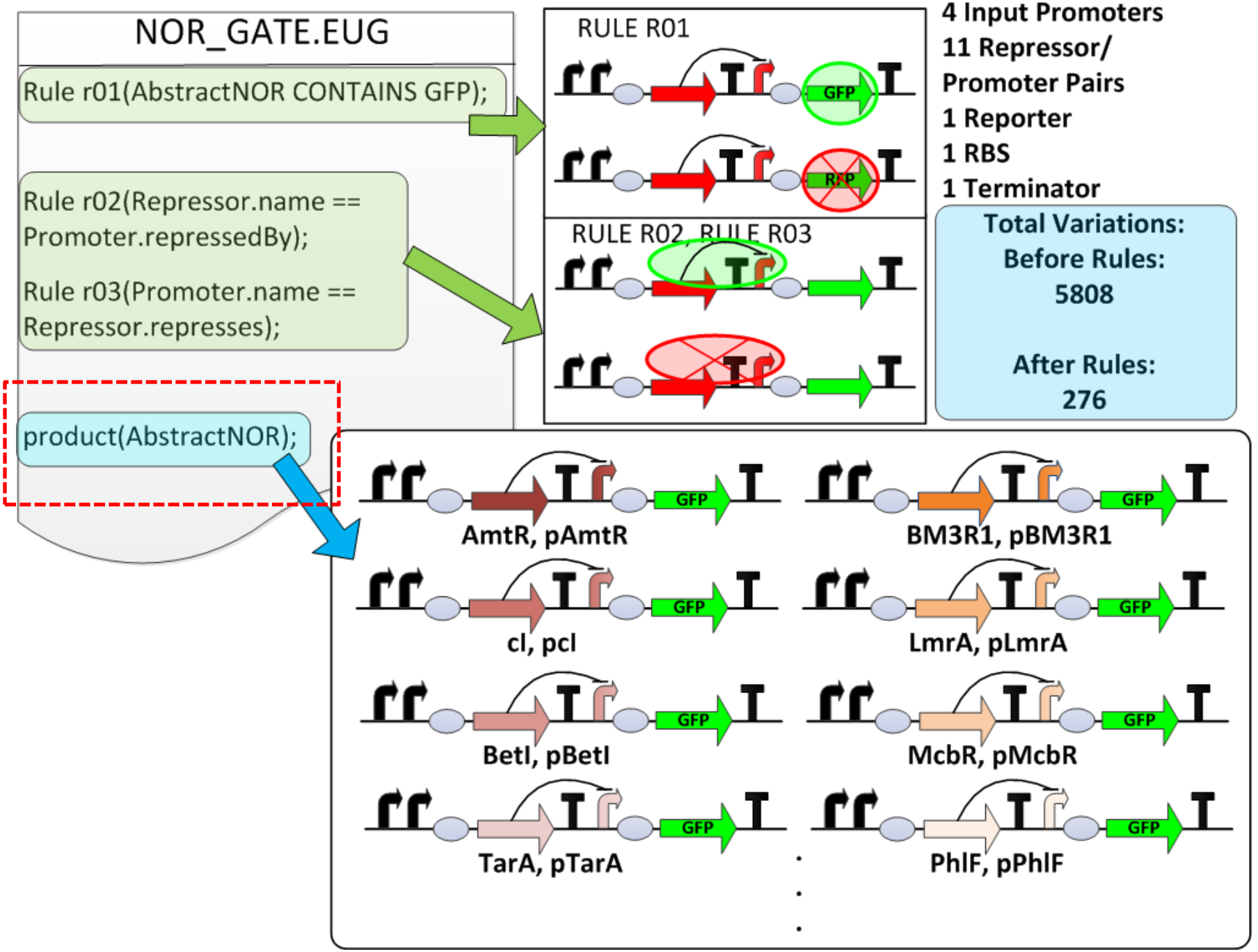
```
Terminator BBa_B0015("BBa_B0015", "ccaggcatca...cctttctgcgtttata");

RBS BBa_B0034("BBa_B0034", "ccaggcatca...cctttctgcgtttata");

Promoter pA("pA", "cgacgtacggtg...tatagataatgctagc");
Promoter pB("pB", "gaaatctttctgaccctgc...accgttgaaccgagcaccgt");
Promoter pC("pC", "ggtaaaaatccg...cgtgaagaaattctggatg");
Promoter pD("pD", "caatgtttt...tcgtgatctgg");

RepressiblePromoter pAmtR("pAmtR", "cgacgtacggtg...tatagataatgctagc", "AmtR");
RepressiblePromoter pBetI("pBetI", "ataccgtcattc...ataatgctagc", "BetI");
RepressiblePromoter pBM3R1("pBM3R1", "cgacgtacggtg...gataatgctagc", "BM3R1");
                                .
                                .
                                .

Repressor AmtR("AmtR", "ctatggactat...gggcccataccc", "pAmtR");
Repressor BetI("BetI", "gctacgacttgc...taaggcgagccc", "pBetI");
Repressor BM3R1("BM3R1", "ctatggactatg...gggcccataccc", "pBM3R1");
                                .
                                .
                                .

Reporter GFP("GFP", "atgcgtaaagga...ttagtagcttaa");
Reporter RFP("RFP", "atgcgtaaagga...ttagtagcttaa");
Reporter YFP("YFP", "atgcgtaaagga...ttagtagcttaa");
```

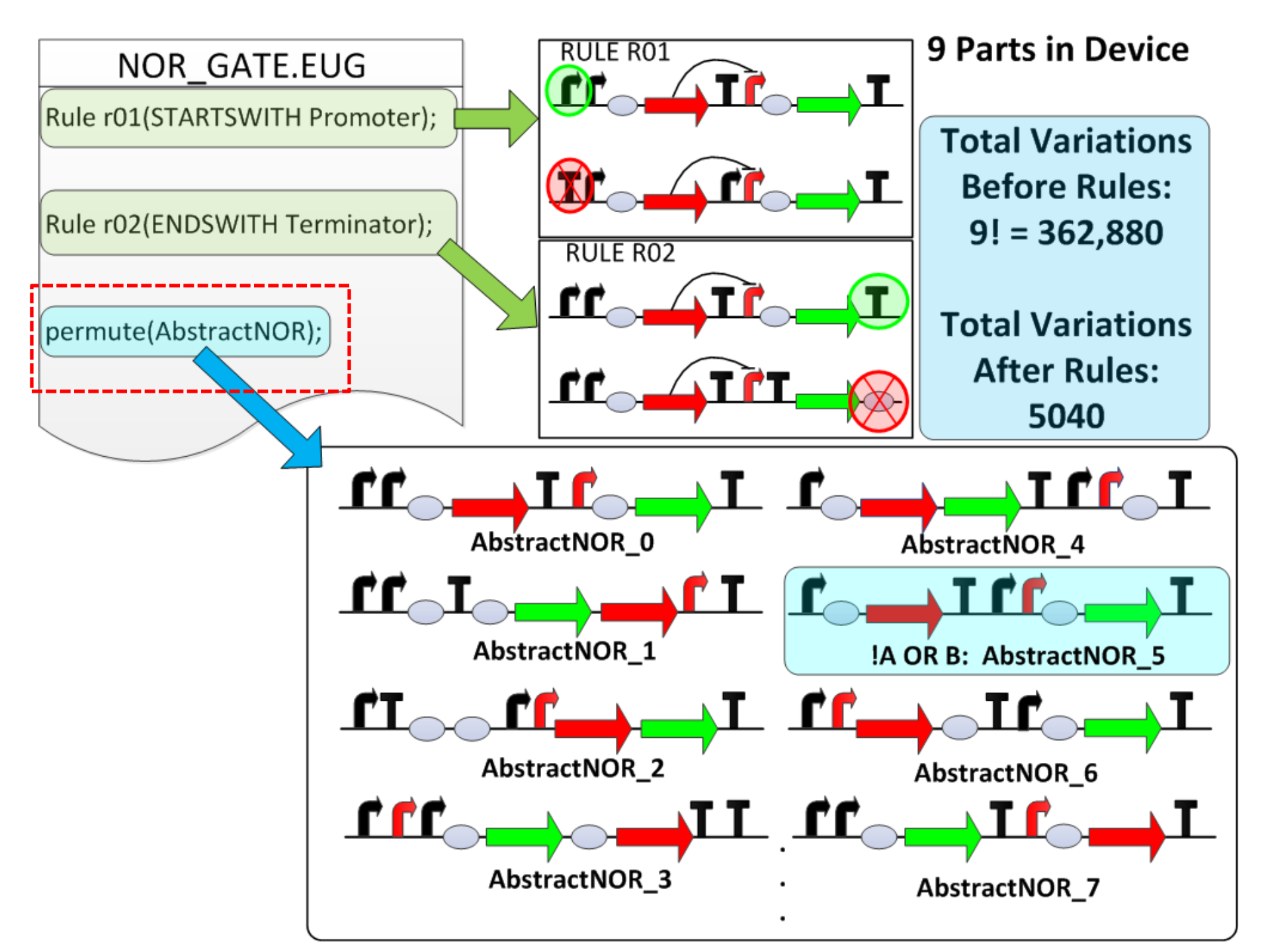
Can import parts from preexisting files or from the Registry of Standard Biological Parts

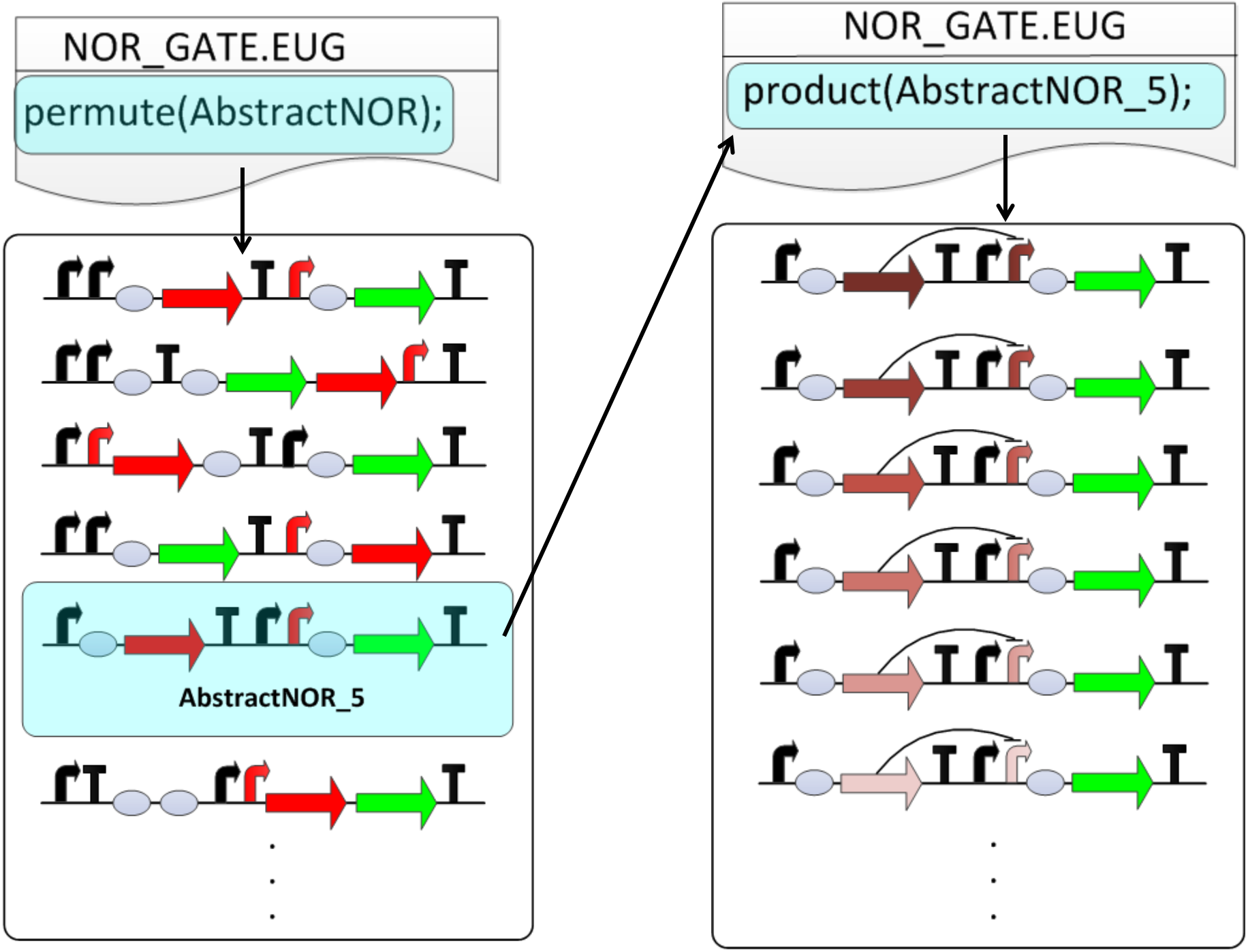
Assign concrete values to the Part Type's properties



Create collection of specific parts to use in design process

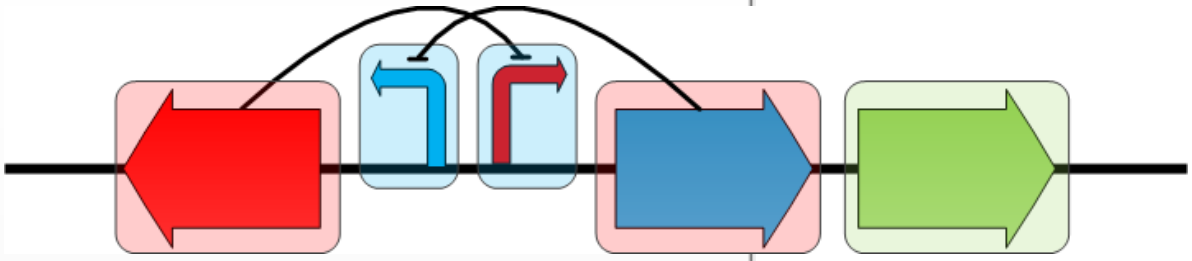

NOR_GATE.EUG
Rule r01(AbstractNOR CONTAINS GFP);
Rule r02(Repressor.name == Promoter.repressedBy);
Rule r03(Promoter.name == Repressor.represses);
product(AbstractNOR);
RULE R01
GFP
RFP
RULE R02, RULE R03
4 Input Promoters
11 Repressor/ Promoter Pairs
1 Reporter
1 RBS
1 Terminator
Total Variations:
Before Rules:
5808
After Rules:
276
AmtR, pAmtR
BM3R1, pBM3R1
cI, pcI
LmrA, pLmrA
BetI, pBetI
McbR, pMcbR
TarA, pTarA
PhlF, pPhlF


NOR_GATE.EUG
Rule r01(STARTSWITH Promoter);
Rule r02(ENDSWITH Terminator);
permute(AbstractNOR);
RULE R01
RULE R02
9 Parts in Device
Total Variations Before Rules: 9! = 362,880
Total Variations After Rules: 5040
AbstractNOR_0
AbstractNOR_1
AbstractNOR_2
AbstractNOR_3
AbstractNOR_4
!A OR B: AbstractNOR_5
AbstractNOR_6
AbstractNOR_7


NOR_GATE.EUG
permute(AbstractNOR);
AbstractNOR_5
NOR_GATE.EUG
product(AbstractNOR_5);

TOGGLE_SWITCH.EUG

Property name(txt);
Property sequence(txt);
Property represses(txt);
Property repressedBy(txt);
Property orientation(txt);



Part Repressor(name, sequence, represses);

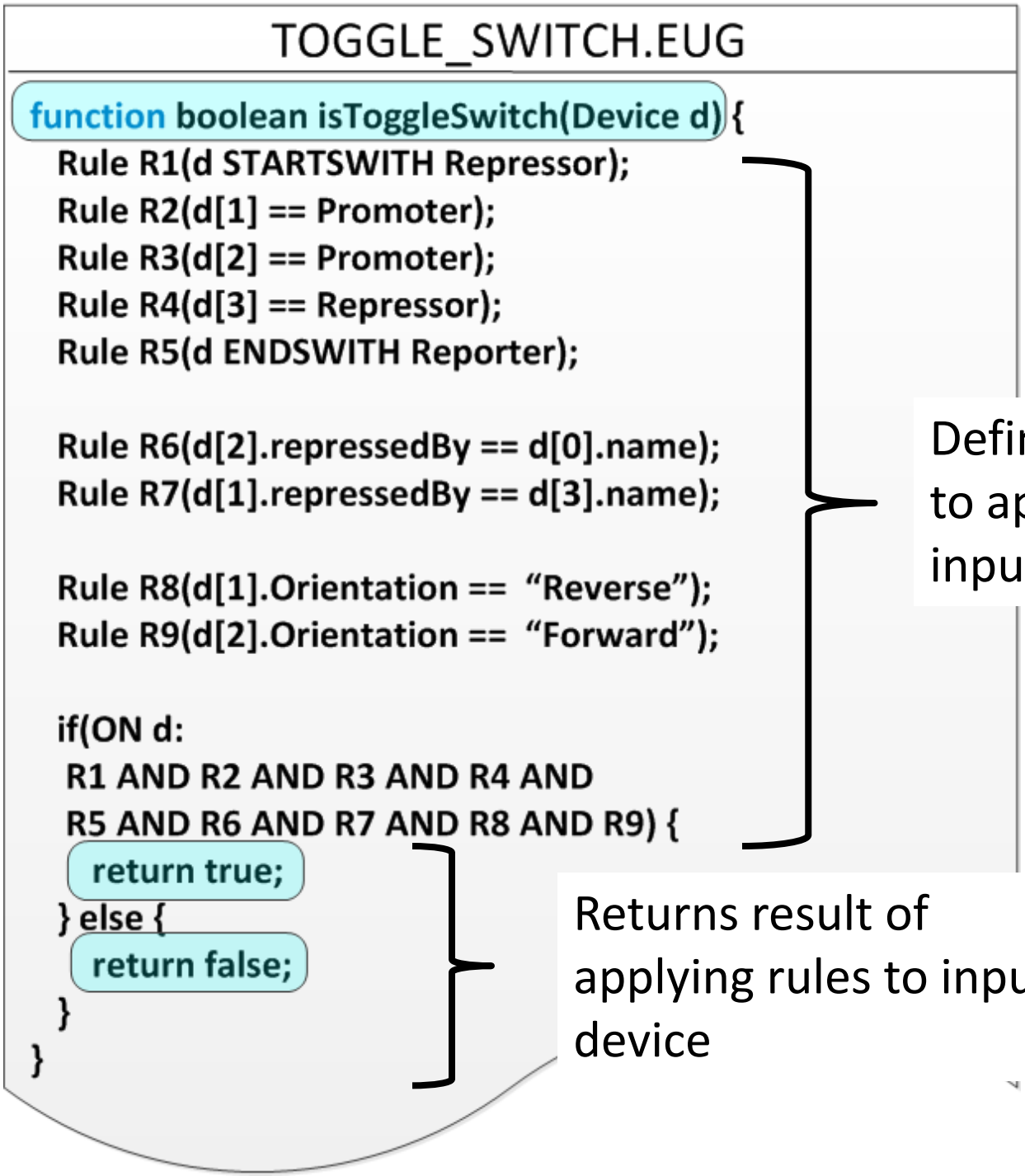
Part Promoter(name, sequence, repressedBy, orientation);

Part Reporter(name, sequence);

Promoter pAmtR("pAmtR", "cgacg...atgctagc", "AmtR", “left”);
Promoter pcl("pcl", "cgacgta...tataatgctagc", "cl", “right”);

Repressor AmtR("AmtR", "ctatggactatg...cccataccc", "pAmtR");
Repressor cl("cl", "ctatggactatgt...tagggcccataccc", "pcl");

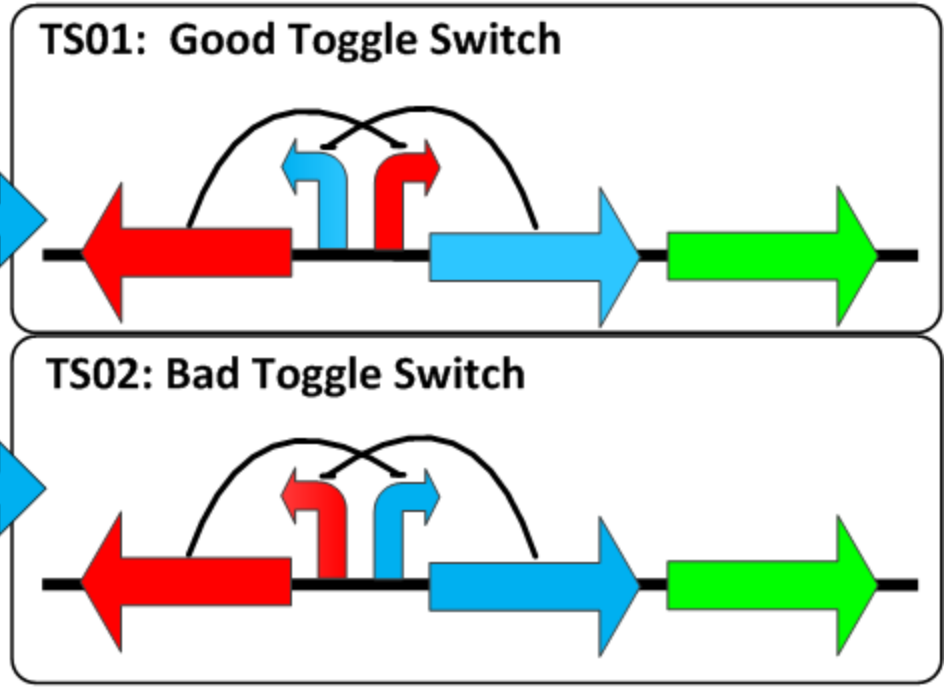
Reporter GFP("GFP_test", "atgcgtaaagg...cgctttagtagcttaa");


TOGGLE_SWITCH.EUG
function boolean isToggleSwitch(Device d) {
Rule R1(d STARTSWITH Repressor);
Rule R2(d[1] == Promoter);
Rule R3(d[2] == Promoter);
Rule R4(d[3] == Repressor);
Rule R5(d ENDSWITH Reporter);
Rule R6(d[2].repressedBy == d[0].name);
Rule R7(d[1].repressedBy == d[3].name);
Rule R8(d[1].Orientation == "Reverse");
Rule R9(d[2].Orientation == "Forward");
if(ON d:
R1 AND R2 AND R3 AND R4 AND
R5 AND R6 AND R7 AND R8 AND R9) {
return true;
} else {
return false;
}
}
Define rules to apply to input device
Returns result of applying rules to input device

Designing Toggle Switches

TOGGLE_SWITCH.EUG

```
// Device 1:  good toggle switch
Device ts01(cI, pAmtR, pcI,
AmtR, GFP);

// Device 2:  bad toggle switch
Device ts02(cI, pcI, pAmtR,
AmtR, GFP);
```

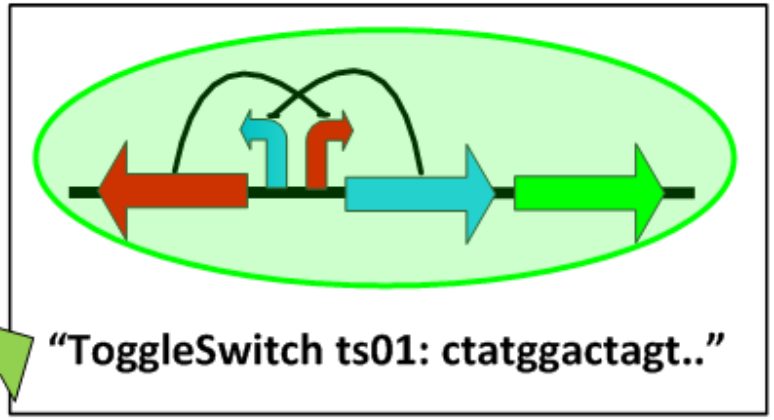


Checking Results

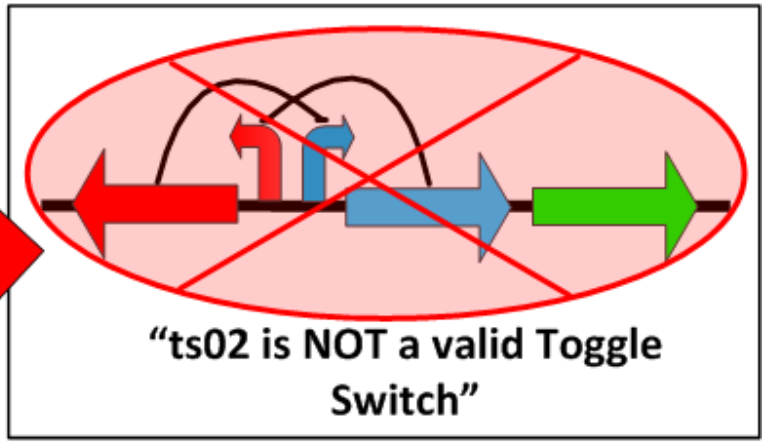
TOGGLE_SWITCH.EUG

```
// get all devices of the design space
Device[] arrDevices;
arrDevices = getAllDevices();

// iterate over all devices
for(num i=0; i<arrDevices.size(); i++) {
   Device d = arrDevices[i];

   // call the isToggleSwitch function to
   // check if the current device
   // is a valid Toggle Switch
   if(isToggleSwitch(d)) {
      // print the sequence of the Toggle Switch
      print("ToggleSwich ",d,":",
            d.toSequence();)
   } else {
      print(d, " is not a valid Toggle Switch");
   }
}
```

ts01



ts02



Summary of Case Studies

- Highlights new features of Eugene
- NOR gate
    - permute()
    - product()
- Toggle switch
    - Function prototyping
    - FOR/WHILE/DO-WHILE
    - IF-ELSE

Future Work

- Integration with assembly workflow
- Integration with characterization and simulation workflows
- Additional language features

Conclusions

- We have shown two case studies that highlight the new features in Eugene.
- Unique in its ability to define complex design space.
- iGEM release
- For source code, full documentation and additional case studies, visit: **http://www.eugenecad.org/**

Acknowledgments

- Agilent Technologies, for funding the further development of Eugene

- Repressor/Promoter pairs are from the Voigt lab

Questions