Towards temporal logic computation using DNA strand displacement reactions (SUPPORTING INFORMATION)

Matthew R. Lakin*[1,2,3] and Darko Stefanovic[2,3]

[1]Department of Chemical & Biological Engineering, University of New Mexico, NM, USA
[2]Department of Computer Science, University of New Mexico, NM, USA
[3]Center for Biomedical Engineering, University of New Mexico, NM, USA

April 7, 2017

S1 DSD code listings

This and the following sections present DSD code for analyzing input signal sequences to see whether they satisfy different temporal logic formulae. In each case, the code was compiled and simulated using the "beta" version of the DSD compiler and simulator [1] that supports species mixing events as well as the inclusion of user-defined reactions, which we use below to implement degradation of the input signals.

In these listings, the signatures of the `Signal`, `CatalystGate`, `AndGate` and `OrGate` modules are slightly more involved than those presented in the main text. This is because they include separate arguments for the toehold and recognition domains for each signal, as well as additional arguments for the number of copies (or, the concentration) of each species to include. Additionally, there is a set of "parameters" at the top of each listing, which contains additional parameters such as the amounts of different species to include, the time to wait between additions of input signals, and the "back pressure" factors for different gate types. "Back pressure" means that, for the first input signal consumed by a gate, we may include some quantity of the corresponding "backward" strand (that is, the strand released by binding of the input) to bias the reaction back towards input release. This technique has been used in previous work [1] to prevent excessive sequestration of the first input by a gate, which in this work is useful because the input signals must be free in solution to be degraded, which is crucial to prevent unwanted circuit responses triggered by input signals left over from earlier in the simulation.

S1.1 DSD code listing for formula $A \sqsubset B \sqsubset C$

```
(* Compilation and simulation directives *)
directive compilation infinite
directive simulation deterministicstiff
directive duration 120000.0 points 12000
directive plot SignalA(); SignalB(); SignalC(); SignalD();
              SignalWire0(); SignalWire1(); SignalWire2(); SignalZ()

(* Parameters - amounts of species to add, and wait time between additions *)
directive parameters [ SignalAmt = 10.0;
                       CatalystGateAmt = 1000.0;
                       AndGateAmt = 1000.0;
                       OrGateAmt1 = 1000.0;
                       OrGateAmt2 = 10.0;
```

*Corresponding author (mlakin@cs.unm.edu)

```
                        CatalystGateBackPressure = 2.0;
                        AndGateBackPressure = 0.0;
                        OrGateBackPressure = 0.0;
                        WaitTime = 20000.0;
                        SignalDegRate = 0.0005 ]

(* Signal sequence: A-B-C-D *)
(* These events are changed to modify the signal sequence *)
directive event SignalA() SignalAmt @ (1.0*WaitTime)
directive event SignalB() SignalAmt @ (2.0*WaitTime)
directive event SignalC() SignalAmt @ (3.0*WaitTime)
directive event SignalD() SignalAmt @ (4.0*WaitTime)

(*****************************************)

(* Generic signal strand *)
def Signal(tx,x) = ( <tx^ x> )

(* Degradation reaction for a degradable signal strand *)
def DegReaction(tx,x) = ( rxn Signal(tx,x) ->{SignalDegRate} )

(* Catalytic transducer gate, derived from Chen,Seelig paper *)
(* Some back-pressure helps with stripping off the catalytic inputs *)
def CatalyticTransducer(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                          *                  <tr^  r>
| (N*int_of_float(CatalystGateBackPressure)) *       <x ty^>
| N                                          * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                          * <i   tz^>
| N                                          *         <z   tx^>
| N                                          *                <x   tr^>
| N                                          * [i]:[tz^ z]:[tx^ x]:[tr^ r]{tq^*} )

(* And gate *)
(* Some back-pressure may help with stripping off the first inputs *)
def AndGate(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                       *                 <tr^   r>
| (N*int_of_float(AndGateBackPressure)) *       <x ty^>
| N                                       * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                       * <i   tz^>
| N                                       *         <z   tr^>
| N                                       * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Or gate *)
(* Some back-pressure may help with stripping off the inputs *)
def OrGate(N1,N2,tx,x,ty,y,tz,z) =
( N1                                      *         <tr^   r>
| (N1*int_of_float(OrGateBackPressure)) *     <x tr^>
| N1                                      * {tx^*}[x tr^]:[r tq^]
| (N1*int_of_float(OrGateBackPressure)) *     <y tr^>
| N1                                      * {ty^*}[y tr^]:[r tq^]
| N1                                      * <i   tz^>
| N1                                      *         <z   tr^>
| N2                                      * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Specific species, for sane plotting in DSD Beta *)

(* Signal strands for catalytic, time varying signals *)
```

```
def SignalA() = Signal(ta,a)
def SignalB() = Signal(tb,b)
def SignalC() = Signal(tc,c)
def SignalD() = Signal(td,d)

(* Degradation reactions for signal strands for catalytic, time varying signals *)
def DegReactionA() = DegReaction(ta,a)
def DegReactionB() = DegReaction(tb,b)
def DegReactionC() = DegReaction(tc,c)
def DegReactionD() = DegReaction(td,d)

(* Signal strands for wire species *)
def SignalWire0() = Signal(twire0,wire0)
def SignalWire1() = Signal(twire1,wire1)
def SignalWire2() = Signal(twire2,wire2)
def SignalZ() = Signal(tz,z)

(****************************************)

( CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire0,wire0,twire1,wire1)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tb,b,twire1,wire1,twire2,wire2)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tc,c,twire2,wire2,tz,z)
| 0 * SignalA()
| 0 * SignalB()
| 0 * SignalC()
| 0 * SignalD()
| DegReactionA()
| DegReactionB()
| DegReactionC()
| DegReactionD()
| int_of_float(SignalAmt) * SignalWire0()
| 0 * SignalWire1()
| 0 * SignalWire2()
| 0 * SignalZ()
)
```

S1.2 DSD code listing for formula $(A \sqsubset B) \wedge (C \sqsubset D)$

```
(* Compilation and simulation directives *)
directive compilation infinite
directive simulation deterministicstiff
directive duration 120000.0 points 12000
directive plot SignalA(); SignalB(); SignalC(); SignalD();
               SignalWire0(); SignalWire1(); SignalX(); SignalWire2();
               SignalWire3(); SignalY(); SignalZ()

(* Parameters - amounts of species to add, and wait time between additions *)
directive parameters [ SignalAmt = 10.0;
                       CatalystGateAmt = 1000.0;
                       AndGateAmt = 1000.0;
                       OrGateAmt1 = 1000.0;
                       OrGateAmt2 = 10.0;
                       CatalystGateBackPressure = 2.0;
                       AndGateBackPressure = 0.0;
                       OrGateBackPressure = 0.0;
                       WaitTime = 20000.0;
                       SignalDegRate = 0.0005 ]

(* Signal sequence: A-B-C-D *)
```

```
(* These events are changed to modify the signal sequence *)
directive event SignalA() SignalAmt @ (1.0*WaitTime)
directive event SignalB() SignalAmt @ (2.0*WaitTime)
directive event SignalC() SignalAmt @ (3.0*WaitTime)
directive event SignalD() SignalAmt @ (4.0*WaitTime)

(*****************************************)

(* Generic signal strand *)
def Signal(tx,x) = ( <tx^ x> )

(* Degradation reaction for a degradable signal strand *)
def DegReaction(tx,x) = ( rxn Signal(tx,x) ->{SignalDegRate} )

(* Catalytic transducer gate, derived from Chen,Seelig paper *)
(* Some back-pressure helps with stripping off the catalytic inputs *)
def CatalyticTransducer(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                          *                  <tr^  r>
| (N*int_of_float(CatalystGateBackPressure)) *       <x ty^>
| N                                          * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                          * <i   tz^>
| N                                          *         <z   tx^>
| N                                          *                  <x   tr^>
| N                                          * [i]:[tz^ z]:[tx^ x]:[tr^ r]{tq^*} )

(* And gate *)
(* Some back-pressure may help with stripping off the first inputs *)
def AndGate(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                       *                 <tr^   r>
| (N*int_of_float(AndGateBackPressure)) *       <x ty^>
| N                                       * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                       * <i   tz^>
| N                                       *         <z   tr^>
| N                                       * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Or gate *)
(* Some back-pressure may help with stripping off the inputs *)
def OrGate(N1,N2,tx,x,ty,y,tz,z) =
( N1                                       *         <tr^   r>
| (N1*int_of_float(OrGateBackPressure)) *     <x tr^>
| N1                                       * {tx^*}[x tr^]:[r tq^]
| (N1*int_of_float(OrGateBackPressure)) *     <y tr^>
| N1                                       * {ty^*}[y tr^]:[r tq^]
| N1                                       * <i   tz^>
| N1                                       *         <z   tr^>
| N2                                       * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Specific species, for sane plotting in DSD Beta *)

(* Signal strands for catalytic, time varying signals *)
def SignalA() = Signal(ta,a)
def SignalB() = Signal(tb,b)
def SignalC() = Signal(tc,c)
def SignalD() = Signal(td,d)

(* Degradation reactions for signal strands for catalytic, time varying signals *)
def DegReactionA() = DegReaction(ta,a)
```

```
def DegReactionB() = DegReaction(tb,b)
def DegReactionC() = DegReaction(tc,c)
def DegReactionD() = DegReaction(td,d)

(* Signal strands for wire species *)
def SignalWire0() = Signal(twire0,wire0)
def SignalWire1() = Signal(twire1,wire1)
def SignalX() = Signal(tx,x)
def SignalWire2() = Signal(twire2,wire2)
def SignalWire3() = Signal(twire3,wire3)
def SignalY() = Signal(ty,y)
def SignalZ() = Signal(tz,z)

(*****************************************)

( CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire0,wire0,twire1,wire1)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tb,b,twire1,wire1,tx,x)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tc,c,twire2,wire2,twire3,wire3)
| CatalyticTransducer(int_of_float(CatalystGateAmt),td,d,twire3,wire3,ty,y)
| AndGate(int_of_float(AndGateAmt),tx,x,ty,y,tz,z)
| 0 * SignalA()
| 0 * SignalB()
| 0 * SignalC()
| 0 * SignalD()
| DegReactionA()
| DegReactionB()
| DegReactionC()
| DegReactionD()
| int_of_float(SignalAmt) * SignalWire0()
| 0 * SignalWire1()
| 0 * SignalX()
| int_of_float(SignalAmt) * SignalWire2()
| 0 * SignalWire3()
| 0 * SignalY()
| 0 * SignalZ()
)
```

S1.3 DSD code listing for formula $(A \sqsubset B) \vee (C \sqsubset D)$

```
(* Compilation and simulation directives *)
directive compilation infinite
directive simulation deterministicstiff
directive duration 120000.0 points 12000
directive plot SignalA(); SignalB(); SignalC(); SignalD();
               SignalWire0(); SignalWire1(); SignalX(); SignalWire2();
               SignalWire3(); SignalY(); SignalZ()

(* Parameters - amounts of species to add, and wait time between additions *)
directive parameters [ SignalAmt = 10.0;
                       CatalystGateAmt = 1000.0;
                       AndGateAmt = 1000.0;
                       OrGateAmt1 = 1000.0;
                       OrGateAmt2 = 10.0;
                       CatalystGateBackPressure = 2.0;
                       AndGateBackPressure = 0.0;
                       OrGateBackPressure = 0.0;
                       WaitTime = 20000.0;
                       SignalDegRate = 0.0005 ]
```

```
(* Signal sequence: A-B-C-D *)
(* These events are changed to modify the signal sequence *)
directive event SignalA() SignalAmt @ (1.0*WaitTime)
directive event SignalB() SignalAmt @ (2.0*WaitTime)
directive event SignalC() SignalAmt @ (3.0*WaitTime)
directive event SignalD() SignalAmt @ (4.0*WaitTime)

(****************************************)

(* Generic signal strand *)
def Signal(tx,x) = ( <tx^ x> )

(* Degradation reaction for a degradable signal strand *)
def DegReaction(tx,x) = ( rxn Signal(tx,x) ->{SignalDegRate} )

(* Catalytic transducer gate, derived from Chen,Seelig paper *)
(* Some back-pressure helps with stripping off the catalytic inputs *)
def CatalyticTransducer(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                          *                 <tr^  r>
| (N*int_of_float(CatalystGateBackPressure)) *       <x ty^>
| N                                          * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                          * <i   tz^>
| N                                          *         <z   tx^>
| N                                          *                 <x   tr^>
| N                                          * [i]:[tz^ z]:[tx^ x]:[tr^ r]{tq^*} )

(* And gate *)
(* Some back-pressure may help with stripping off the first inputs *)
def AndGate(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                       *                 <tr^   r>
| (N*int_of_float(AndGateBackPressure)) *       <x ty^>
| N                                       * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                       * <i   tz^>
| N                                       *         <z   tr^>
| N                                       * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Or gate *)
(* Some back-pressure may help with stripping off the inputs *)
def OrGate(N1,N2,tx,x,ty,y,tz,z) =
( N1                                      *         <tr^   r>
| (N1*int_of_float(OrGateBackPressure)) *     <x tr^>
| N1                                      * {tx^*}[x tr^]:[r tq^]
| (N1*int_of_float(OrGateBackPressure)) *     <y tr^>
| N1                                      * {ty^*}[y tr^]:[r tq^]
| N1                                      * <i   tz^>
| N1                                      *         <z   tr^>
| N2                                      * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Specific species, for sane plotting in DSD Beta *)

(* Signal strands for catalytic, time varying signals *)
def SignalA() = Signal(ta,a)
def SignalB() = Signal(tb,b)
def SignalC() = Signal(tc,c)
def SignalD() = Signal(td,d)

(* Degradation reactions for signal strands for catalytic, time varying signals *)
```

```
def DegReactionA() = DegReaction(ta,a)
def DegReactionB() = DegReaction(tb,b)
def DegReactionC() = DegReaction(tc,c)
def DegReactionD() = DegReaction(td,d)

(* Signal strands for wire species *)
def SignalWire0() = Signal(twire0,wire0)
def SignalWire1() = Signal(twire1,wire1)
def SignalX() = Signal(tx,x)
def SignalWire2() = Signal(twire2,wire2)
def SignalWire3() = Signal(twire3,wire3)
def SignalY() = Signal(ty,y)
def SignalZ() = Signal(tz,z)

(*****************************************)

( CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire0,wire0,twire1,wire1)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tb,b,twire1,wire1,tx,x)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tc,c,twire2,wire2,twire3,wire3)
| CatalyticTransducer(int_of_float(CatalystGateAmt),td,d,twire3,wire3,ty,y)
| OrGate(int_of_float(OrGateAmt1),int_of_float(OrGateAmt2),tx,x,ty,y,tz,z)
| 0 * SignalA()
| 0 * SignalB()
| 0 * SignalC()
| 0 * SignalD()
| DegReactionA()
| DegReactionB()
| DegReactionC()
| DegReactionD()
| int_of_float(SignalAmt) * SignalWire0()
| 0 * SignalWire1()
| 0 * SignalX()
| int_of_float(SignalAmt) * SignalWire2()
| 0 * SignalWire3()
| 0 * SignalY()
| 0 * SignalZ()
)
```

S1.4 DSD code listing for formula $A \sqsubset B \sqsubset A \sqsubset B$

```
(* Compilation and simulation directives *)
directive compilation infinite
directive simulation deterministicstiff
directive duration 120000.0 points 12000
directive plot SignalA(); SignalB(); SignalWire0(); SignalWire1();
               SignalWire2(); SignalWire3(); SignalZ()

(* Parameters - amounts of species to add, and wait time between additions *)
directive parameters [ SignalAmt = 10.0;
                       CatalystGateAmt = 1000.0;
                       AndGateAmt = 1000.0;
                       OrGateAmt1 = 1000.0;
                       OrGateAmt2 = 10.0;
                       CatalystGateBackPressure = 2.0;
                       AndGateBackPressure = 0.0;
                       OrGateBackPressure = 0.0;
                       WaitTime = 20000.0;
                       SignalDegRate = 0.0005 ]
```

```
(* Signal sequence: A-A-B-B *)
(* These events are changed to modify the signal sequence *)
directive event SignalA() SignalAmt @ (1.0*WaitTime)
directive event SignalA() SignalAmt @ (2.0*WaitTime)
directive event SignalB() SignalAmt @ (3.0*WaitTime)
directive event SignalB() SignalAmt @ (4.0*WaitTime)

(****************************************)

(* Generic signal strand *)
def Signal(tx,x) = ( <tx^ x> )

(* Degradation reaction for a degradable signal strand *)
def DegReaction(tx,x) = ( rxn Signal(tx,x) ->{SignalDegRate} )

(* Catalytic transducer gate, derived from Chen,Seelig paper *)
(* Some back-pressure helps with stripping off the catalytic inputs *)
def CatalyticTransducer(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                          *                  <tr^  r>
| (N*int_of_float(CatalystGateBackPressure)) *       <x ty^>
| N                                          * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                          * <i   tz^>
| N                                          *         <z   tx^>
| N                                          *                  <x   tr^>
| N                                          * [i]:[tz^ z]:[tx^ x]:[tr^ r]{tq^*} )

(* And gate *)
(* Some back-pressure may help with stripping off the first inputs *)
def AndGate(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                       *                 <tr^   r>
| (N*int_of_float(AndGateBackPressure)) *       <x ty^>
| N                                       * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                       * <i   tz^>
| N                                       *         <z   tr^>
| N                                       * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Or gate *)
(* Some back-pressure may help with stripping off the inputs *)
def OrGate(N1,N2,tx,x,ty,y,tz,z) =
( N1                                      *         <tr^   r>
| (N1*int_of_float(OrGateBackPressure)) *     <x tr^>
| N1                                      * {tx^*}[x tr^]:[r tq^]
| (N1*int_of_float(OrGateBackPressure)) *     <y tr^>
| N1                                      * {ty^*}[y tr^]:[r tq^]
| N1                                      * <i   tz^>
| N1                                      *         <z   tr^>
| N2                                      * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Specific species, for sane plotting in DSD Beta *)

(* Signal strands for catalytic, time varying signals *)
def SignalA() = Signal(ta,a)
def SignalB() = Signal(tb,b)

(* Degradation reactions for signal strands for catalytic, time varying signals *)
def DegReactionA() = DegReaction(ta,a)
def DegReactionB() = DegReaction(tb,b)
```

```
(* Signal strands for wire species *)
def SignalWire0() = Signal(twire0,wire0)
def SignalWire1() = Signal(twire1,wire1)
def SignalWire2() = Signal(twire2,wire2)
def SignalWire3() = Signal(twire3,wire3)
def SignalZ() = Signal(tz,z)

(****************************************)

( CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire0,wire0,twire1,wire1)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tb,b,twire1,wire1,twire2,wire2)
| CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire2,wire2,twire3,wire3)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tb,b,twire3,wire3,tz,z)
| 0 * SignalA()
| 0 * SignalB()
| DegReactionA()
| DegReactionB()
| int_of_float(SignalAmt) * SignalWire0()
| 0 * SignalWire1()
| 0 * SignalWire2()
| 0 * SignalWire3()
| 0 * SignalZ()
)
```

S1.5 DSD code listing for formula $((A \sqsubset B) \vee (A \sqsubset C)) \wedge (C \sqsubset D \sqsubset E)$

```
(* Compilation and simulation directives *)
directive compilation infinite
directive simulation deterministicstiff
directive duration 140000.0 points 14000
directive plot SignalA(); SignalB(); SignalC(); SignalD();
               SignalE(); SignalWire0(); SignalWire1(); SignalV();
               SignalWire2(); SignalWire3(); SignalW(); SignalX();
               SignalWire4(); SignalWire5(); SignalWire6(); SignalY(); SignalZ()

(* Parameters - amounts of species to add, and wait time between additions *)
directive parameters [ SignalAmt = 10.0;
                       CatalystGateAmt = 1000.0;
                       AndGateAmt = 1000.0;
                       OrGateAmt1 = 1000.0;
                       OrGateAmt2 = 10.0;
                       CatalystGateBackPressure = 2.0;
                       AndGateBackPressure = 0.0;
                       OrGateBackPressure = 0.0;
                       WaitTime = 20000.0;
                       SignalDegRate = 0.0005 ]

(* Signal sequence: A-B-C-D-E *)
(* These events are changed to modify the signal sequence *)
directive event SignalA() SignalAmt @ (1.0*WaitTime)
directive event SignalB() SignalAmt @ (2.0*WaitTime)
directive event SignalC() SignalAmt @ (3.0*WaitTime)
directive event SignalD() SignalAmt @ (4.0*WaitTime)
directive event SignalE() SignalAmt @ (5.0*WaitTime)

(****************************************)

(* Generic signal strand *)
```

```
def Signal(tx,x) = ( <tx^ x> )

(* Degradation reaction for a degradable signal strand *)
def DegReaction(tx,x) = ( rxn Signal(tx,x) ->{SignalDegRate} )

(* Catalytic transducer gate, derived from Chen,Seelig paper *)
(* Some back-pressure helps with stripping off the catalytic inputs *)
def CatalyticTransducer(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                          *                 <tr^  r>
| (N*int_of_float(CatalystGateBackPressure)) *       <x ty^>
| N                                          * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                          * <i   tz^>
| N                                          *         <z   tx^>
| N                                          *                 <x   tr^>
| N                                          * [i]:[tz^ z]:[tx^ x]:[tr^ r]{tq^*} )

(* And gate *)
(* Some back-pressure may help with stripping off the first inputs *)
def AndGate(N,tx,x,ty,y,tz,z) =
new tr new r new tq
( N                                       *                 <tr^   r>
| (N*int_of_float(AndGateBackPressure)) *       <x ty^>
| N                                       * {tx^*}[x ty^]:[y tr^]:[r tq^]
| N                                       * <i   tz^>
| N                                       *         <z   tr^>
| N                                       * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Or gate *)
(* Some back-pressure may help with stripping off the inputs *)
def OrGate(N1,N2,tx,x,ty,y,tz,z) =
( N1                                      *         <tr^   r>
| (N1*int_of_float(OrGateBackPressure)) *     <x tr^>
| N1                                      * {tx^*}[x tr^]:[r tq^]
| (N1*int_of_float(OrGateBackPressure)) *     <y tr^>
| N1                                      * {ty^*}[y tr^]:[r tq^]
| N1                                      * <i   tz^>
| N1                                      *         <z   tr^>
| N2                                      * [i]:[tz^ z]:[tr^ r]{tq^*} )

(* Specific species, for sane plotting in DSD Beta *)

(* Signal strands for catalytic, time varying signals *)
def SignalA() = Signal(ta,a)
def SignalB() = Signal(tb,b)
def SignalC() = Signal(tc,c)
def SignalD() = Signal(td,d)
def SignalE() = Signal(te,e)

(* Degradation reactions for signal strands for catalytic, time varying signals *)
def DegReactionA() = DegReaction(ta,a)
def DegReactionB() = DegReaction(tb,b)
def DegReactionC() = DegReaction(tc,c)
def DegReactionD() = DegReaction(td,d)
def DegReactionE() = DegReaction(te,e)

(* Signal strands for wire species *)
def SignalWire0() = Signal(twire0,wire0)
def SignalWire1() = Signal(twire1,wire1)
```

```
def SignalV() = Signal(tv,v)
def SignalWire2() = Signal(twire2,wire2)
def SignalWire3() = Signal(twire3,wire3)
def SignalW() = Signal(tw,w)
def SignalX() = Signal(tx,x)
def SignalWire4() = Signal(twire4,wire4)
def SignalWire5() = Signal(twire5,wire5)
def SignalWire6() = Signal(twire6,wire6)
def SignalY() = Signal(ty,y)
def SignalZ() = Signal(tz,z)

(*****************************************)

( CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire0,wire0,twire1,wire1)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tb,b,twire1,wire1,tv,v)
| CatalyticTransducer(int_of_float(CatalystGateAmt),ta,a,twire2,wire2,twire3,wire3)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tc,c,twire3,wire3,tw,w)
| OrGate(int_of_float(OrGateAmt1),int_of_float(OrGateAmt2),tv,v,tw,w,tx,x)
| CatalyticTransducer(int_of_float(CatalystGateAmt),tc,c,twire4,wire4,twire5,wire5)
| CatalyticTransducer(int_of_float(CatalystGateAmt),td,d,twire5,wire5,twire6,wire6)
| CatalyticTransducer(int_of_float(CatalystGateAmt),te,e,twire6,wire6,ty,y)
| AndGate(int_of_float(AndGateAmt),tx,x,ty,y,tz,z)
| 0 * SignalA()
| 0 * SignalB()
| 0 * SignalC()
| 0 * SignalD()
| 0 * SignalE()
| DegReactionA()
| DegReactionB()
| DegReactionC()
| DegReactionD()
| DegReactionE()
| int_of_float(SignalAmt) * SignalWire0()
| 0 * SignalWire1()
| 0 * SignalV()
| int_of_float(SignalAmt) * SignalWire2()
| 0 * SignalWire3()
| 0 * SignalW()
| 0 * SignalX()
| int_of_float(SignalAmt) * SignalWire4()
| 0 * SignalWire5()
| 0 * SignalWire6()
| 0 * SignalY()
| 0 * SignalZ()
)
```

References

[1] Yordanov B, Kim J, Petersen RL, Shudy A, Kulkarni VV, Phillips A. Computational Design of Nucleic Acid Feedback Control Circuits. ACS Synthetic Biology. 2014;3(8):600–616.