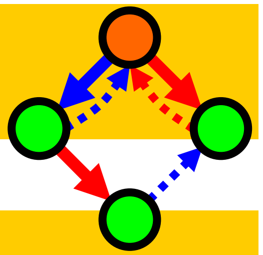
A formal manipulator in mathematics often experiences the discomforting feeling that his pencil surpasses him in intelligence. Howard W. Eves.

Artificial Biochemistry



Brane Calculi

Luca Cardelli

Microsoft Research

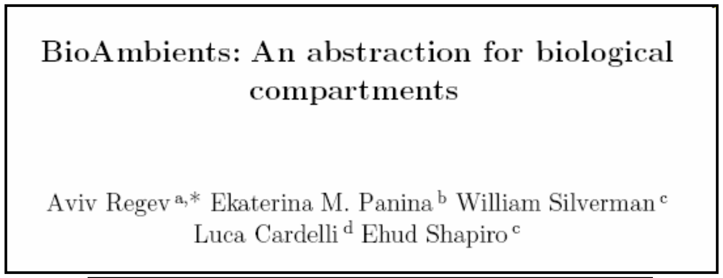
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology

Trento, 2006-05-22..26

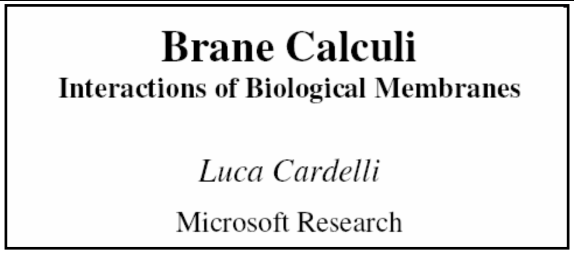
www.luca.demon.co.uk/ArtificialBiochemistry.htm

Related Work

- Membrane Computing
  - From computability theory (now being applied to biological modeling)
- BioAmbients
  - From distributed systems theory (then applied to biological modeling)
- Brane Calculi
  - Bio-inspired membrane operations
- Beta-Binders
  - Bio-inspired process interfaces

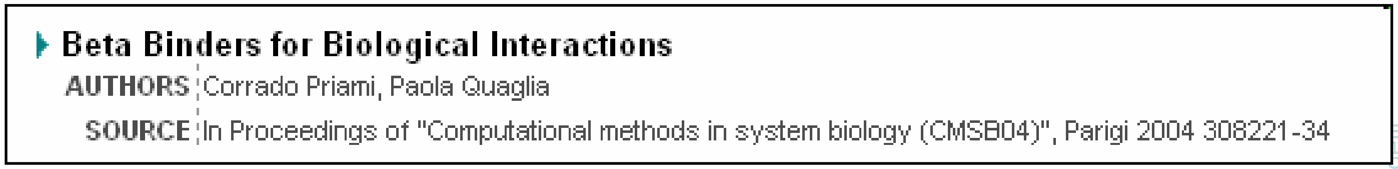

BioAmbients: An abstraction for biological compartments

Aviv Regev[a,*] Ekaterina M. Panina[b] William Silverman[c] Luca Cardelli[d] Ehud Shapiro[c]

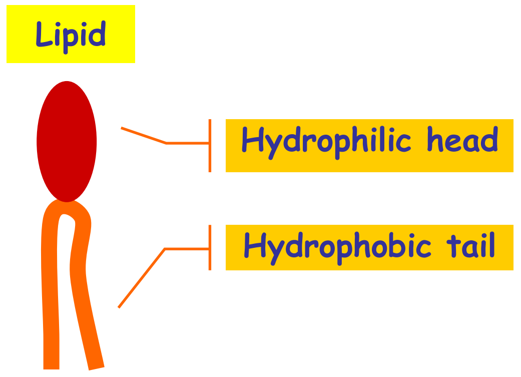

Brane Calculi
Interactions of Biological Membranes

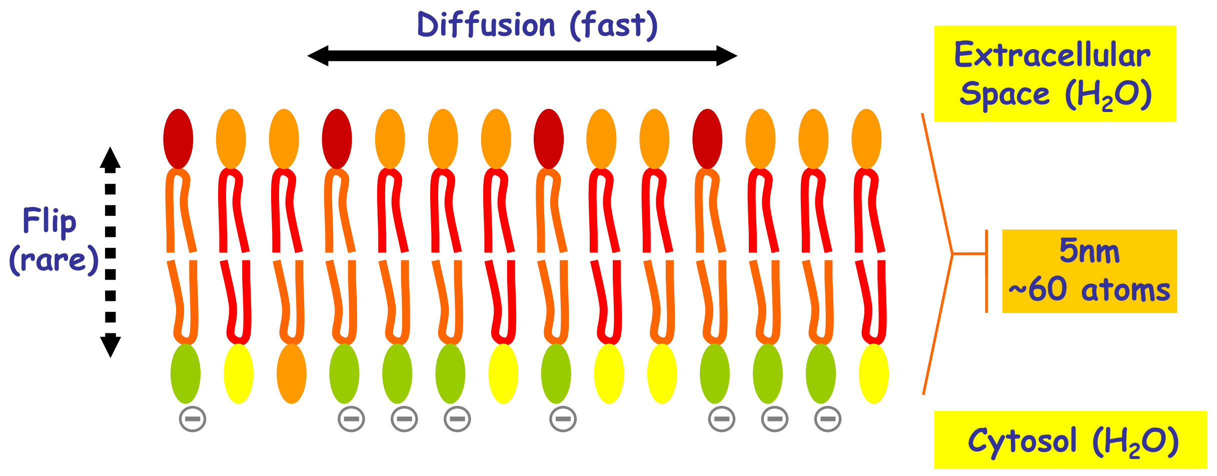
Luca Cardelli

Microsoft Research


Beta Binders for Biological Interactions
AUTHORS Corrado Priami, Paola Quaglia
SOURCE In Proceedings of "Computational methods in system biology (CMSB04)", Parigi 2004 308221-34

Membranes are Oriented 2D Surfaces





Lipid Bilayer

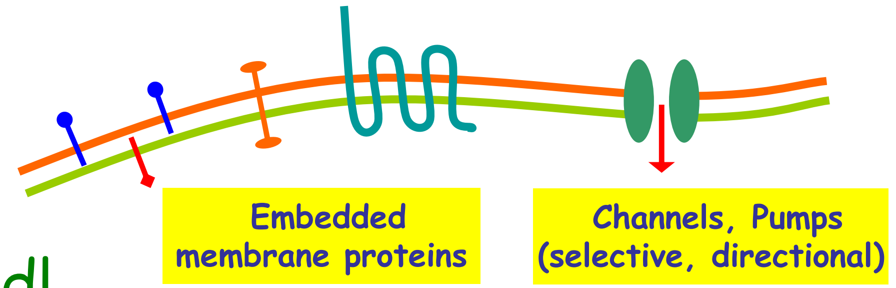
Self-assembling

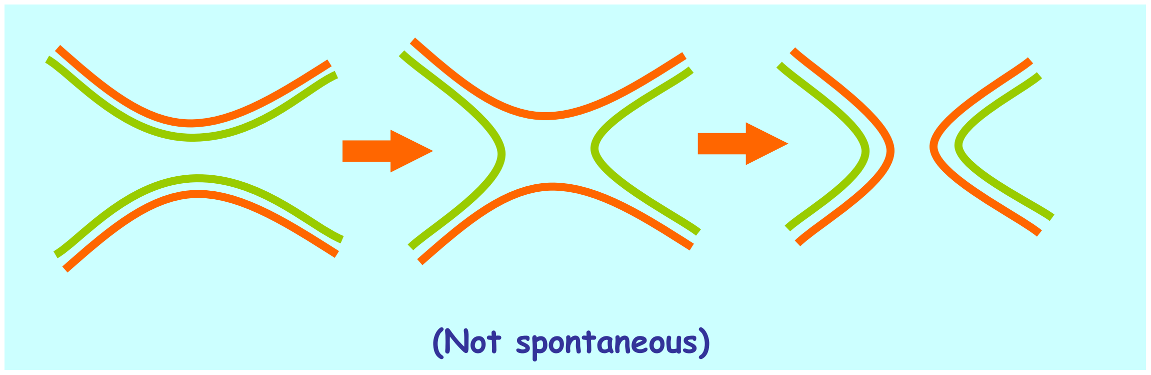
Largely impermeable

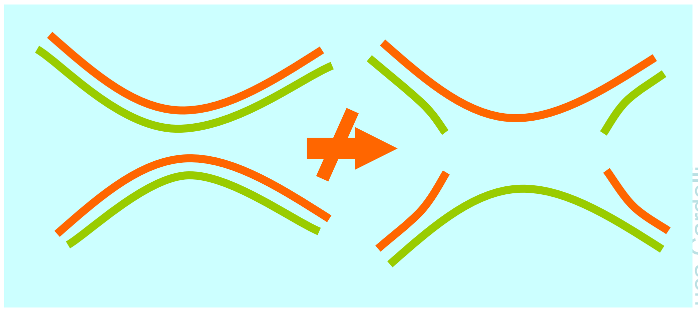
Asymmetrical (in real cells)

With embedded proteins

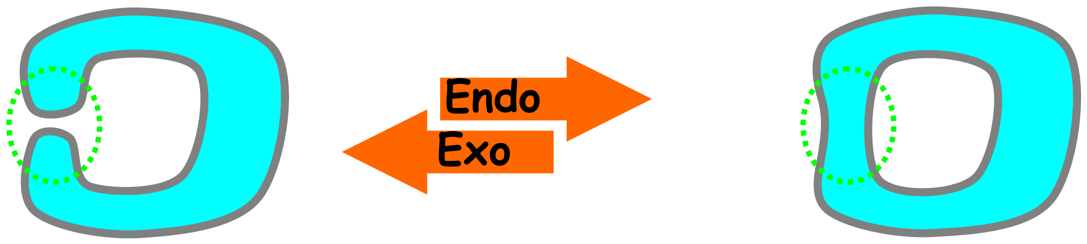
A 2D fluid inside a 3D fluid!

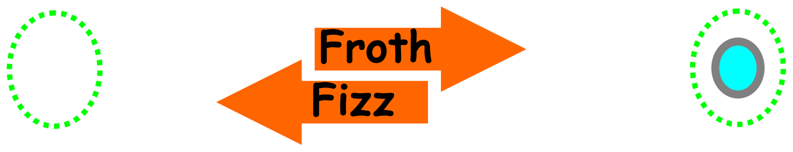

Embedded membrane proteins
Channels, Pumps (selective, directional)



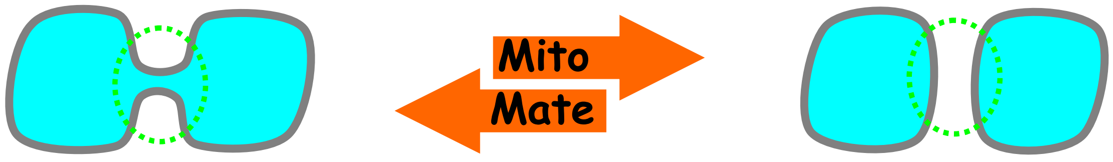


A Complete Set of Bitonal Reactions



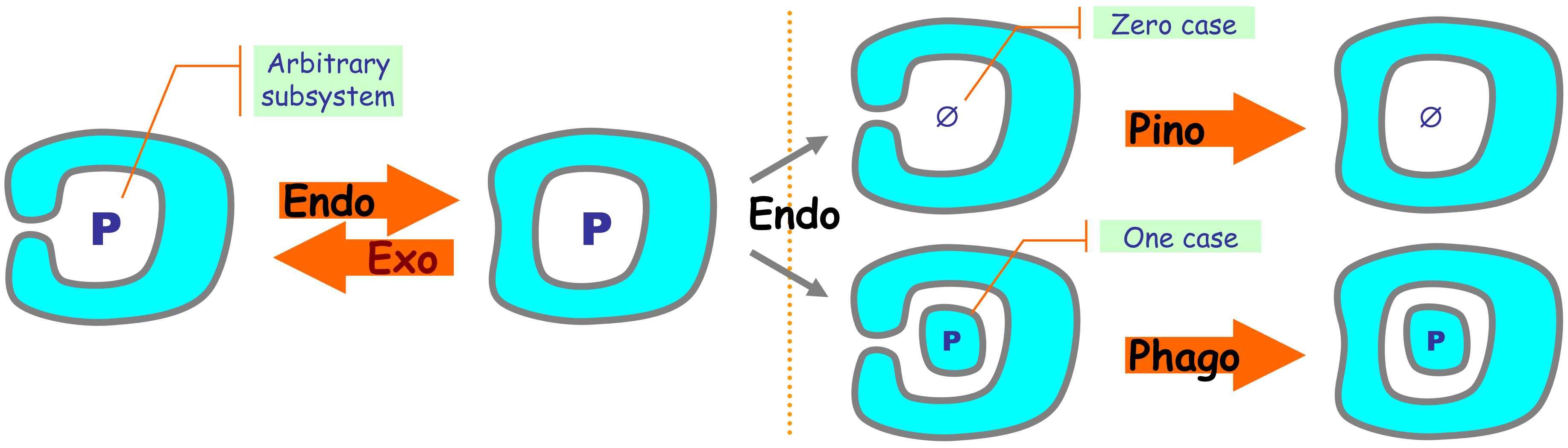


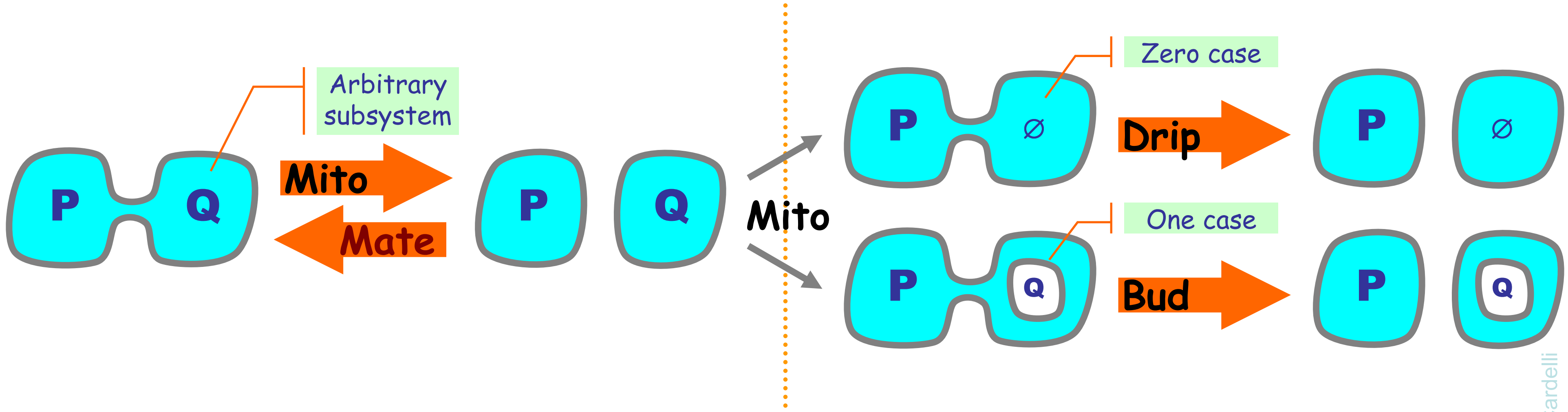
Others bitonal reactions are Derivable, e.g.:



Are *all* other derivable? YES!

"Determinization"

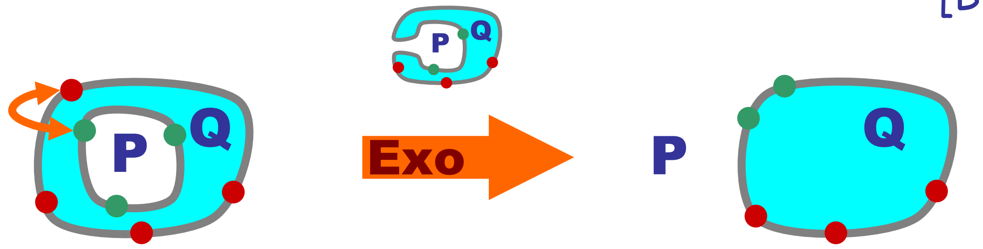


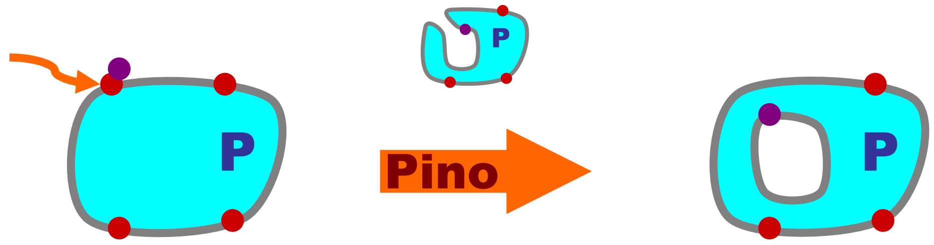


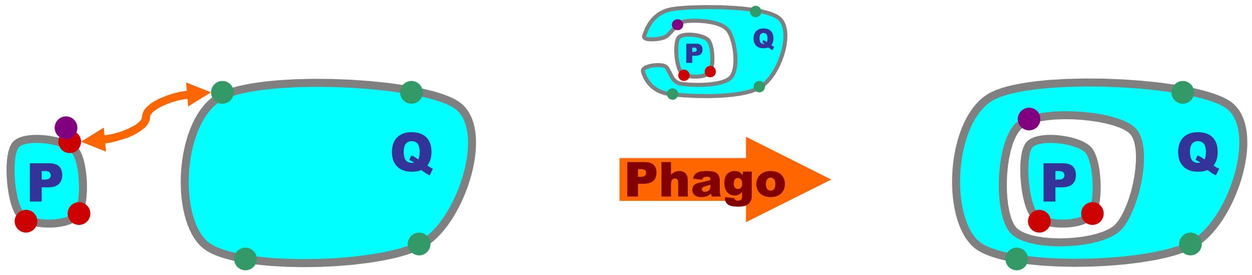
Basic Calculus

Brane Reactions (Cartoons)

A Turing-Complete language
[Busi Gorrieri]

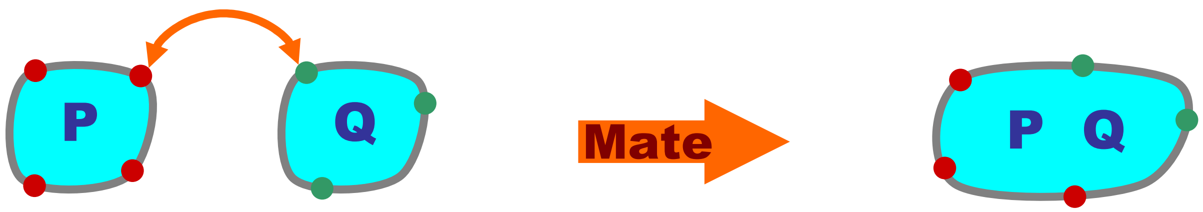


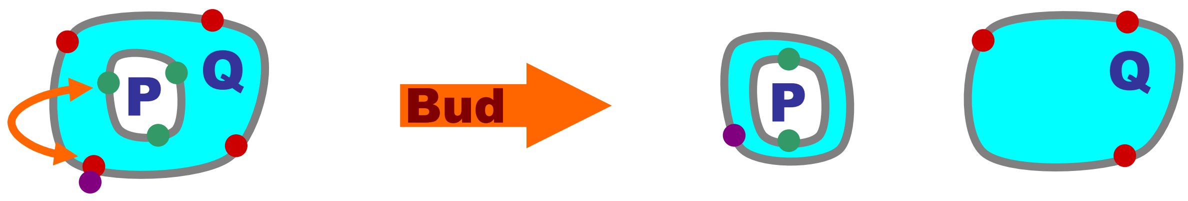


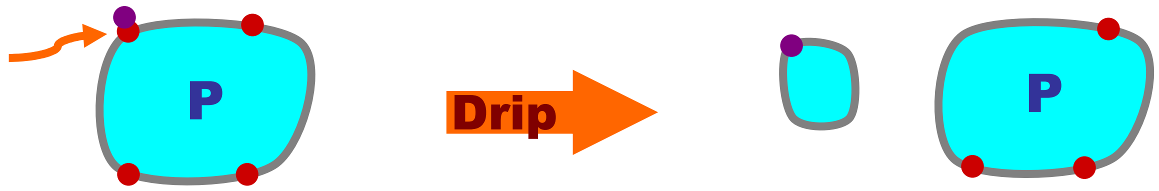


Derivable Reactions (Cartoons)

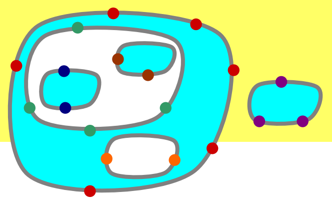
A Decidable-Termination language [Busi Gorrieri]







Brane Calculi



systems	$P,Q \ ::= \ \diamond \mid P \circ Q \mid {*}P \mid \sigma(\!| P |\!)$	nests of membranes
branes	$\sigma,\tau \ ::= \ 0 \mid \sigma\|\tau \mid {*}\sigma \mid a.\sigma$	combinations of **actions**
actions	$a \ ::= \ 1 \mid \ldots$	(fill in as needed)

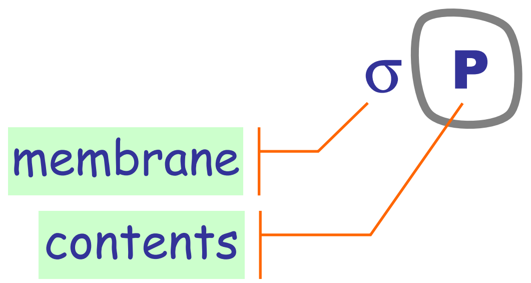
1D fluids (σ) inside a 2D fluid (P)

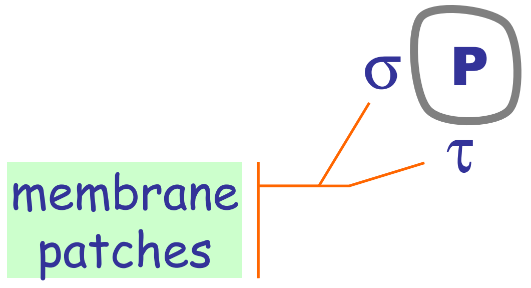
TWO commutative monoids instead of ONE of normal process calculi

$\sigma(\!| P |\!)$

$\sigma|\tau(\!| P |\!)$

$a.\sigma|\tau = (a.\sigma)|\tau$





N.B. Restriction (νn) could be added to both systems and branes. It usually would originate in branes, but would extrude to whole systems.

Congruence $\equiv$ and Reaction $\rightarrow$

	System	Brane
Fluidity	$P \circ Q \equiv Q \circ P$	$\sigma \mid \tau \equiv \tau \mid \sigma$
	$P \circ (Q \circ R) \equiv (P \circ Q) \circ R$	$\sigma \mid (\tau \mid \rho) \equiv (\sigma \mid \tau) \mid \rho$
	$P \circ \diamond \equiv P$	$\sigma \mid 0 \equiv \sigma$
Plentitude	$*P \equiv P \circ *P$ etc.	$*\sigma \equiv \sigma \mid *\sigma$ etc.
Units	$0 \llparenthesis \diamond \rrparenthesis \equiv \diamond$ Froth/Fizz	$1.\sigma \equiv \sigma$ Inaction
Congruence	$P \equiv Q \Rightarrow P \circ R \equiv Q \circ R$	$\sigma \equiv \tau \Rightarrow \sigma \mid \rho \equiv \tau \mid \rho$
	$P \equiv Q \Rightarrow *P \equiv *Q$	$\sigma \equiv \tau \Rightarrow *\sigma \equiv *\tau$
	$P \equiv Q \wedge \sigma \equiv \tau \Rightarrow \sigma \llparenthesis P \rrparenthesis \equiv \tau \llparenthesis Q \rrparenthesis$	$\sigma \equiv \tau \Rightarrow a.\sigma \equiv a.\tau$

Reaction is up to congruence

$$P \equiv P' \wedge P' \rightarrow Q' \wedge Q' \equiv Q \Rightarrow P \rightarrow Q$$

Reactions in solution

$$P \rightarrow Q \Rightarrow P \circ R \rightarrow Q \circ R$$

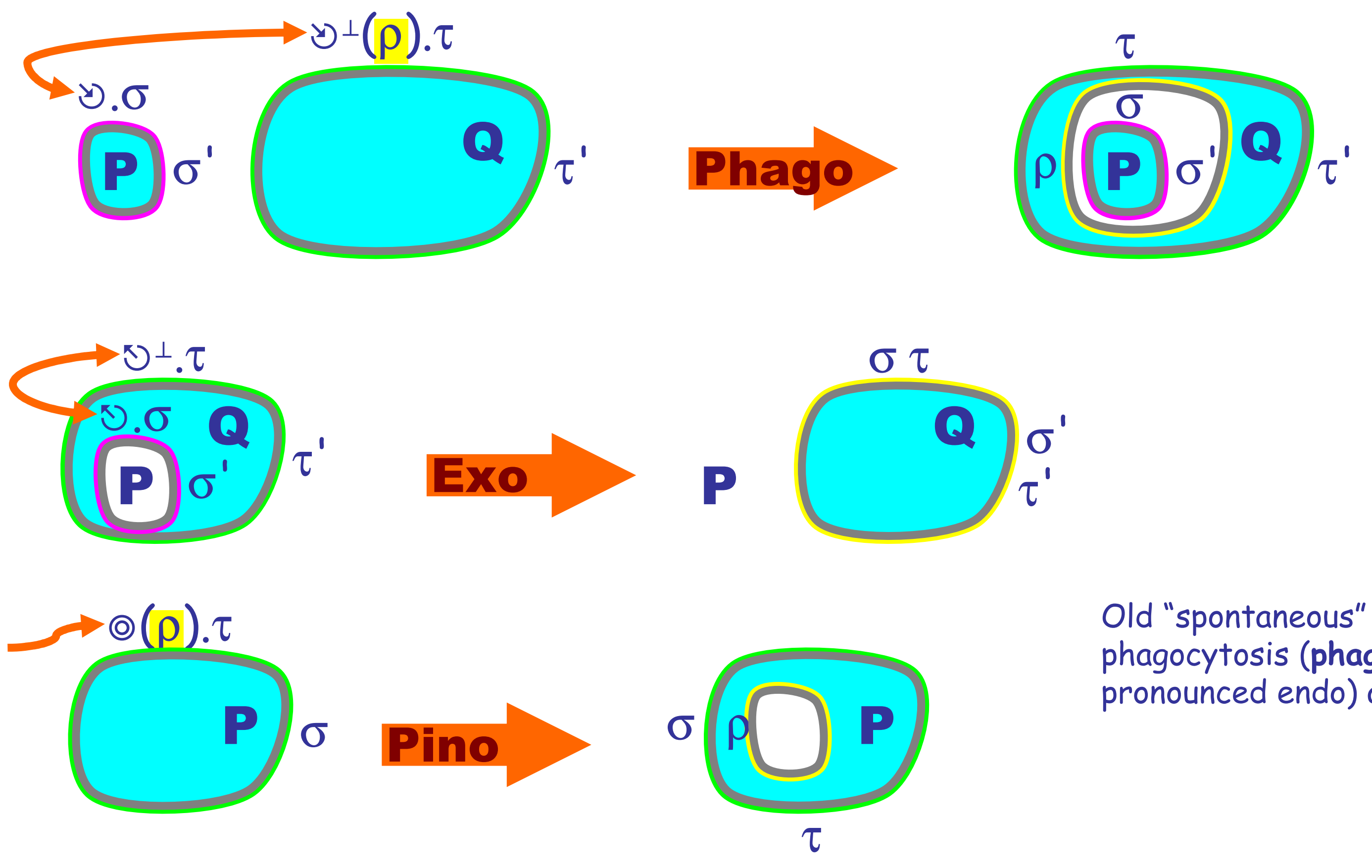
$$P \rightarrow Q \Rightarrow \sigma \llparenthesis P \rrparenthesis \rightarrow \sigma \llparenthesis Q \rrparenthesis$$

This is the whole semantics, except for the effects of individual actions.

Brane Reactions

actions a ::= … | $⟳_n$ | $⟳^{\perp}_n(\rho)$ | $⟲_n$ | $⟲^{\perp}_n$ | $⊚(\rho)$ phago ⟳, exo ⟲, pino ⊚

coordination tags sometimes omitted



Old "spontaneous" **endo** splits into phagocytosis (**phago**, often still pronounced endo) and pinocytosis (**pino**).

…formally…

Phago $\circlearrowright_n.\sigma|\sigma'\llparenthesis P\rrparenthesis \circ \circlearrowright^{\perp}_n(\rho).\tau|\tau'\llparenthesis Q\rrparenthesis \rightarrow \tau|\tau'\llparenthesis \rho\llparenthesis\sigma|\sigma'\llparenthesis P\rrparenthesis\rrparenthesis\circ Q\rrparenthesis$

Exo $\circlearrowleft^{\perp}_n.\tau|\tau'\llparenthesis \circlearrowleft_n.\sigma|\sigma'\llparenthesis P\rrparenthesis\circ Q\rrparenthesis \rightarrow P \circ \sigma|\sigma'|\tau|\tau'\llparenthesis Q\rrparenthesis$

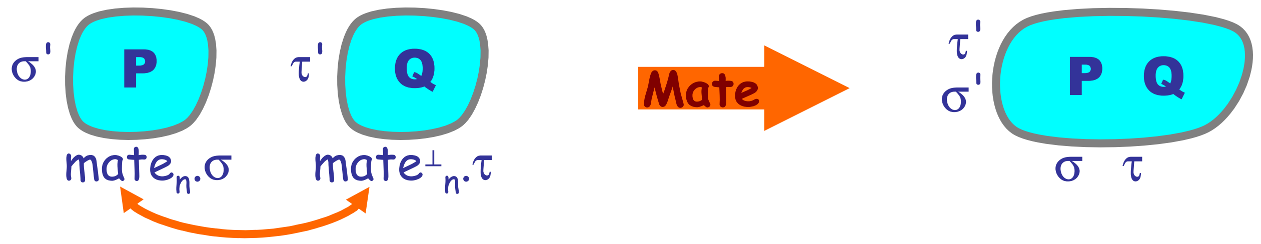
Pino $\circledcirc(\rho).\sigma|\sigma'\llparenthesis P\rrparenthesis \rightarrow \sigma|\sigma'\llparenthesis \rho\llparenthesis\diamond\rrparenthesis\circ P\rrparenthesis$

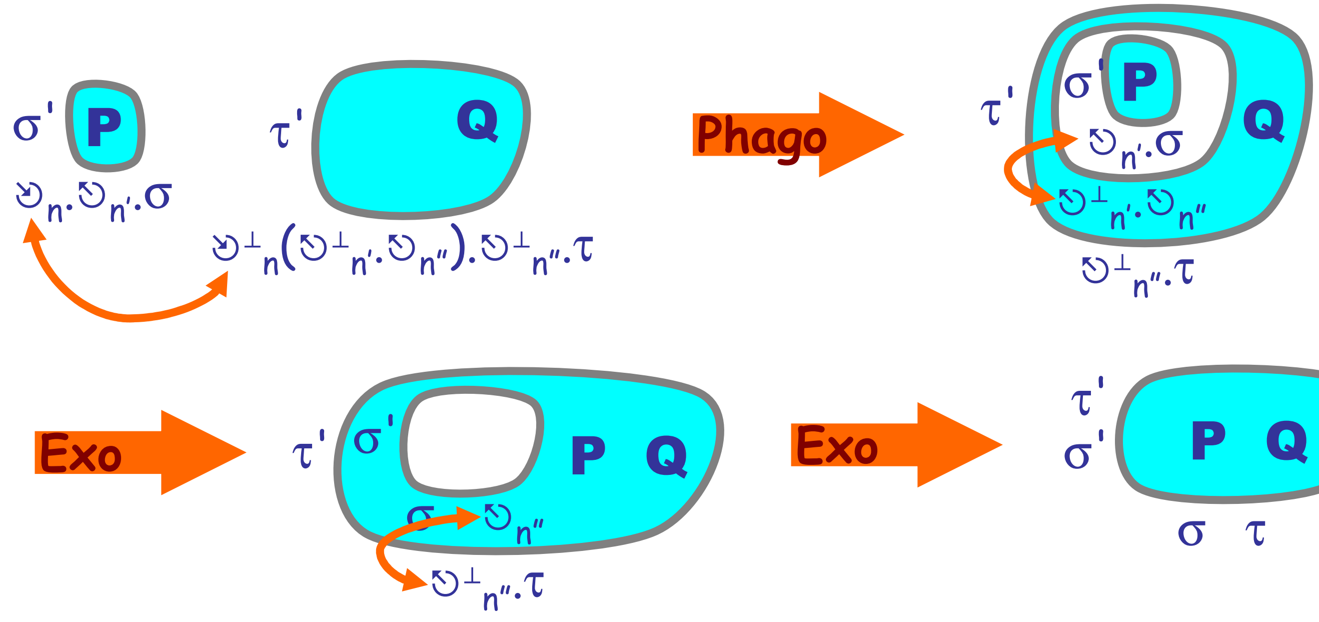
N.B.: the parity of nesting of P and Q is preserved;
this makes the reactions preserve bitonality.

Abbreviations: Mate

Mate $\mathrm{mate}_n.\sigma = \circlearrowright_n.\circlearrowleft_{n'}.\sigma$

$\mathrm{mate}^{\perp}_n.\tau = \circlearrowright^{\perp}_n(\circlearrowleft^{\perp}_{n'}.\circlearrowleft_{n''}).\circlearrowleft^{\perp}_{n''}.\tau$





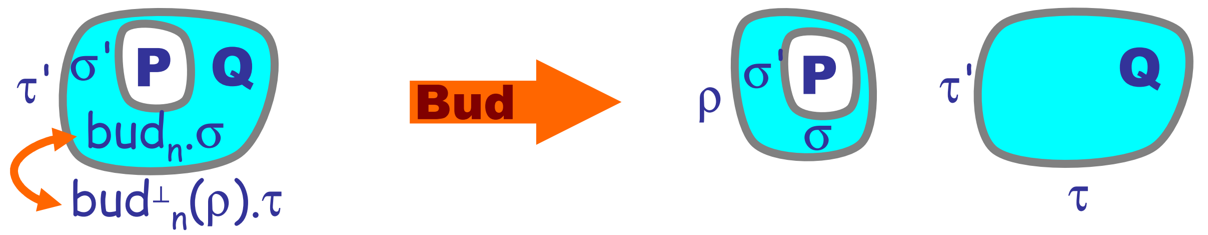
Abbreviations: Bud

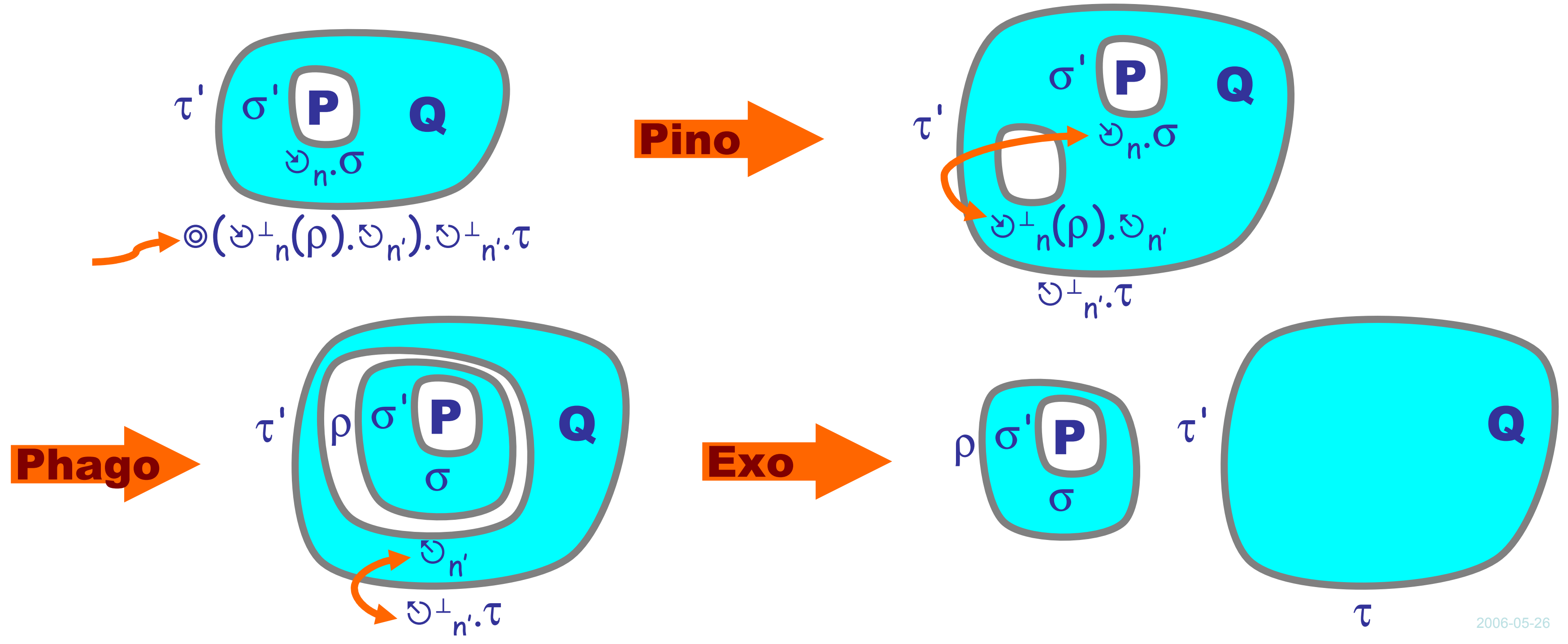
Bud

$$\mathrm{bud}_n.\sigma = \circlearrowleft_n.\sigma$$

$$\mathrm{bud}^{\perp}_n(\rho).\tau = \circledcirc(\circlearrowleft^{\perp}_n(\rho).\circlearrowright_{n'}).\circlearrowright^{\perp}_{n'}.\tau$$

A budding version of old "spontaneous" mito, to avoid arbitrary splits. Follows the pattern of inverse-mate.



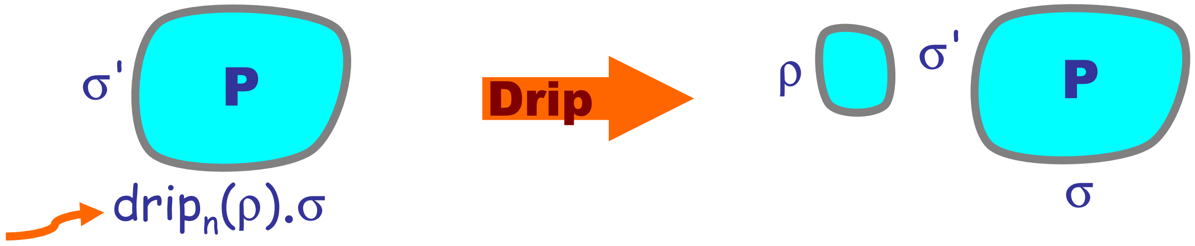


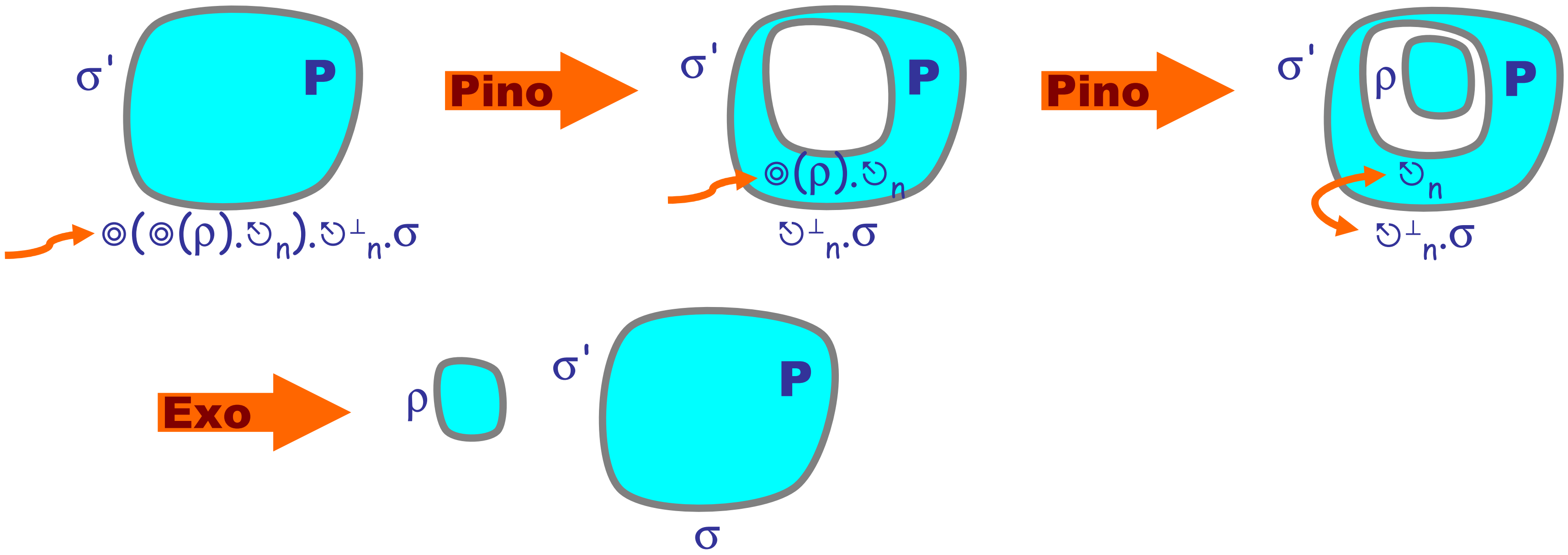
τ'
σ'
P
Q
$\circlearrowleft_n.\sigma$
$\circledcirc(\circlearrowleft^{\perp}_n(\rho).\circlearrowright_{n'}).\circlearrowright^{\perp}_{n'}.\tau$
Pino
τ'
σ'
P
Q
$\circlearrowleft_n.\sigma$
$\circlearrowleft^{\perp}_n(\rho).\circlearrowright_{n'}$
$\circlearrowright^{\perp}_{n'}.\tau$
Phago
τ'
ρ
σ'
P
σ
Q
$\circlearrowright_{n'}$
$\circlearrowright^{\perp}_{n'}.\tau$
Exo
ρ
σ'
P
σ
τ'
Q
τ

Abbreviations: Drip

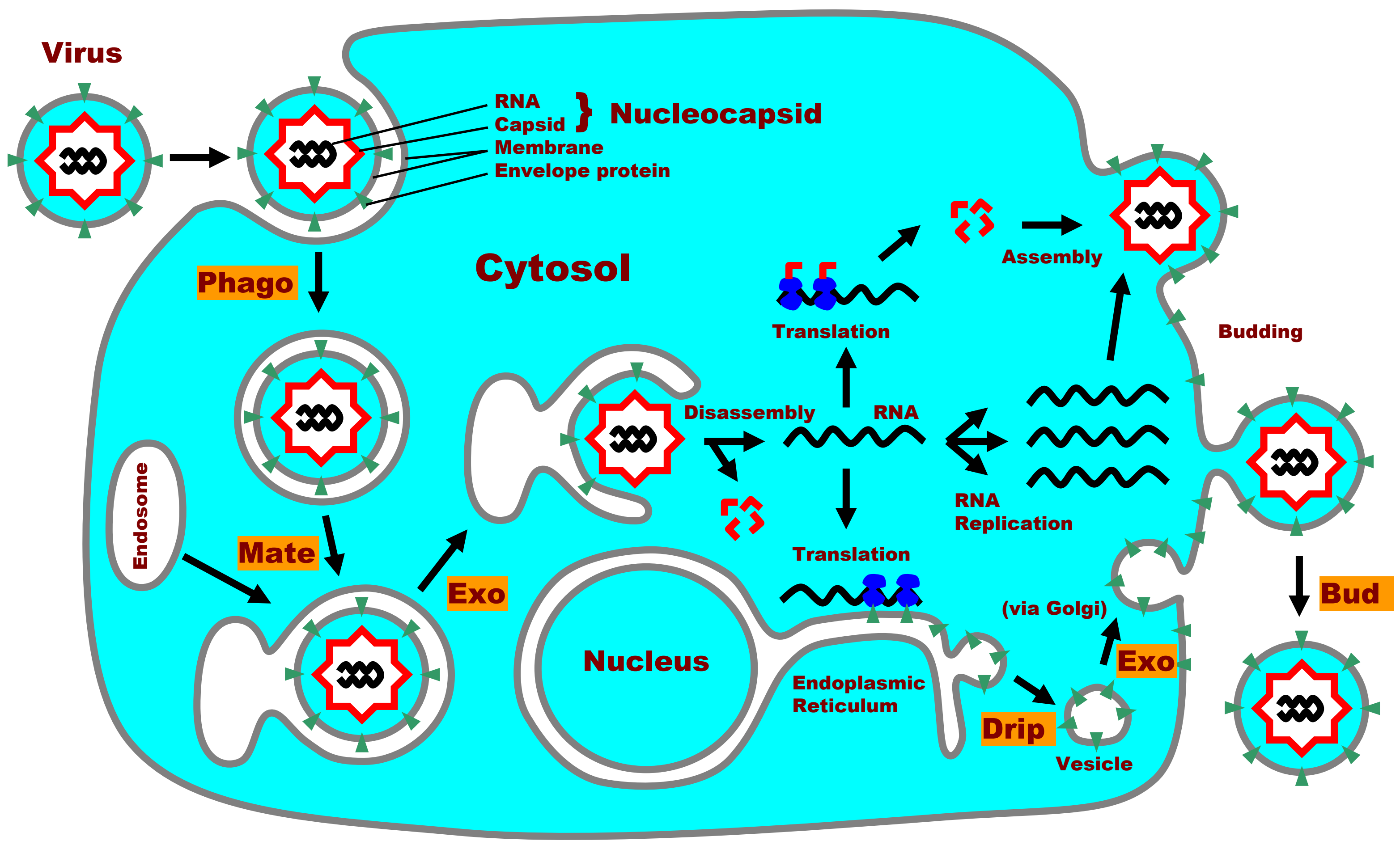
Drip $\mathrm{drip}_n(\rho).\sigma = ⊚(⊚(\rho).⟲_n).⟲^{\perp}_n.\sigma$

A zero-expelled-membranes version of old "spontaneous" mito, to avoid arbitrary splits. Follows the pattern of inverse-mate.





Ex: Viral Reproduction



Infection | Replication | Progeny

[MBC p.279]

annotated

Ex: Viral Infection

$$\overbrace{\text{↘}.\text{↺}⦅\mathit{nucap}⦆}^{\mathit{virus}} \circ \overbrace{\underbrace{*\text{↘}^{\perp}(\text{mate})}_{\mathit{membrane}}⦅\underbrace{*\text{mate}^{\perp} \mid *\text{↺}^{\perp}⦅\diamond⦆}_{\mathit{endosome}} \circ \mathit{cytosol}⦆}^{\mathit{cell}} \longrightarrow \text{Phago}$$

$$\underbrace{*\text{↘}^{\perp}(\text{mate})}_{\mathit{membrane}}⦅\underbrace{\text{mate}⦅\text{↺}⦅\mathit{nucap}⦆⦆}_{\mathit{vesicle}} \circ \underbrace{*\text{mate}^{\perp} \mid *\text{↺}^{\perp}⦅\diamond⦆}_{\mathit{endosome}} \circ \mathit{cytosol}⦆ \longrightarrow \text{Mate}$$

$$\underbrace{*\text{↘}^{\perp}(\text{mate})}_{\mathit{membrane}}⦅\underbrace{*\text{mate}^{\perp} \mid *\text{↺}^{\perp}⦅\text{↺}⦅\mathit{nucap}⦆⦆}_{\mathit{endosome}} \circ \mathit{cytosol}⦆ \longrightarrow \text{Exo}$$

$$\underbrace{*\text{↘}^{\perp}(\text{mate})}_{\mathit{membrane}}⦅\underbrace{*\text{mate}^{\perp} \mid *\text{↺}^{\perp}⦅\diamond⦆}_{\mathit{endosome}} \circ \mathit{nucap} \circ \mathit{cytosol}⦆$$

Ex: Viral Progeny

Assume:

$$nucap \circ cytosol \longrightarrow\!\!\!\!\longrightarrow nucap^{n} \circ envelope\text{-}vesicle^{m} \circ cytosol'$$

by available cellular machinery

Then:

$$\overbrace{*\circlearrowleft^{\perp}\llparenthesis \underbrace{\circlearrowleft.bud^{\perp}(\circlearrowright.\circlearrowleft)\llparenthesis\diamond\rrparenthesis}_{envelope\text{-}vesicle} \circ \underbrace{*bud|\sigma\llparenthesis vRNA\rrparenthesis}_{nucap} \circ cytosol''\rrparenthesis}^{cell} \xrightarrow{\ \ } \text{Exo}$$

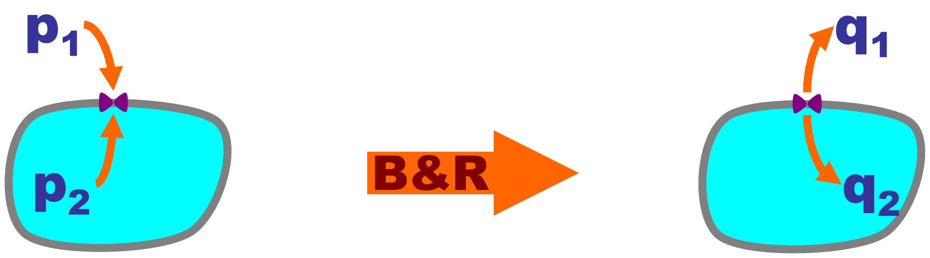
$$*\circlearrowleft^{\perp}|\underbrace{bud^{\perp}(\circlearrowright.\circlearrowleft)}_{envelope}\underbrace{\llparenthesis *bud|\sigma\llparenthesis vRNA\rrparenthesis}_{nucap} \circ cytosol''\rrparenthesis \xrightarrow{\ \ } \text{Bud}$$

$$\underbrace{*\circlearrowleft^{\perp}\llparenthesis cytosol''\rrparenthesis}_{cell} \circ \underbrace{\circlearrowright.\circlearrowleft\llparenthesis nucap\rrparenthesis}_{virus}$$

Molecules

Brane-Molecule Reactions (Cartoons)

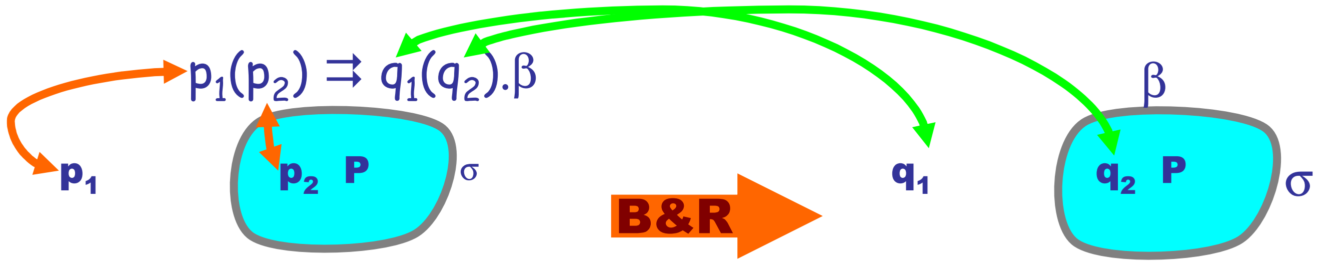
With *molecule multisets* **p**,**q**:



Molecules

We now add *molecules* to the model:

systems	$P,Q ::= \ldots \mid m$	$m \in M$ molecules
	$p,q ::= m_1 \circ \ldots \circ m_k$	molecule multisets
actions	$a ::= \ldots \mid p_1(p_2) \rightrightarrows q_1(q_2)$	bind&release



This single operation can essentially account for the whole Protein Machine, including its interactions with membranes. Except that, one must add some form of protein complexation, either as in BioSPi by adding restriction, or as in κ-calculus by adding complex molecules.

B&R Reaction

B&R $p_1 \circ p_1(p_2) \rightrightarrows q_1(q_2).\alpha|\sigma(\!| p_2 \circ P |\!) \longrightarrow q_1 \circ \alpha|\sigma(\!| q_2 \circ P |\!)$

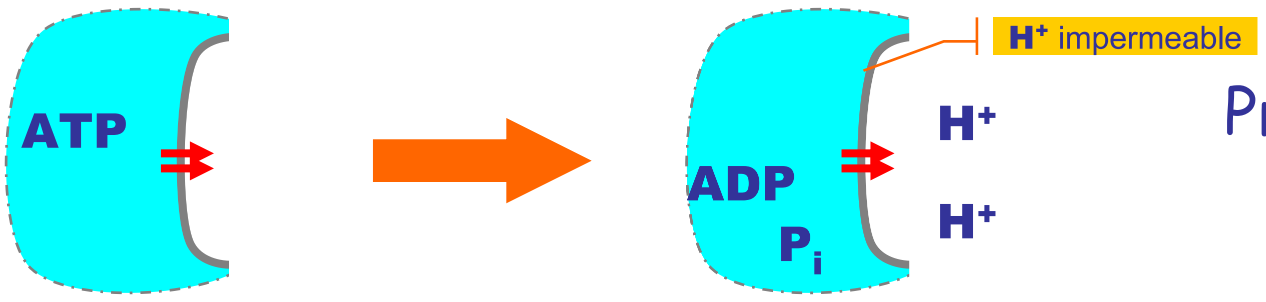
(multiset rewriting, inside and outside membranes)

Simple bindings and releases - "$\diamond(\diamond)$" is omitted:

$m(\diamond)\rightrightarrows$	bind out	$\rightrightarrows m(\diamond)$	release out
$\diamond(m)\rightrightarrows$	bind in	$\rightrightarrows \diamond(m)$	release in

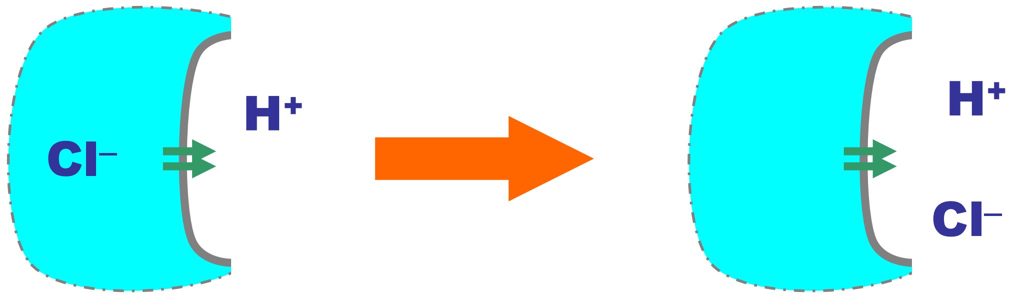
Ex: Molecular Pumps and Channels

E.g. plant vacuole (white).

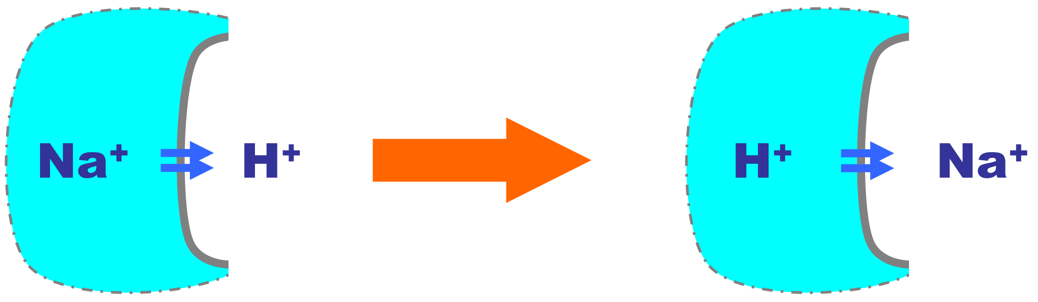


Proton Pump

ATP charges up the vacuole with H^+. Several other pumps work off that charge.



Ion Channel



Proton Antiporter

A plant vacuole membrane has all those things on it.

...

$\text{ProtonPump} = {*}\ \text{ATP}(\diamond) \rightrightarrows \text{ADP}\circ\text{P}_i(\text{H}^+\circ\text{H}^+)$

$\text{IonChannel} = {*}\ \text{Cl}^-(\text{H}^+) \rightrightarrows \diamond(\text{H}^+\circ\text{Cl}^-)$

$\text{ProtonAntiporter} = {*}\ \text{Na}^+(\text{H}^+) \rightrightarrows \text{H}^+(\text{Na}^+)$

$\text{PlantVacuole} =$

$\quad \text{ProtonPump} \mid \text{IonChannel} \mid \text{ProtonAntiporter} \ (\!|\diamond|\!)$

Hence this reaction notation, $\rightrightarrows$, is "like" chemical reaction notation, $\rightarrow$, but talking about both sides on a membrane at once.

(N.B. no built-in conservation of mass in either case.)

Special Cases of B&R

Chemical reaction catalysis (inside a compartment)

$p \longrightarrow q \quad \triangleq \quad * \, p(\diamond) \rightrightarrows q(\diamond) (\!(\,)\!)$

$p \Longleftrightarrow q \quad \triangleq \quad p \longrightarrow q \circ q \longrightarrow p$

E.g. peptide bond between two aminoacids $R^1 R^2$:

$R^1\text{-COOH} \circ H_2N\text{-}R^2 \longrightarrow R^1\text{-CO-HN-}R^2 \circ H_2O$

Compartment conditions (on the membrane of a compartment)

$p \dot{\rightarrow} q \quad \triangleq \quad * \, \diamond(p) \rightrightarrows \diamond(q)$

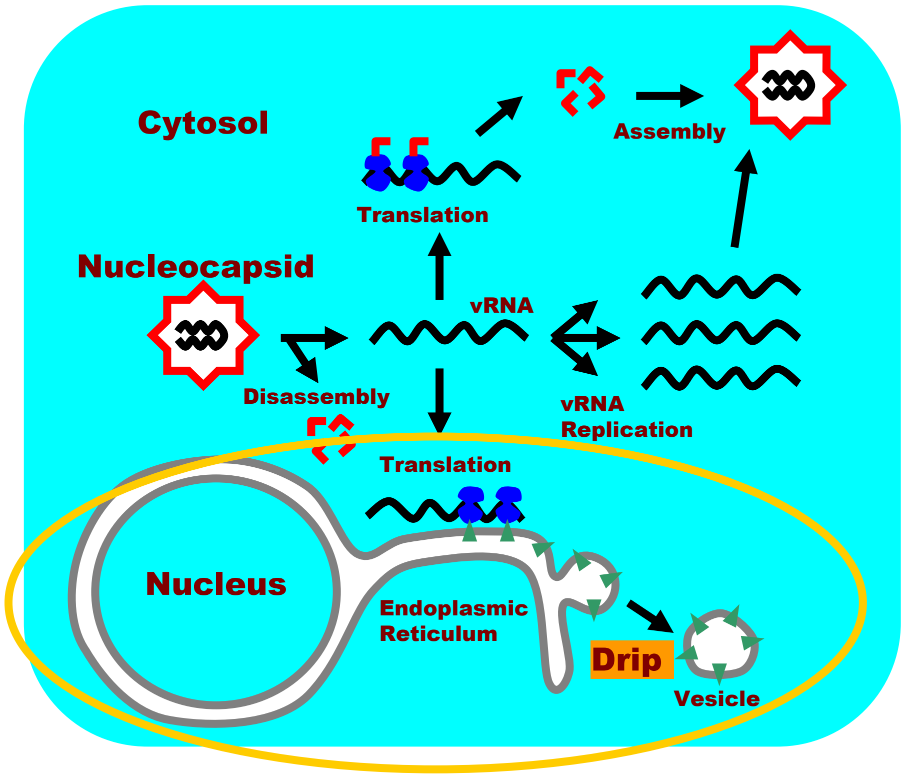
$p \dot{\rightarrow} q | \sigma (\!(P)\!)$ Condition affecting P

E.g. a condition-driven reaction:

$p \dot{\rightarrow} q | \sigma (\!(p)\!) \Rightarrow p \dot{\rightarrow} q | \sigma (\!(q)\!)$

Ex: Virus Replication

$$nucap \circ cytosol \longrightarrow\!\!\!\longrightarrow nucap^{n} \circ envelope\text{-}vesicle^{m} \circ cytosol'$$



$$ER \triangleq {*}vRNA(\diamond) \rightrightarrows vRNA(\diamond).\ drip(\circlearrowleft.bud^{\perp}(\circlearrowleft.\circlearrowleft))(\!|Nucleus|\!)$$

when triggered by vRNA

exo to cell membrane

nucap budding receptor

virus membrane

envelope-vesicle

(See paper for the other two vRNA pathways)

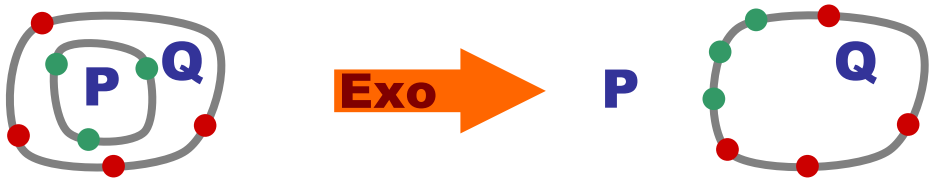
Other

Adding Frills to the Framework

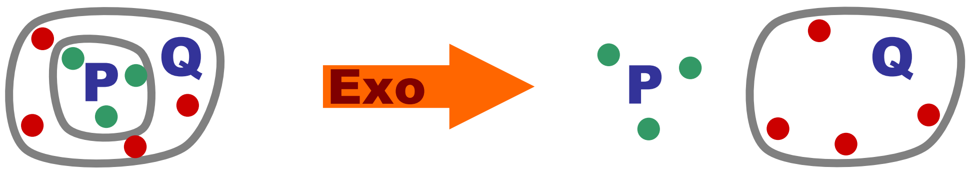
- So far, purely combinatorial:
  - No name binding, channel creation, communication…
  - Closer to combinatorial flavor of protein interactions
  - Goes a long way: do not try to extend needlessly.
- But one can easily add all that, and more:
  - CCS-style communication
    - Diffusion of molecules on cellular membrane
  - BioAmbients-style communication
    - Diffusion of molecules across cellular membrane
  - BioAmbients-like mobility
    - Non-bitonal
  - π-style restriction
- We have a framework where we can plug&play a rich set of interactions, while supporting compartments.

"On Brane" vs. "In Brane"

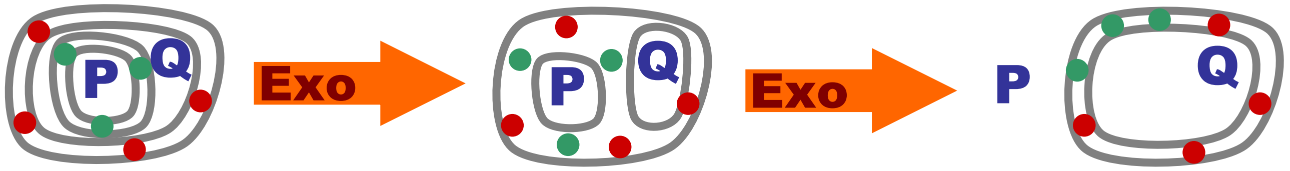
Why do we need brane calculi, again?



Original "on brane" Exo of Brane Calculus



"In brane" encoding (e.g. in BioAmbients or SMBL) goes wrong



"Ball bearing" encoding; best we can do "in brane"

Awkward encoding. And all kinds of things can go wrong in the intermediate state.

- One cannot easily represent the Exo reaction in BioAmbients or any such compartment-based calculus, nor can one easily add it as a new primitive!
- But we can add BioAmbients-like In/Out out to Brane Calculi if we want to.

A SPiM-Conservative Brane Extension

Membranes

We want to be "upward compatible" from π-calculus/SPiM.

Simplifying idea: a π-process is a process on the surface of a membrane; in π-calculus there is implicitly a single membrane where all processes live. (C.f. a π-process could instead be seen as a process "free-swimming" inside a membrane; this turns out to be more complicated because then one must have both free-swimming activities and membrane-bound activities, and some elaborate way of connecting the two.)

We then introduce nested and contiguous membranes. All processes are on the surface of membranes, but a "free-swimming" process can be obtained as process on the surface of an empty membrane.

Processes can communicate only on the same membrane, but membrane can merge or split to change communication medium. Main advantage: no need to add up/down/across communication: π communication is unchanged.

A Stochastic Brane Calculus

A ::=	delay@r | ?n(m) | !n(m)	s-π actions
	| fizz@r(P) | fuse!n | fuse?n | eat!n | eat?n(P)	brane actions
	| drip@r(P) | mate!n | mate?n | bud!n | bud?n(P)	definable actions

P ::=	Processes
0 | A.P | P+P | (P | P) | P*| (νn)P	standard π processes (with extra actions)

S ::=	Systems
◇	empty
| P⦇S⦈	brane with surface P and contents S
| S∘S	contiguous subsystems
| S*	system replication
| (νn)S	system restriction

Brane Reductions

see p.4,6

Reduction Context

$_C\lfloor X \rfloor = X+Q \mid R$ (any (νn) must be extruded first)

Reduction Residual

$_{\underline{C}}\lfloor X \rfloor = X \mid R$

$$\begin{array}{lcl}
_C\lfloor \mathrm{fizz}(R).\ P \rfloor \llparenthesis S \rrparenthesis & \rightarrow & _{\underline{C}}\lfloor P \rfloor \llparenthesis R \llparenthesis\rrparenthesis \circ S \rrparenthesis \\
_C\lfloor \mathrm{fuse}?n.\ P \rfloor \llparenthesis {}_D\lfloor \mathrm{fuse}!n.\ Q \rfloor \llparenthesis S \rrparenthesis \circ T \rrparenthesis & \rightarrow & ({}_{\underline{C}}\lfloor P \rfloor \mid {}_{\underline{D}}\lfloor Q \rfloor) \llparenthesis T \rrparenthesis \circ S \\
_C\lfloor \mathrm{eat}!n.\ P \rfloor \llparenthesis S \rrparenthesis \circ {}_D\lfloor \mathrm{eat}?n(R).\ Q \rfloor \llparenthesis T \rrparenthesis & \rightarrow & _{\underline{D}}\lfloor Q \rfloor \llparenthesis R \llparenthesis {}_{\underline{C}}\lfloor P \rfloor \llparenthesis S \rrparenthesis \rrparenthesis \circ T \rrparenthesis \\
_C\lfloor \mathrm{drip}(R).\ P \rfloor \llparenthesis S \rrparenthesis & \rightarrow & R \llparenthesis\rrparenthesis \circ {}_{\underline{C}}\lfloor P \rfloor \llparenthesis S \rrparenthesis \\
_C\lfloor \mathrm{mate}!n.\ P \rfloor \llparenthesis S \rrparenthesis \circ {}_D\lfloor \mathrm{mate}?n.\ Q \rfloor \llparenthesis T \rrparenthesis & \rightarrow & ({}_{\underline{C}}\lfloor P \rfloor \mid {}_{\underline{D}}\lfloor Q \rfloor) \llparenthesis S \circ T \rrparenthesis \\
_C\lfloor \mathrm{bud}?n(R).\ P \rfloor \llparenthesis {}_D\lfloor \mathrm{bud}!n.\ Q \rfloor \llparenthesis S \rrparenthesis \circ T \rrparenthesis & \rightarrow & _{\underline{C}}\lfloor P \rfloor \llparenthesis T \rrparenthesis \circ R \llparenthesis {}_{\underline{D}}\lfloor Q \rfloor \llparenthesis S \rrparenthesis \rrparenthesis
\end{array}$$

SPiM Syntax Extensions

Type ::=
- … the existing syntax
- | sys(Type$_1$, …, Type$_N$)

Process ::=
- … exactly the existing syntax

System ::=

brane Process ‘{’ System ‘}’	
(System$_1$, … , System$_N$)	N>=0
if Value then System else System	
int of System	
Name(Value$_1$, … , Value$_N$)	N>=0
(Declaration$_1$ … Declaration$_N$ System)	N>0

Definition ::=
- Name(Pattern$_1$, … , Pattern$_N$) = Process
- | sys Name(Pattern$_1$, … , Pattern$_N$) = System

Declaration ::=
- …
- | run sys System$_1$, … , System$_N$ N>=1

(N.B. the extra “sys” remove parsing ambiguities; we do not want any parsing point where either a Process or a System can start).

Action ::=	(brane calculus notation:)
delay@rate	
!Channel{(Value$_1$, … , Value$_N$)}	
?Channel{(Pattern$_1$, … , Pattern$_N$)}	
fizz{@rate}(Process)	⊚(ρ)
fuse!Channel	$↺_n$
fuse?Channel	$↺_n^\perp$
eat!Channel	$↻_n$
eat?Channel(Process)	$↻_n^\perp$(ρ)
drip{@rate}(Process)	drip$_n$(ρ)
mate!Channel	mate$_n$
mate?Channel	mate$_n^\perp$
bud!Channel	bud$_n$
bud?Channel(Process)	bud$_n^\perp$(ρ)

(N.B. fizz and drip without a rate are instant actions).

Ex: Virus

```
new n@1.0:chan  (* dummy global for all interactions *)

let sys virus() = brane eat!n; fuse!n  {nucap()}
and sys nucap() = brane (replicate bud!n | X()) {vRNA()}
and sys vRNA() = ...
and X() = ...

let sys cell() = brane (replicate eat?n(mate!n) | replicate fuse?n) {cytosol()}
and sys cytosol() = (endosome(), Z())
and sys endosome() = brane (replicate mate?n | replicate fuse?n) {()}
and sys Z() = ...

let viralEnvelope() = bud?n(eat!n; fuse!n)
and sys envelopeVesicle() = brane fuse!n; viralEnvelope() {()}

run sys virus(), cell()
```

see p.7 of Brane Calculus paper

Ex: Bind and Release

see p.11

```
let activity(n:chan) = mate!n + fuse!n
let sys particle(n:chan) = brane activity(n) {()}

(* bindOut(n) = mate?n        releaseOut(n) = drip(activity(n))
   bindIn(n) = fuse?n         releaseIn(n) = fizz(activity(n))   *)

let protonPump() =
  replicate mate?ATP; drip(activity(ADP)); drip(activity(Pi));
    fizz(activity(Proton)); fizz(activity(Proton))

let ionChannel() =
  replicate mate?ClIon; fuse?Proton; drip(activity(Proton)); drip(activity(ClIon))

let protonAntiporter() =
  replicate mate?NaIon; fuse?Proton; drip(activity(Proton)); fizz(activity(NaIon))

let sys plantVacuole() =
  brane protonPump() | ionChannel() | protonAntiporter() {()}

run sys plantVacuole(), 10 of (particle(ATP), particle(ClIon), particle(NaIon))
```

Ex: Encodings

see p.6

```
let MateE(n:chan, cont:proc) =
  eat!n; fuse!n; cont()
and MateQ(n:chan, cont:proc) =
  eat?n(fuse?n; fuse!n); fuse?n; cont()

let BudE …
```

Summary

- Brane Calculi
  - Calculi with membranes are needed to model “the whole system”.
  - E.g. a full virus invasion pathway.

- Turning them into (simulation) languages
  - Too early to tell precisely what primitives are useful/necessary for large/realistic modeling of compartments.
  - But the basis for such experimentation are ready.

Q?