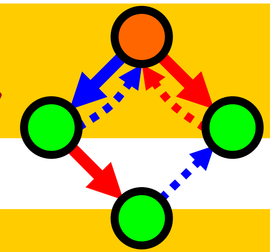
Measure what is measurable, and make measurable what is not so. Galileo Galilei.

Artificial Biochemistry



From Processes to ODEs

Luca Cardelli

Microsoft Research

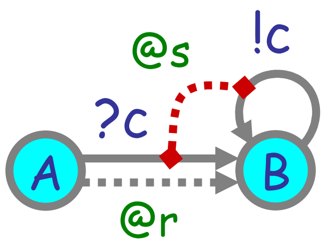
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology

Trento, 2006-05-22..26

www.luca.demon.co.uk/ArtificialBiochemistry.htm

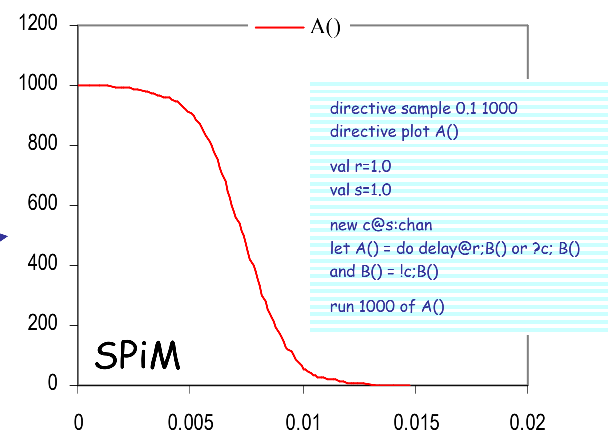
From Processes to ODEs in Two Easy Steps

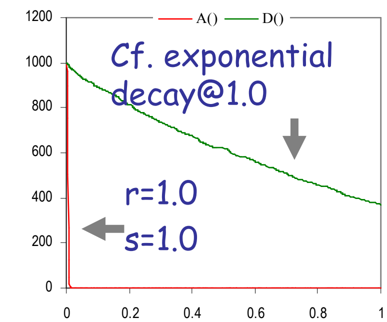
Example: Fast Transitions



$A = \tau_r;B \oplus ?c_{(s)};B$

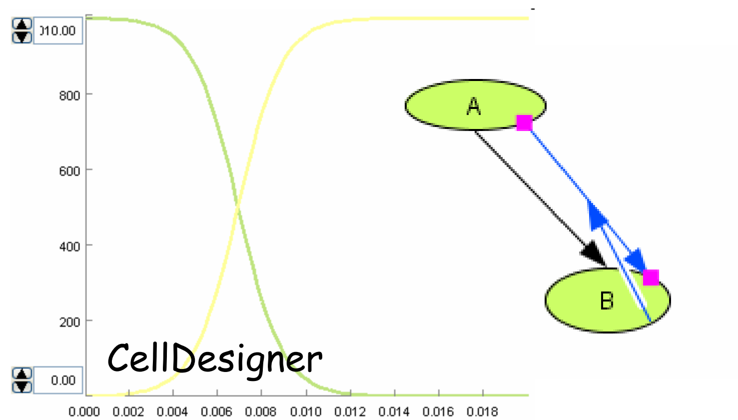
$B = !c_{(s)};B$





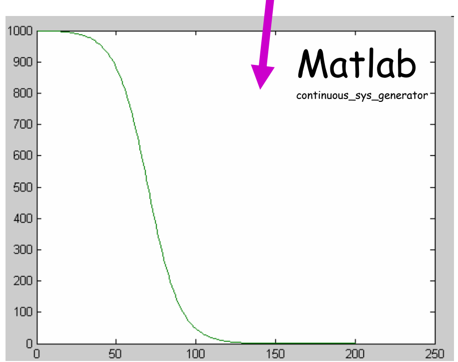
$A \to^r B$

$A+B \to^s B+B$

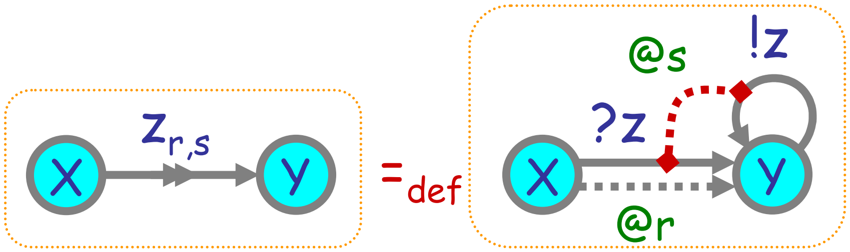


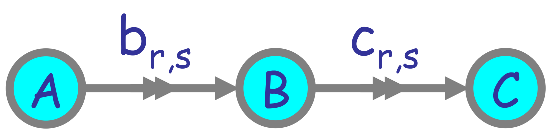
$[A]^\bullet = -r[A]-s[A][B]$

$[B]^\bullet = r[A]+s[A][B]$

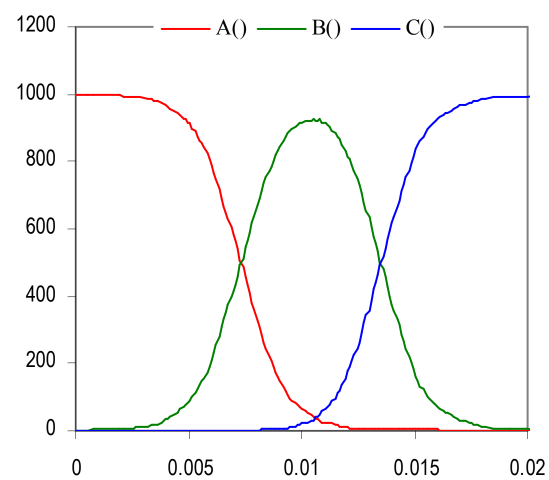


Fast Transitions in Sequence

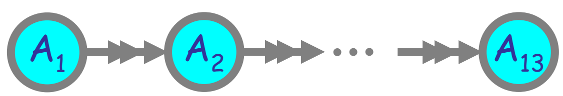




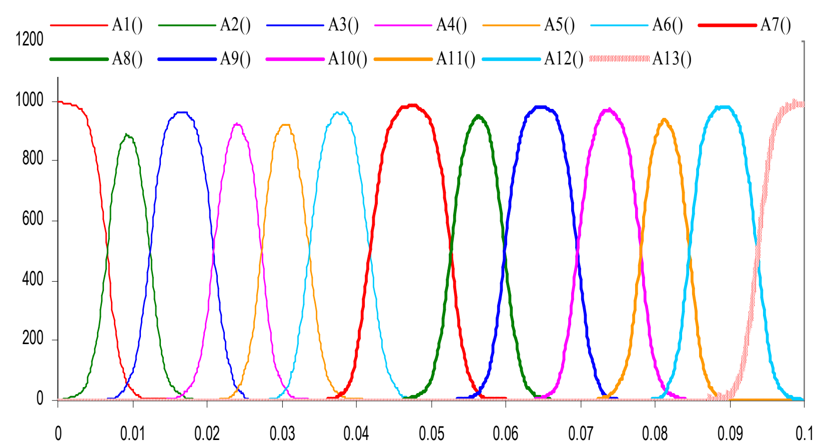
$[B]^\bullet = r([A]-[B])+s[B]([A]-[C])$




```
directive sample 0.1 1000
directive plot A(); B(); C()

val r=1.0
val s=1.0

new b@s:chan new c@s:chan
let A() = do delay@r;B() or ?b; B()
and B() = do !b;B() or delay@r;C() or ?c; C()
and C() = !c;C()

run 1000 of A()
```

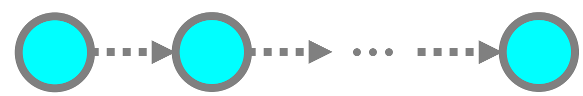


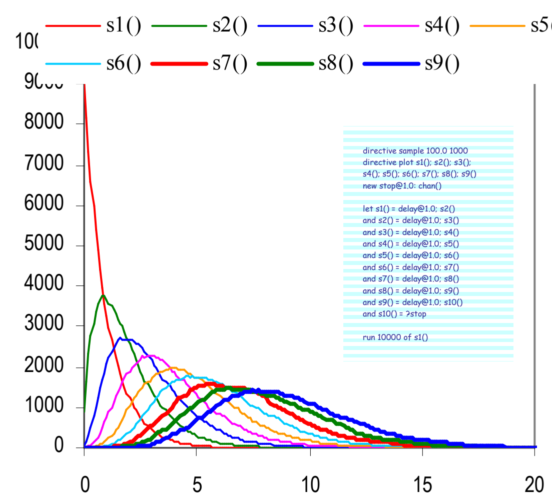
No signal degradation? Why?




```
directive sample 0.1 1000
directive plot A1(); A2(); A3(); A4(); A5(); A6(); A7(); A8();
A9(); A10(); A11(); A12(); A13()

val r=1.0 val s=1.0

new a2@s:chan new a3@s:chan new a4@s:chan
new a5@s:chan new a6@s:chan new a7@s:chan
new a8@s:chan new a9@s:chan new a10@s:chan
new a11@s:chan new a12@s:chan new a13@s:chan
let A1() = do delay@r;A2() or ?a2; A2()
and A2() = do !a2;A2() or delay@r;A3() or ?a3; A3()
and A3() = do !a3;A3() or delay@r;A4() or ?a4; A4()
and A4() = do !a4;A4() or delay@r;A5() or ?a5; A5()
and A5() = do !a5;A5() or delay@r;A6() or ?a6; A6()
and A6() = do !a6;A6() or delay@r;A7() or ?a7; A7()
and A7() = do !a7;A7() or delay@r;A8() or ?a8; A8()
and A8() = do !a8;A8() or delay@r;A9() or ?a9; A9()
and A9() = do !a9;A9() or delay@r;A10() or ?a10; A10()
and A10() = do !a10;A10() or delay@r;A11() or ?a11; A11()
and A11() = do !a11;A11() or delay@r;A12() or ?a12; A12()
and A12() = do !a12;A12() or delay@r;A13() or ?a13; A13()
and A13() = !a13;A13()

run 1000 of A1()
```

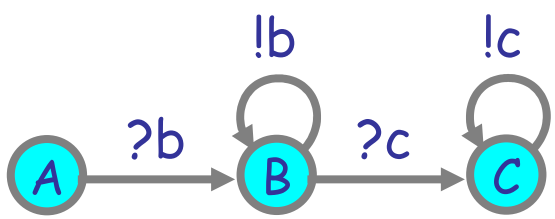
Cf.: Erlang signal degradation



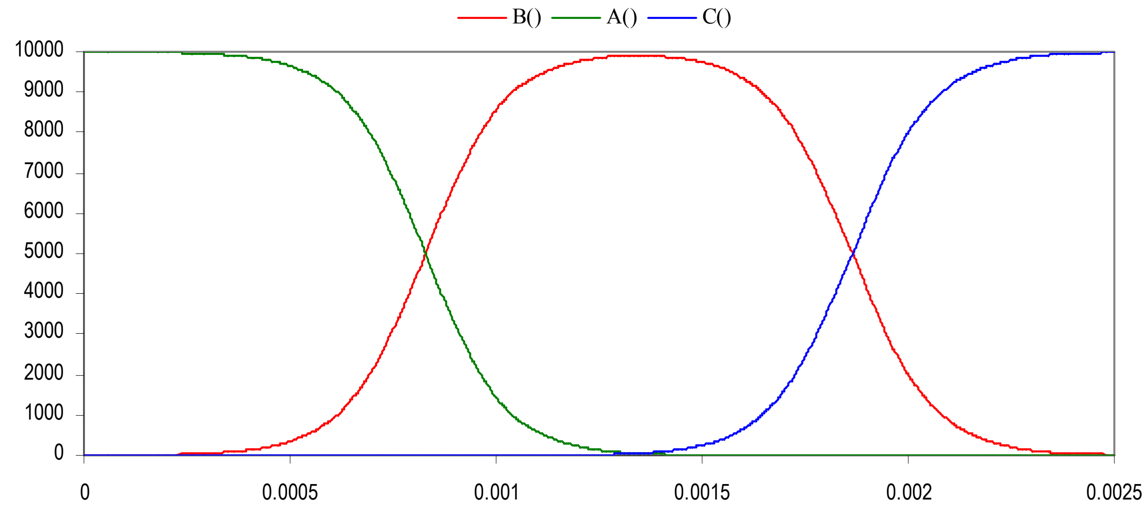


Answer to Bell Exercise

Build a *small* network where one node has a distribution like B():



[B]• = [B]([A]-[C])



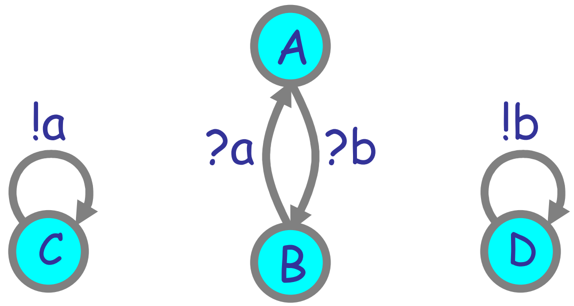

```
directive sample 0.0025 1000
directive plot B(); A(); C()

new b@1.0:chan new c@1.0:chan

let A() = ?b; B()
and B() = do !b;B() or ?c; C()
and C() = !c;C()

run ((10000 of A()) | B() | C())
```

$A = ?b_{(1)};B$
$B = !b_{(1)};B \oplus ?c_{(1)};C$
$C = !c_{(1)};C$

$A+B \rightarrow^1 B+B$
$B+C \rightarrow^1 C+C$

$[A]^\bullet = -[A][B]$
$[B]^\bullet = [A][B]-[B][C]$
$[C]^\bullet = [B][C]$

Exercise: Percentage Sensor




```
directive sample 0.1 1000
directive plot A(); B()

val r=1.0
val s=1.0

new a@s:chan new b@s:chan
let C() = !a;C()
and D() = !b;D()

and A() = ?b;B()
and B() = ?a;A()

run (300 of C() | 100 of D() | 100 of A())
```

Assume there are 100 copies of AB and that all the rates are 1.0.

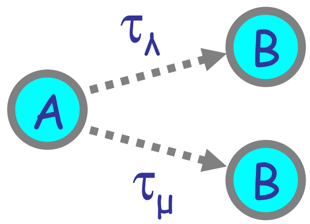
Show that at steady state:
[A] = 100[C]/([C]+[D])

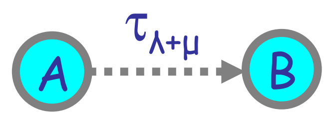
I.e., the A state computes the percentage of C in the total C+D for any amount of C and D.
Note that [C] and [D] are unaffected and could be part of a larger network.

Laws by ODEs

Choice Law by ODEs

$\tau_\lambda;B \oplus \tau_\mu;B = \tau_{\lambda+\mu};B$





$A = \tau_\lambda;B \oplus \tau_\mu;B$

↓

$A \rightarrow^{\lambda} B$
$A \rightarrow^{\mu} B$

↓

$[A]^\bullet = -\lambda[A] - \mu[A]$
$[B]^\bullet = \lambda[A] + \mu[A]$

=

$A = \tau_{\lambda+\mu};B$

↓

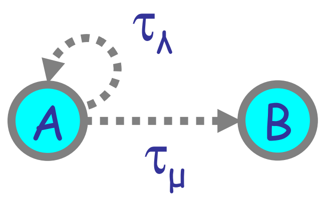
$A \rightarrow^{\lambda+\mu} B$

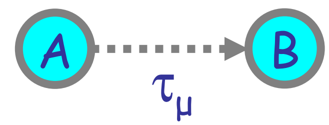
↓

$[A]^\bullet = -(\lambda+\mu)[A]$
$[B]^\bullet = (\lambda+\mu)[A]$

Idle Delay Law by ODEs

$A = \tau_\lambda;A \oplus \tau_\mu;B \quad = \quad A = \tau_\mu;B$





$A = \tau_\lambda;A \oplus \tau_\mu;B$

↓

$A \rightarrow^{\lambda} A$
$A \rightarrow^{\mu} B$

↓

$[A]^{\bullet} = -\mu[A]$
$[B]^{\bullet} = \mu[A]$

=

$A = \tau_\mu;B$

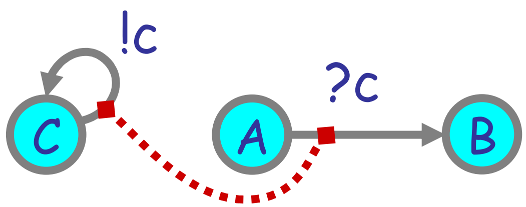
↓

$A \rightarrow^{\mu} B$

↓

$[A]^{\bullet} = -\mu[A]$
$[B]^{\bullet} = \mu[A]$

Idle Interaction Law by ODEs



A = ?c;B
C = !c;C

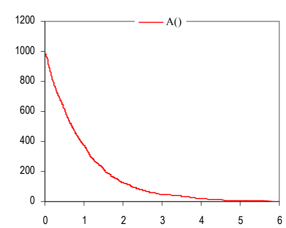
↓

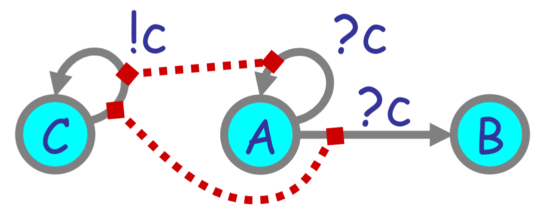
A+C →ʳ B+C

↓

$[A]^\bullet = -r[A][C]$
$[B]^\bullet = r[A][C]$
$[C]^\bullet = 0$

```
directive sample 6.0 1000
directive plot A()

new c@1.0:chan

let A() = ?c; B()
and B() = ()
and C() = !c; C()

run (C() | 1000 of A())
```





A = ?c;A ⊕ ?c;B
C = !c;C

It may seem like A should decrease half as fast, but NO! Two ways to explain:

- State A is *memoryless* of any past idling.
- Activity on c is double

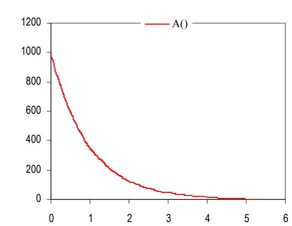
↓

A+C →ʳ A+C
A+C →ʳ B+C

↓

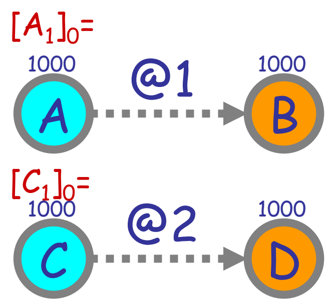
$[A]^\bullet = -r[A][C]$
$[B]^\bullet = r[A][C]$
$[C]^\bullet = 0$

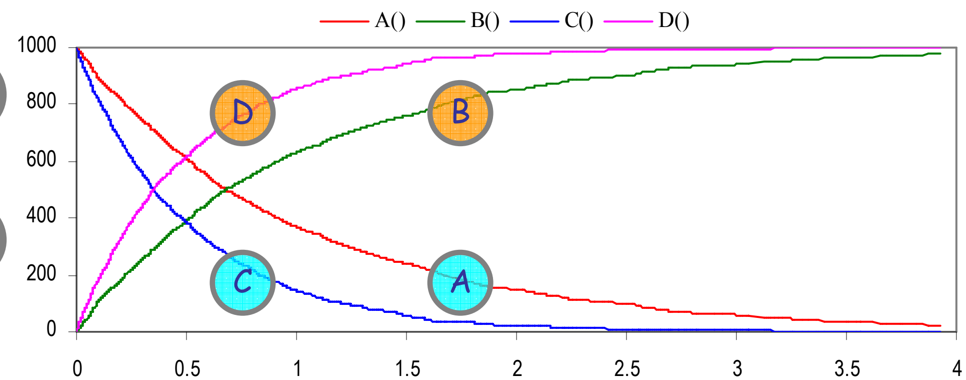
```
directive sample 6.0 1000
directive plot A()

new c@1.0:chan

let A() = do ?c; B() or ?c; A()
and B() = ()
and C() = !c; C()

run (C() | 1000 of A())
```



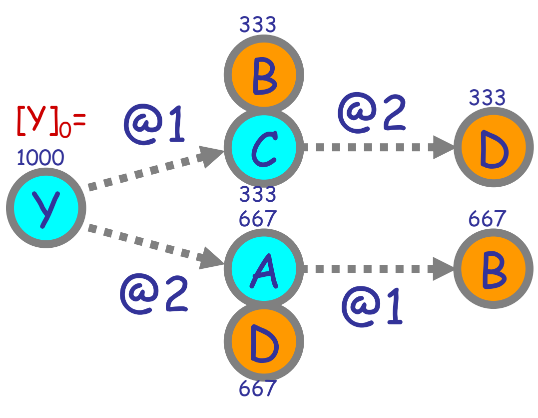
Asynchronous Interleaving

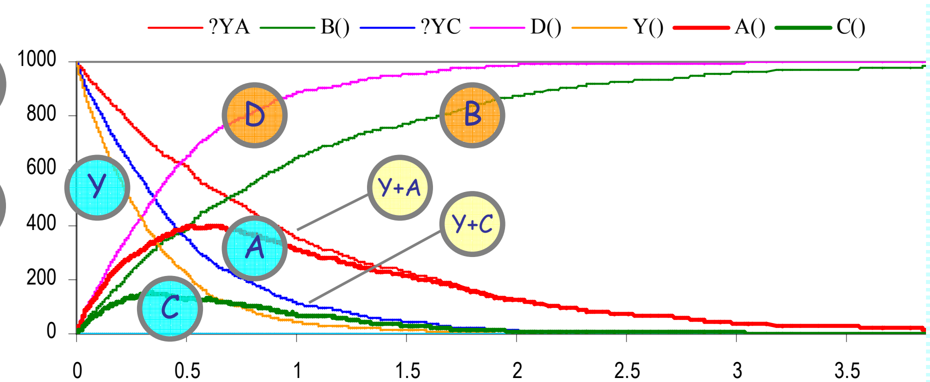
$$\tau_\lambda;B \mid \tau_\mu;D \;=\; \tau_\lambda;(B \mid \tau_\mu;D) + \tau_\mu;(\tau_\lambda;B \mid D)$$






```
directive sample 4.0 10000
directive plot A(); B(); C(); D()

let A() = delay@1.0; B()
and B() = ()

let C() = delay@2.0; D()
and D() = ()

run 1000 of (A() | C())
```






```
directive sample 4.0 10000
directive plot
   ?YA; B(); ?YC; D(); Y(); A(); C()
new YA@1.0:chan new YC@1.0:chan

let A() = do delay@1.0; B() or ?YA
and B() = ()

let C() = do delay@2.0; D() or ?YC
and D() = ()

let Y() =
   do delay@1.0; (B() | C())
   or delay@2.0; (A() | D())
   or ?YA or ?YC

run 1000 of Y()
```

Amazingly, the B's and the D's from the two branches sum up to exponential distributions

Asynchronous Interleaving Law by ODEs

$$\tau_\lambda;B \mid \tau_\mu;D = \tau_\lambda;(B \mid \tau_\mu;D) + \tau_\mu;(\tau_\lambda;B \mid D)$$

Want to show that B and D on both sides have the "same behavior" (equal quantities of B and D produced at all times)

$A_1 = \tau_\lambda;B$
$C_1 = \tau_\mu;D$
$A_1 \mid C_1$

$A_1 \to^\lambda B$
$C_1 \to^\mu D$
$A_1 + C_1$

$[A_1]^\bullet = -\lambda[A_1]$
$[B]^\bullet = \lambda[A_1]$
$[C_1]^\bullet = -\mu[C_1]$
$[D]^\bullet = \mu[C_1]$

=?

$Y = \tau_\lambda;(B \mid C_2) \oplus \tau_\mu;(A_2 \mid D)$
$C_2 = \tau_\mu;D$
$A_2 = \tau_\lambda;B$
Y

$Y \to^\lambda B + C_2$
$Y \to^\mu A_2 + D$
$C_2 \to^\mu D$
$A_2 \to^\lambda B$
Y

$[Y]^\bullet = -\lambda[Y]-\mu[Y]$
$[A_2]^\bullet = \mu[Y]-\lambda[A_2]$
$[B]^\bullet = \lambda[Y]+\lambda[A_2]$
$[C_2]^\bullet = \lambda[Y]-\mu[C_2]$
$[D]^\bullet = \mu[Y]+\mu[C_2]$

$[Y+A_2]^\bullet = -\lambda[Y+A_2]$
$[B]^\bullet = \lambda[Y+A_2]$
$[Y+C_2]^\bullet = -\mu[Y+C_2]$
$[D]^\bullet = \mu[Y+C_2]$

$[Y+A_2]^\bullet = [Y]^\bullet+[A_2]^\bullet$
$= -\lambda[Y]-\mu[Y]+\mu[Y]-\lambda[A_2]$
$= -\lambda[Y]-\lambda[A_2]$
$= -\lambda[Y+A_2]$ — $[Y+A_2]$ decays exponentially!

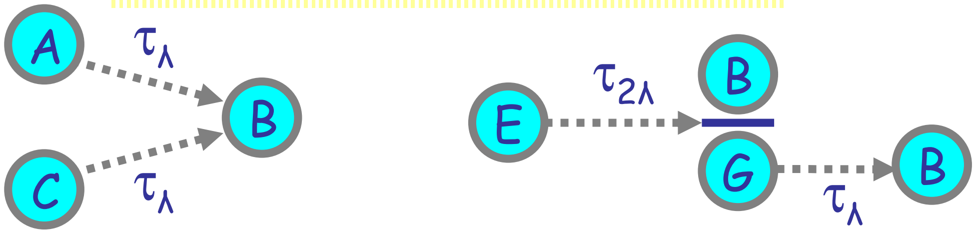
[B] and [D] have equal time evolutions on the two sides provided that $[A_1]=[Y+A_2]$ and $[C_1]=[Y+C_2]$.
This imposes the constraint, in particular, that $[A_1]_0=[Y+A_2]_0$ and $[C_1]_0=[Y+C_2]_0$ (at time zero).
The initial conditions of the right hand system specify that $[A_2]_0=[C_2]_0=0$ (since only Y is present).
Therefore, we obtain that $[A_1]_0=[C_1]_0=[Y]_0$.

So, for example, if we run a stochastic simulation of the left hand side with 1000*A1 and 1000*C1, we obtain the same curves for B and D than a stochastic simulation of the right hand side with 1000*Y.

Equiconfluence Law by ODEs

$$\tau_\lambda;B \mid \tau_\lambda;B = \tau_{2\lambda};(B \mid \tau_\lambda;B)$$

Want to show that B on both sides has the "same behavior" (equal quantities of B produced at all times)



$A = \tau_\lambda;B$
$C = \tau_\lambda;B$
$A \mid C$

=?

$E = \tau_{2\lambda};(B \mid G)$
$G = \tau_\lambda;B$
E

$A \to^{\lambda} B$
$C \to^{\lambda} B$
$A + C$

=?

$E \to^{2\lambda} B + G$
$G \to^{\lambda} B$
E

$[E]^\bullet = -2\lambda[E]$
$[G]^\bullet = 2\lambda[E] - \lambda[G]$
$[B]^\bullet = 2\lambda[E] + \lambda[G]$

$[A]^\bullet = -\lambda[A]$
$[C]^\bullet = -\lambda[C]$
$[B]^\bullet = \lambda[A] + \lambda[C]$

=?

$[A']^\bullet = -\lambda[A']$
$[C']^\bullet = -\lambda[C']$
$[B]^\bullet = \lambda[A'] + \lambda[C']$

let $A' = C' = E+G/2$

$\lambda[A'] + \lambda[C']$
$= \lambda[E+G/2] + \lambda[E+G/2]$
$= 2\lambda[E] + \lambda[G] = [B]^\bullet$

$[A']^\bullet = [E+G/2]^\bullet = [E]^\bullet+[G]^\bullet/2$
$= -2\lambda[E]+(2\lambda[E]-\lambda[G])/2$
$= -\lambda[E]-\lambda[G]/2$
$= -\lambda[E+G/2] = -\lambda[A']$

[B] has equal time evolutions on the two sides provided that $[A]=[E+G/2]$ and $[C]=[E+G/2]$. This imposes the constraint, in particular, that $[A]_0=[E+G/2]_0$ and $[C]_0=[E+G/2]_0$ (at time zero). The initial conditions of the right hand system specify that $[G/2]_0=0$ (since only E is present). Therefore, we obtain that $[A]_0=[C]_0=[E]_0$.

Exercise

- Derive the ODEs for the Repressilator.

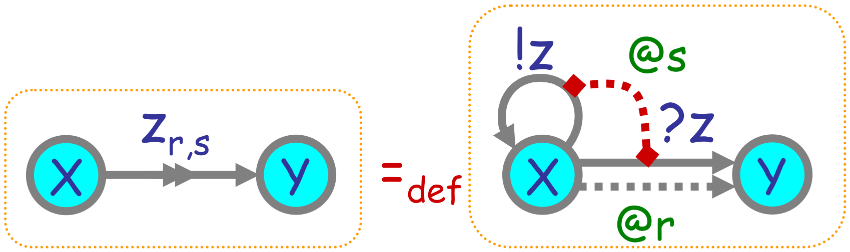
Summary

- From Processes to ODEs
  - Now in two easy steps
  - Caveat: possibly wrong (in the chemistry-to-ODE bit) if stochastic effects are significant (need to use stochastic ODEs?).
- Process Laws by ODEs
  - ODE "semantics" can be used to show process equivalences
- Compositionality
  - Processes are naturally compositional
  - Parametric processes are even better: generate many wildly different ODEs from the same basic process "library" by parameter instantiation

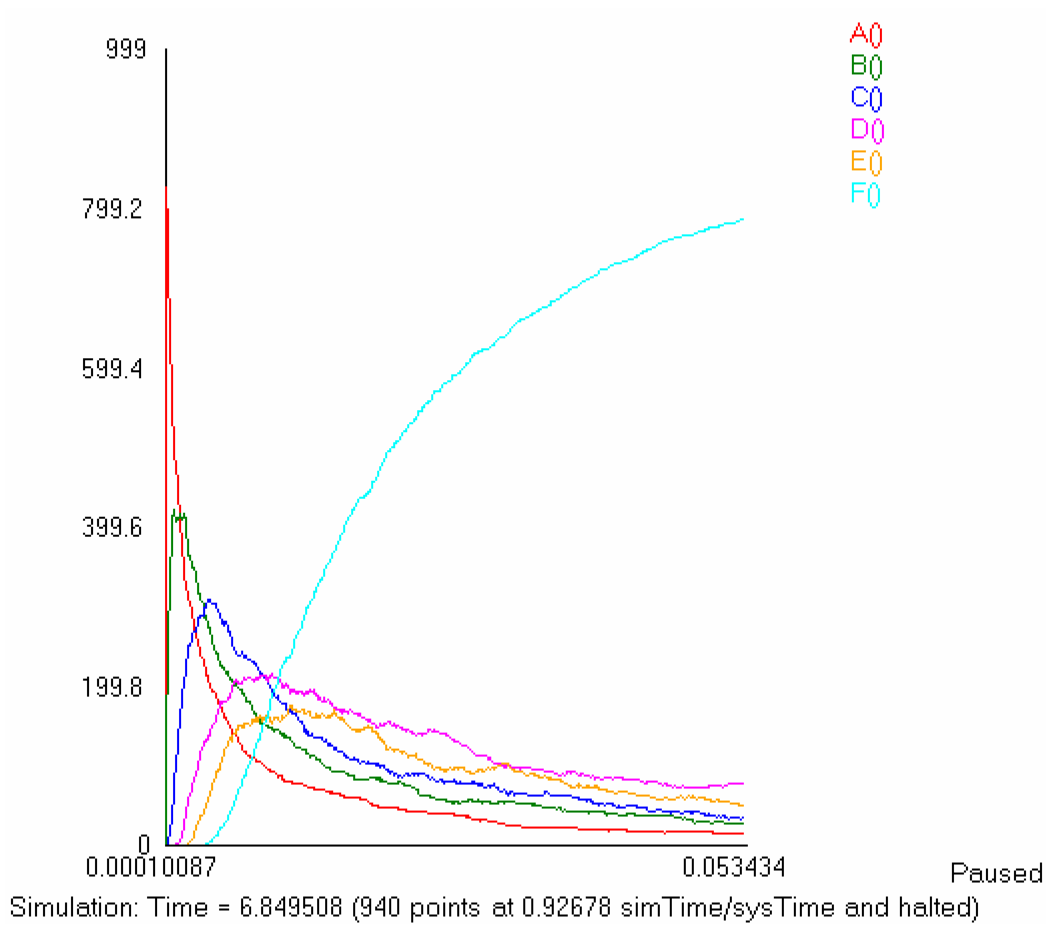
Q?

Appendix

Fast Push




```
directive sample 0.1 1000
directive plot A()

val r=1.0
val s=1.0

new c@s:chan
let A() = do !c; A() or delay@r;B()
or ?c; B()
and B() = ()

run 1000 of A()
```




```
directive sample 0.1 1000
directive plot A(); B(); C(); D(); E(); F()

val r=1.0
val s=1.0

new b@s:chan new c@s:chan
new d@s:chan new e@s:chan
new f@s:chan
let A() = do !b;A() or delay@r;B() or ?b; B()
and B() = do !c;B() or delay@r;C() or ?c; C()
and C() = do !d;C() or delay@r;D() or ?d; D()
and D() = do !e;D() or delay@r;E() or ?d; E()
and E() = do !f;E() or delay@r;F() or ?e; F()
and F() = ()

run 1000 of A()
```