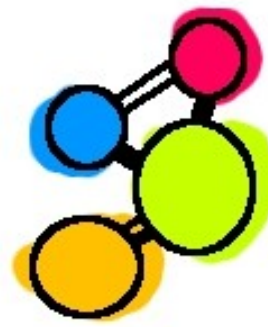
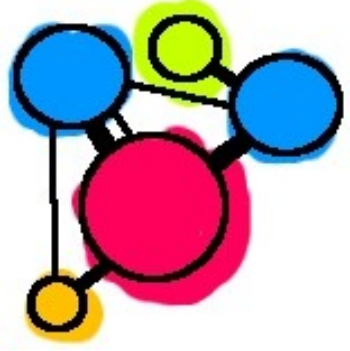


# Sound Bisimulations for Higher-Order Distributed Process Calculus

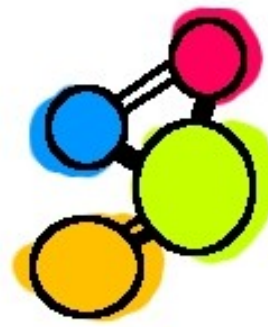
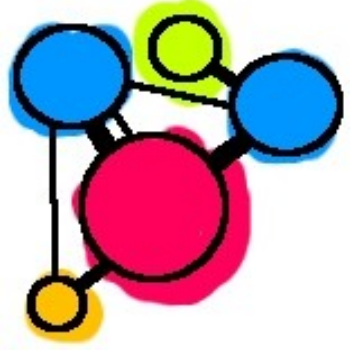
Adrien PIÉRARD, Eijiro SUMII  
Tohoku University



# Motivation

- Today's computing is **distributed** (mobile computing, cloud computing, etc)
- **Running programs can be moved** or stored as data

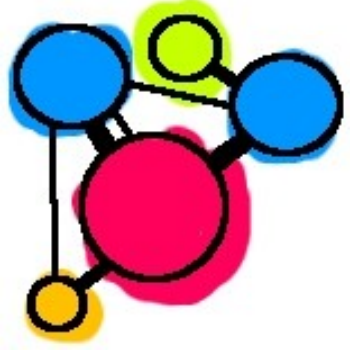
It is hard to prove that such systems  
have **no bug**  
(i.e. behave as expected)



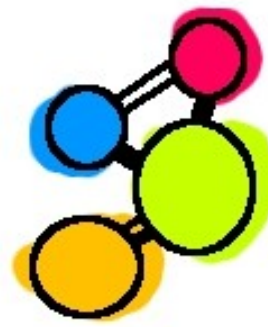
# Real-Life Examples

**Distribution** and **program passing** are commonly used with

- *Load balancing* (distribution of computations across computers)
- *Fault tolerance* (resumption of computations from a saved state after a crash)
- *Remote execution* (webmails, web video players, smartphone applications)
- You and your laptop/smartphone!

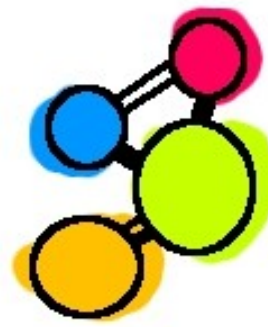
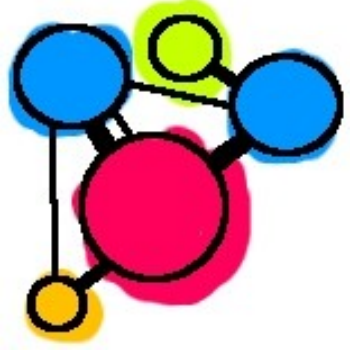


# Reasoning About Distributed Systems



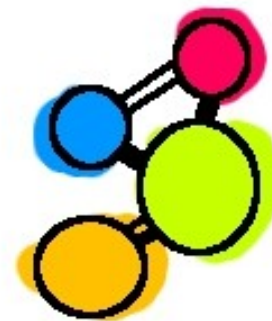
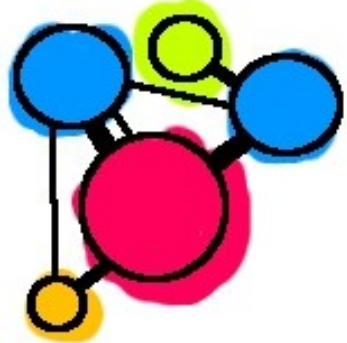
**Correctness** of a system / program can be stated as an **equivalence** (e.g. **reduction-closed barbed equivalence**) by comparing it to its specification (in a model like a **process calculus**)

E.g. a fault-tolerant program behaves *functionally* like its "ideal" infallible version



# Outline

- Modeling higher-order and distribution
- Correctness as equivalence
- Environmental bisimulations
- Example
- Conclusion

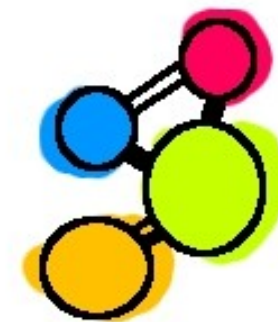
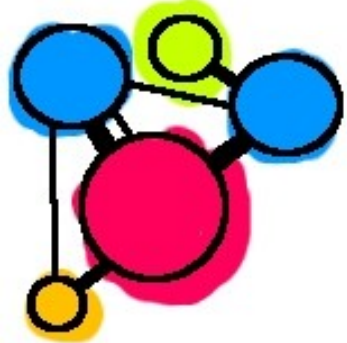


# Our Model: HO $\pi$ P

The **higher-order**  $\pi$ -calculus with **passivation**  
(HO $\pi$ P) [Lenglet et al. 09]

- A dialect of the  $\pi$ -calculus, with
  - **Process-passing**
  - **Distribution**

Can express various behaviours of distribution



# Higher-Order in HO $\pi$ P

Output

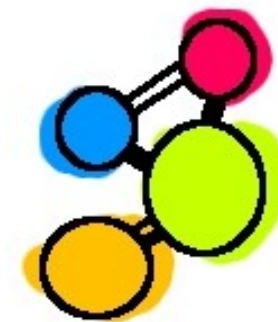
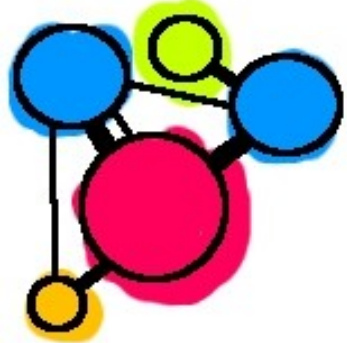
$$\bar{a}\langle P \rangle . Q \xrightarrow{\bar{a}\langle P \rangle} Q$$

Input

$$a(X) . (X \mid X) \xrightarrow{a(P)} P \mid P$$

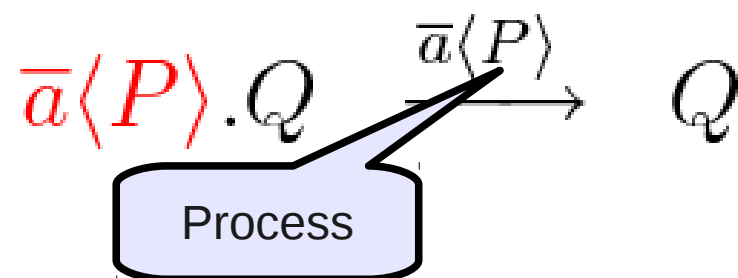
Reaction

$$\bar{a}\langle P \rangle \mid a(X) . (X \mid X) \xrightarrow{\tau} 0 \mid P \mid P$$

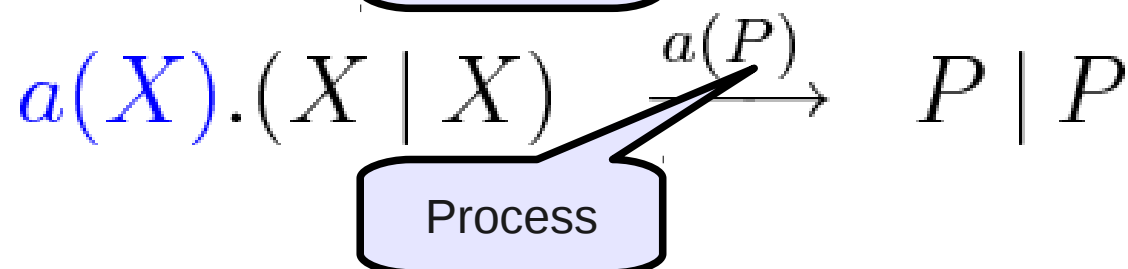


# Higher-Order in HO $\pi$ P

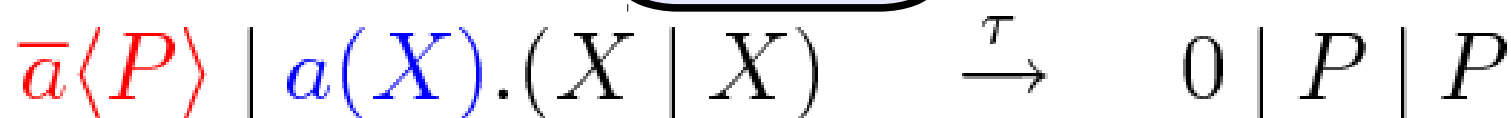
Output

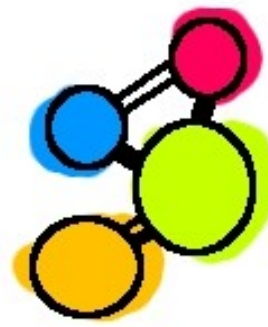
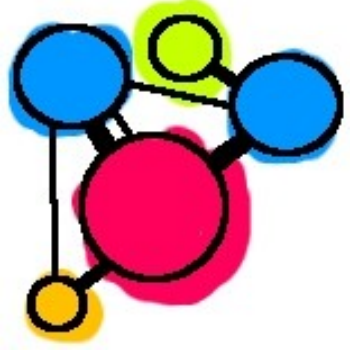


Input



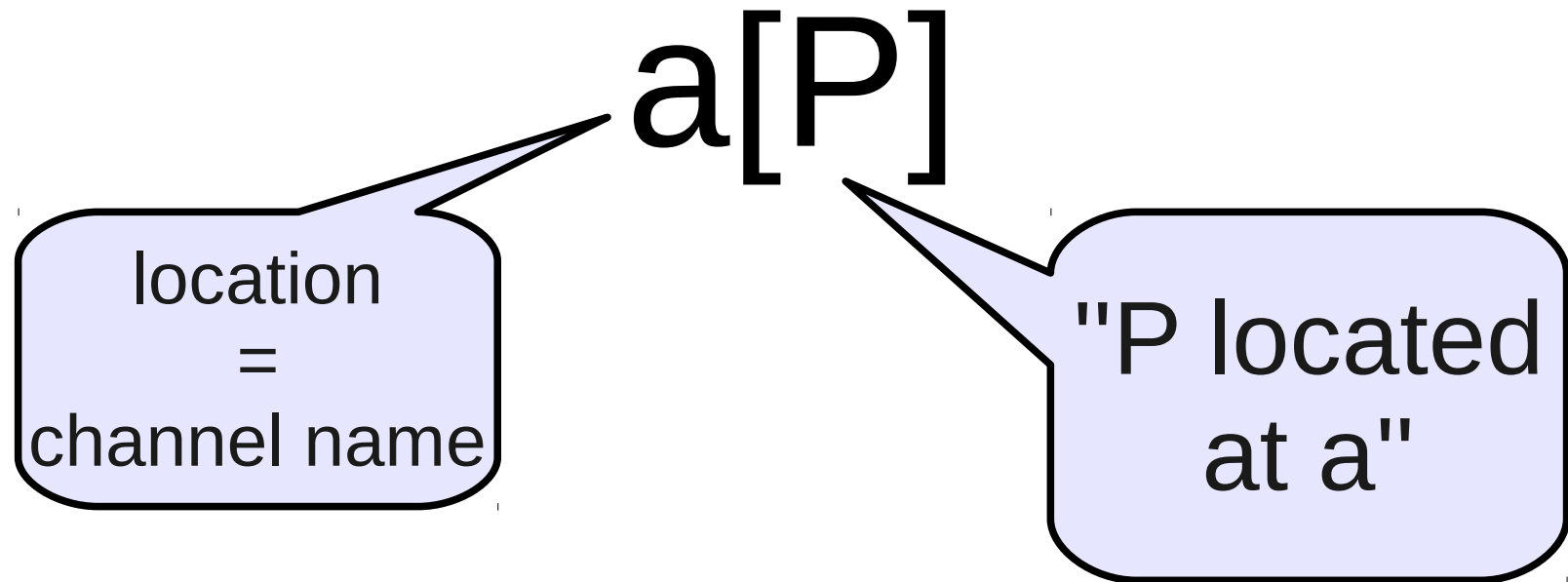
Reaction

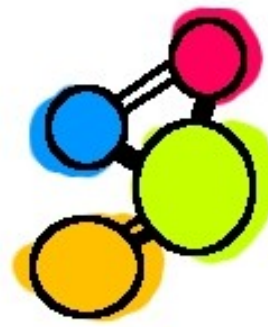
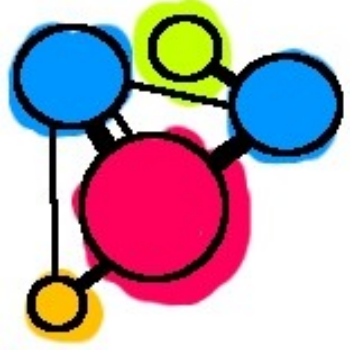




# Distribution in $\text{HO}\pi\text{P}$

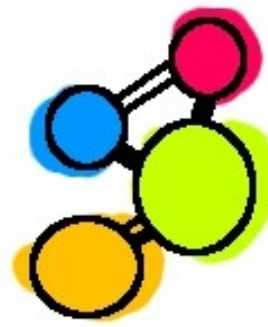
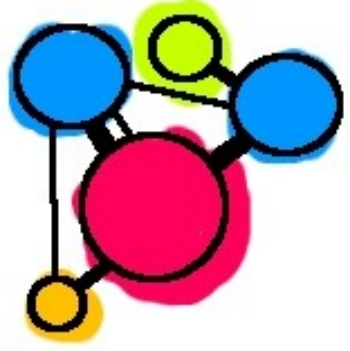
Distribution: location dependent behaviour





# Distribution in $\text{HO}\pi\text{P}$

$$\frac{P \xrightarrow{\alpha} P'}{a[P] \xrightarrow{\alpha} a[P']}^{\text{TRANSP}}$$

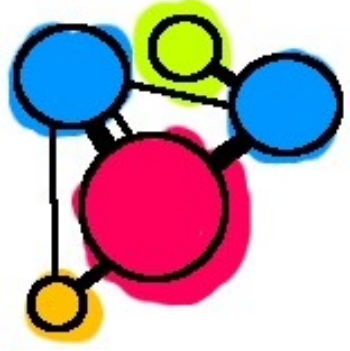


# Distribution in $\text{HO}\pi\text{P}$

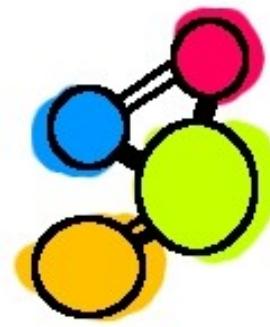
---

$$a[P] \xrightarrow{\bar{a} \langle P \rangle} 0 \quad \text{PASSIV}$$

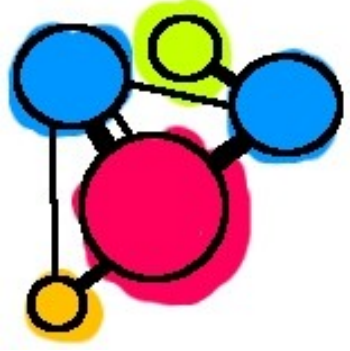
higher-order  
output



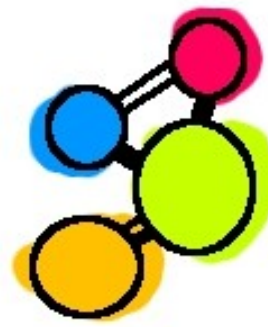
# Failure in HO $\pi$ P



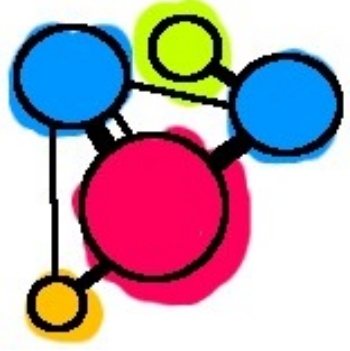
$$a[\textcolor{red}{P}] \mid a(X).\overline{\text{fail}} \xrightarrow{\tau} 0 \mid \overline{\text{fail}}$$



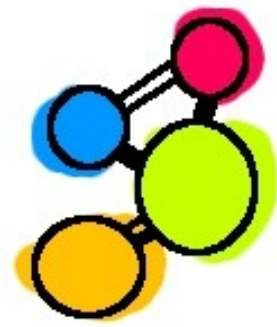
# Migration in HO $\pi$ P



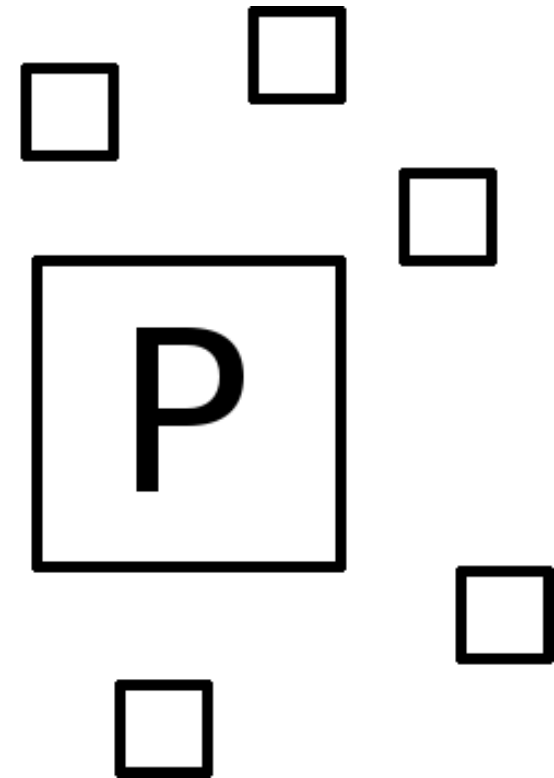
$$b[\textcolor{red}{P}] \mid b(\textcolor{blue}{X}).c[\textcolor{black}{X}] \xrightarrow{\tau} 0 \mid c[\textcolor{red}{P}]$$

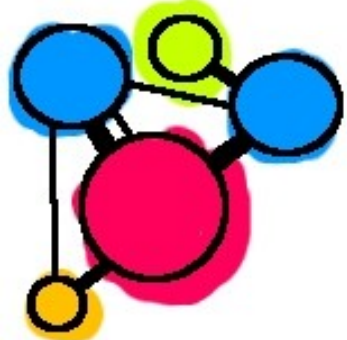


# Example

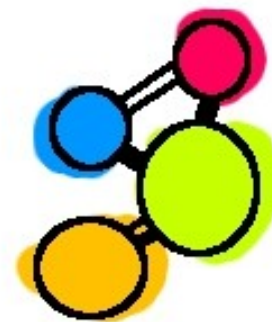


$$\nu f.(f[P] \mid !f(X).f[X])$$



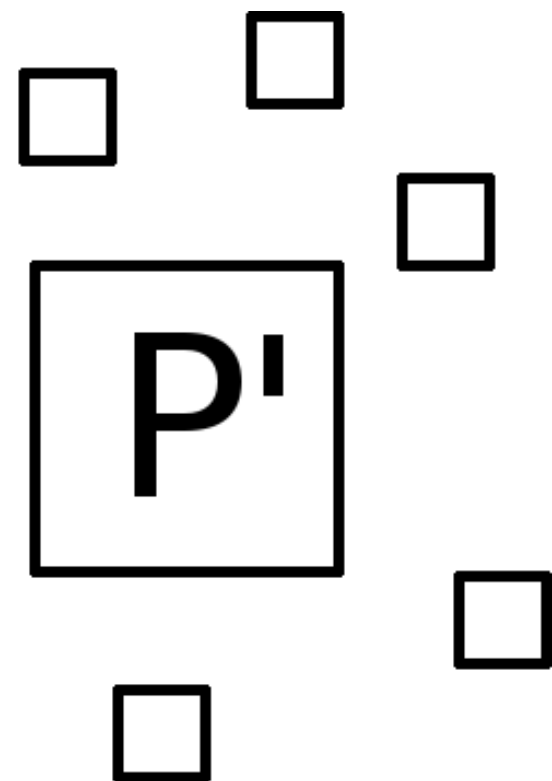


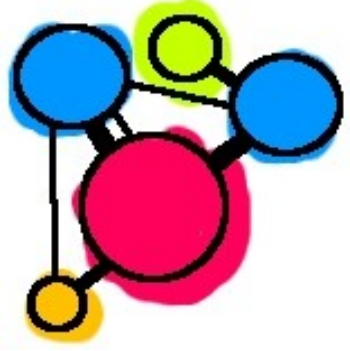
# Example



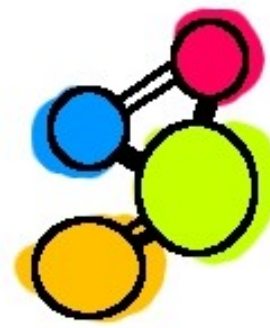
$$\xrightarrow{\alpha} \nu f. (f[P] \mid !f(X).f[X])$$

$$\nu f. (f[\textcolor{red}{P}'] \mid !f(X).f[X])$$

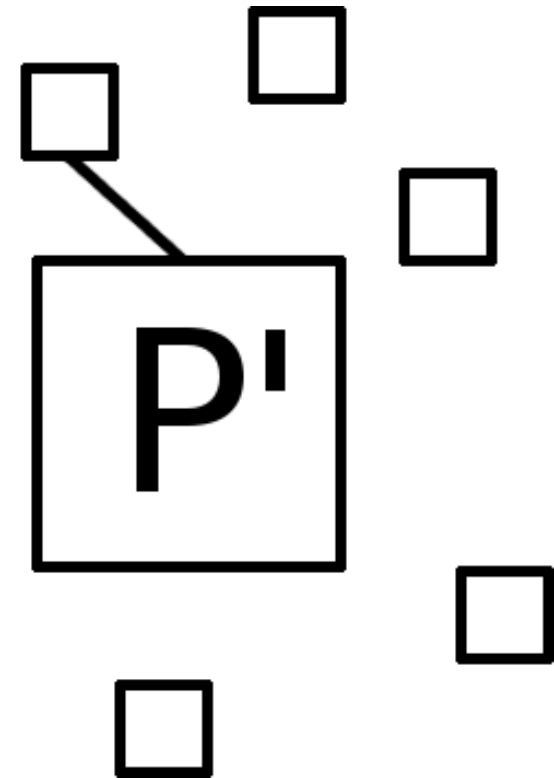




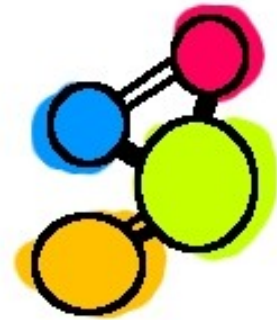
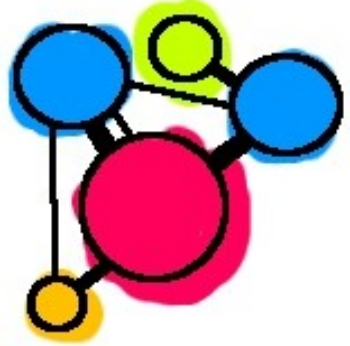
# Example



$$\begin{aligned}
 & \nu f. (f[P] \mid !f(X).f[X]) \\
 \xrightarrow{\alpha} & \nu f. (f[P'] \mid !f(X).f[X]) \\
 \equiv & \nu f. (f[P'] \mid !f(X).f[X] \\
 & \quad \mid f(Y).f[Y])
 \end{aligned}$$







# Example

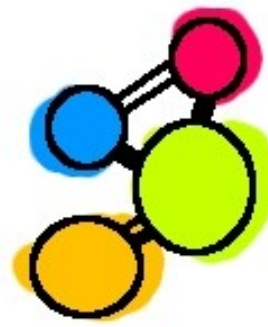
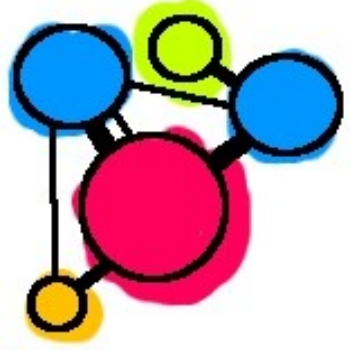
$$\begin{aligned}
 & \nu f.(f[P] \mid !f(X).f[X]) \\
 \xrightarrow{\alpha} & \nu f.(f[P'] \mid !f(X).f[X]) \\
 \equiv & \nu f.(f[P'] \mid !f(X).f[X] \\
 & \quad \mid f(Y).f[Y]) \\
 \xrightarrow{\tau} & \nu f.(0 \mid !f(X).f[X] \mid f[P']) \\
 \xrightarrow{\alpha} & \nu f.(0 \mid !f(X).f[X] \mid f[\textcolor{red}{P''}])
 \end{aligned}$$

$P''$   $\square$

$\square$

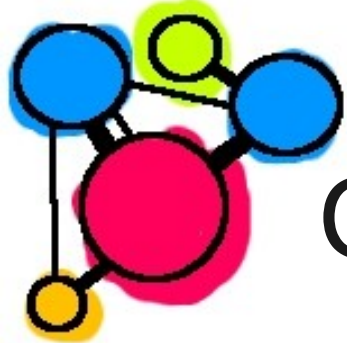
$\square$

$\square$

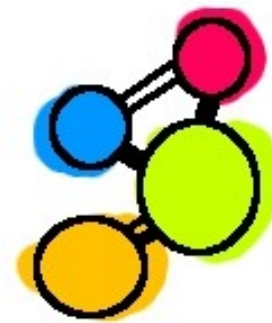


# Outline

- Modeling higher-order and distribution
- Correctness as equivalence
- Environmental bisimulations
- Example
- Conclusion



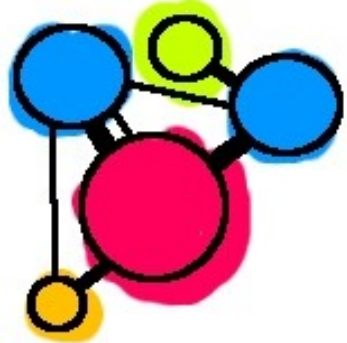
# Correctness of Our Example



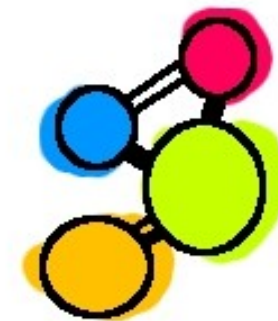
The system  $\nu f.(f[P] \mid !f(X).f[X])$  looks equivalent to the ideal system  $P$  (if  $f \notin \text{fn}(P)$ )

- The  $f$  cannot be observed from outside
  - nor react with  $P$
- Both  $P$ 's have the same transitions

Formally, what equivalence? Does it hold?

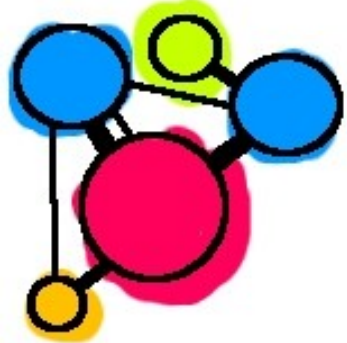


# Reduction-Closed Barbed Equivalence (RCBE)



Largest relation such that  $P \approx Q$  implies

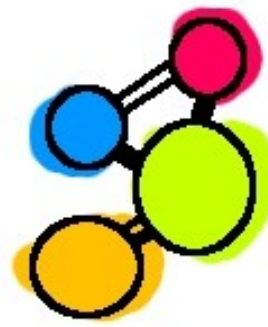
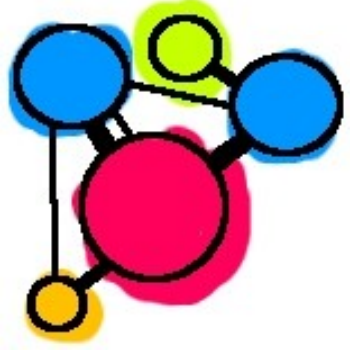
1. if  $P \rightarrow P'$  then  $Q \Rightarrow Q'$  and  $P' \approx Q'$
2. if  $P \xrightarrow{\mu}$  then  $Q \xRightarrow{\mu}$  (for  $\mu = a, \bar{a}$ )
3. the converse of the two above on  $Q$ , and
4. for all  $R$ ,  $P \mid R \approx Q \mid R$



# Reduction-Closed Barbed Equivalence (RCBE)

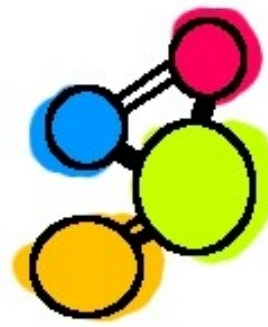
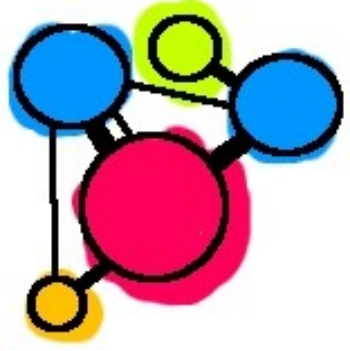
Largest relation considering all R's

1. if  $P \rightarrow P'$  then  $Q \rightarrow Q'$
2. if  $P \xrightarrow{\mu} Q$  then  $Q' \xrightarrow{\mu} Q''$  for  $\mu = a, \bar{a}$
3. the converse of the two above on  $Q$ , and
4. for all  $R$ ,  $P \mid R \approx Q \mid R$



# Outline

- Modeling higher-order and distribution
- Correctness as equivalence
- **Environmental bisimulations**
- Example
- Conclusion

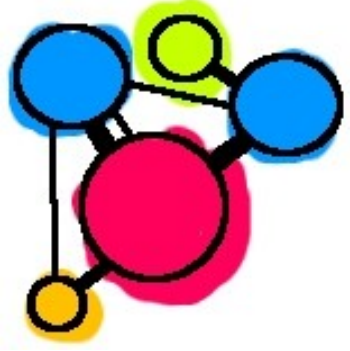


# Alternative to RCBE

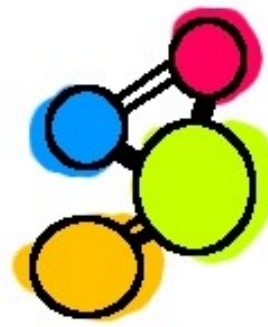
We want another equivalence

- Practical to use
- Implying reduction-closed barbed equivalence

We consider **environmental bisimulations** [Sumii-Pierce '04, Sumii-Pierce '05, Sangiorgi-Kobayashi-Sumii '07,...]

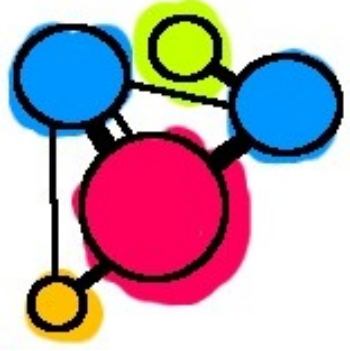


# Environmental Relation

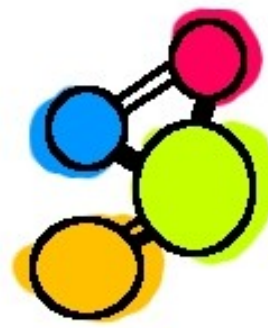


- A set of  $(E, P, Q)$  where
  - $P$  and  $Q$  are processes
  - $E$  the environment, is a binary relation on processes

environment  $\leftrightarrow$  "knowledge"

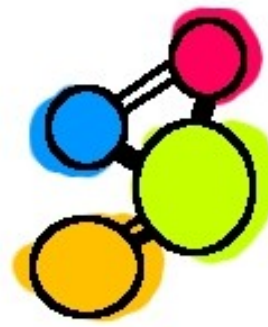
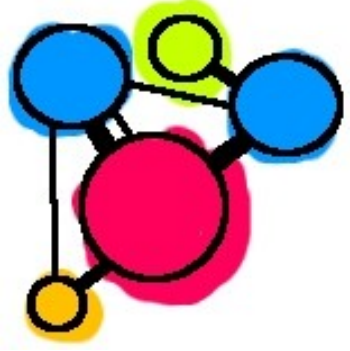


# Environmental Bisimulation (EB)



An environmental relation such that if  $P$  and  $Q$  are related:

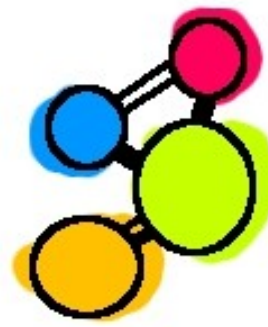
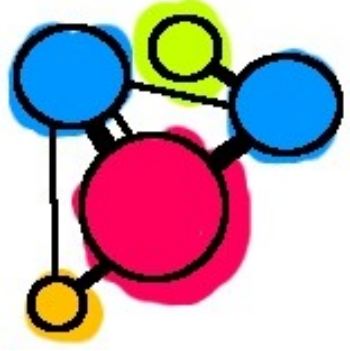
- Whatever  $P$  can do,  $Q$  must be able to do
  - Weakly and conversely
- If  $P$  (and thus  $Q$ ) outputs, **the environment learns the outputs**
- If  $P$  inputs,  $Q$  must be able to input **any input composed from the environment** (i.e. attacker's knowledge)



# EB and Output

Whenever  $(\mathcal{E}, P, Q) \in \mathcal{X}$  and  $P \xrightarrow{\bar{a}\langle M \rangle} P'$ ,

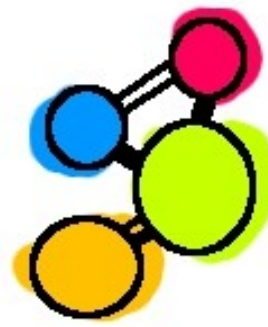
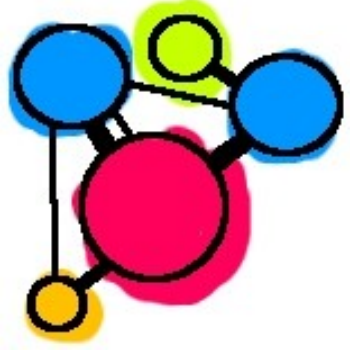
- $Q \xRightarrow{\bar{a}\langle N \rangle} Q'$ , and
- $(\mathcal{E} \cup \{(M, N)\}, P', Q') \in \mathcal{X}$



# EB and Input

Whenever  $(\mathcal{E}, P, Q) \in \mathcal{X}$  and  $P \xrightarrow{a(\textcolor{red}{C}[\widetilde{M}])} P'$ ,

- for all  $(\widetilde{M}, \widetilde{N}) \in \mathcal{E}$ ,  $Q \xRightarrow{a(\textcolor{red}{C}[\widetilde{N}])} Q'$ , and
- $(\mathcal{E}, P', Q') \in \mathcal{X}$



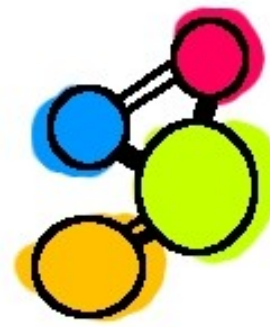
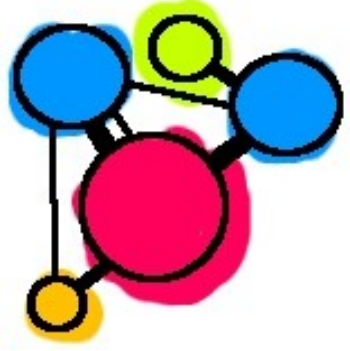
# EB and Spawn

The attacker can

- **spawn processes** next to those tested
- **use previous observations** (the environment)

Whenever  $(\mathcal{E}, P, Q) \in \mathcal{X}$ , for all  $(\widetilde{M}, \widetilde{N}) \in \mathcal{E}$ ,  
 $(\mathcal{E}, P|C[\widetilde{M}], Q|C[\widetilde{N}]) \in \mathcal{X}$

Even stronger than RCBE!! :-)

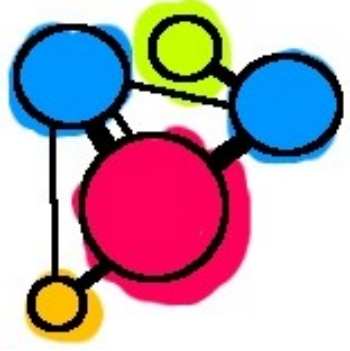


# EB and Spawn

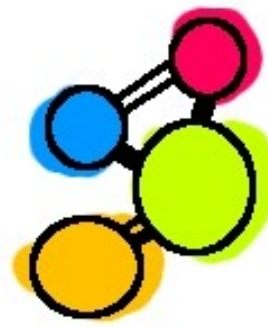
Previous research [Sangiorgi et al. '07, Sato et al. '09] used

$$(\mathcal{E}, P|M, Q|N) \in \mathcal{X} \text{ for all } (M, N) \in \mathcal{E}$$

However, it leads to **unsound** bisimulations in our settings

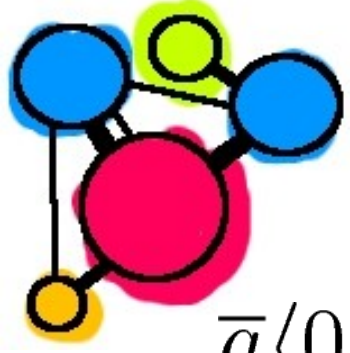


# Unsoundness of the Previous Condition



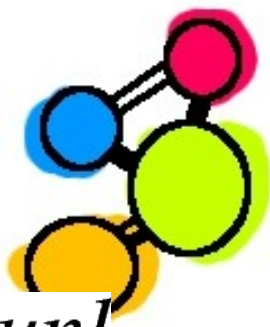
The attacker can **spawn processes**, but also **passivate them en route** *when spawned under a location*

Problematic during the evaluation of a sequential process



# On Unsoundness

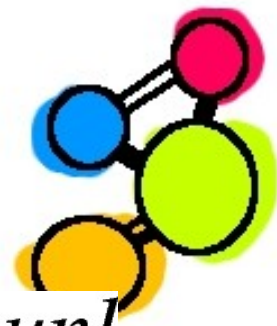
$\bar{a}\langle 0 \rangle.!\textit{lck.unl}$



$\bar{a}\langle \textit{lck.unl} \rangle.!\textit{lck.unl}$



# On Unsoundness



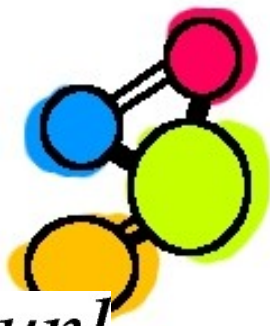
$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*



# On Unsoundness



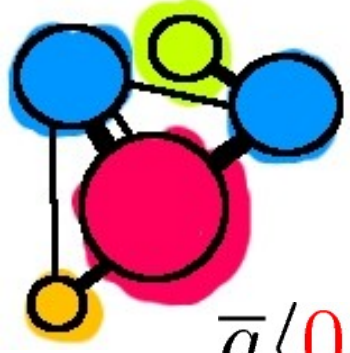
$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

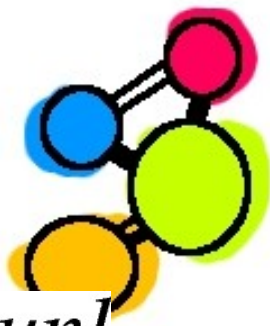
*output*

$!lck.unl$

$!lck.unl$



# On Unsoundness



$\bar{a}\langle 0 \rangle. !lck.unl$

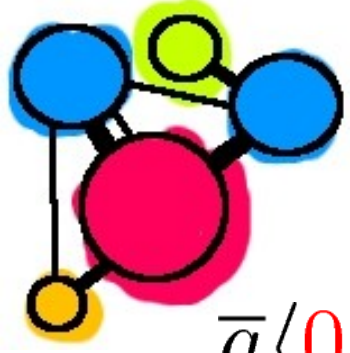
$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

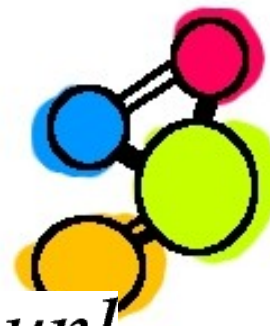
*!lck.unl*

*!lck.unl*

*spawn*



# On Unsoundness



$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

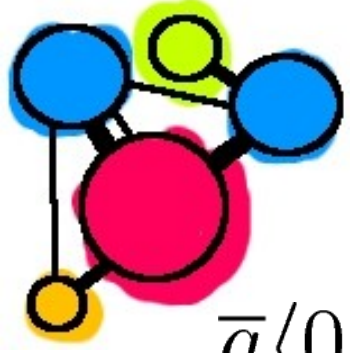
$!lck.unl$

$!lck.unl$

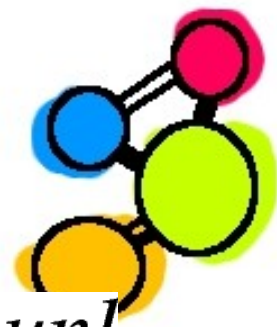
*spawn*

$!lck.unl \mid a[0]$

$!lck.unl \mid a[lck.unl]$



# On Unsoundness



$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

$!lck.unl$

$!lck.unl$

*spawn*

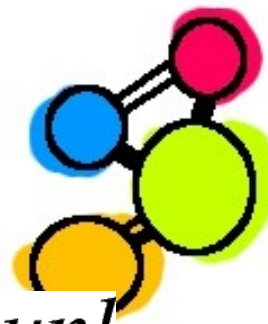
$!lck.unl \mid a[0]$

$!lck.unl \mid a[\textcolor{red}{lck.unl}]$

*input*



# On Unsoundness



$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

$!lck.unl$

$!lck.unl$

*spawn*

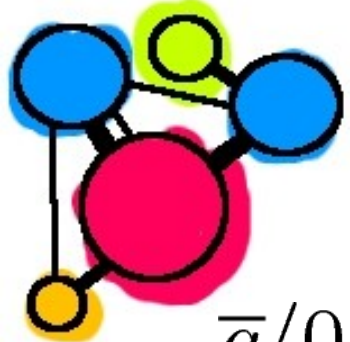
$!lck.unl \mid a[0]$

$!lck.unl \mid a[lck.unl]$

*input*

$!lck.unl \mid unl \mid a[0]$

$!lck.unl \mid a[unl]$



# On Unsoundness



$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

$!lck.unl$

$!lck.unl$

*spawn*

$!lck.unl \mid a[0]$

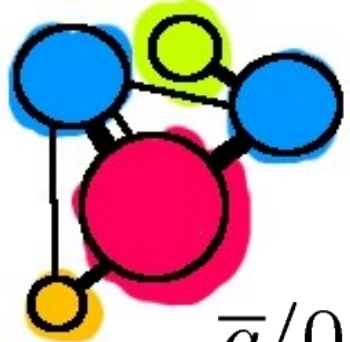
$!lck.unl \mid a[lck.unl]$

*input*

$!lck.unl \mid unl \mid a[0]$

$!lck.unl \mid a[unl]$

*passiv*



# On Unsoundness



$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

$!lck.unl$

$!lck.unl$

*spawn*

$!lck.unl \mid a[0]$

$!lck.unl \mid a[lck.unl]$

*input*

$!lck.unl \mid unl \mid a[0]$

$!lck.unl \mid a[unl]$

*passiv*

$!lck.unl \mid unl \mid 0$

$!lck.unl \mid 0$



# On Unsoundness

$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

$!lck.unl$

$!lck.unl$

*spawn*

$!lck.unl \mid a[0]$

$!lck.unl \mid a[lck.unl]$

*input*

$!lck.unl \mid unl \mid a[0]$

$!lck.unl \mid a[unl]$

*passiv*

$!lck.unl \mid \textcolor{red}{unl} \mid 0$

$!lck.unl \mid 0$

*input*



# On Unsoundness

$\bar{a}\langle 0 \rangle. !lck.unl$

$\bar{a}\langle lck.unl \rangle. !lck.unl$

*output*

$!lck.unl$

$!lck.unl$

*spawn*

$!lck.unl \mid a[0]$

$!lck.unl \mid a[lck.unl]$

*input*

$!lck.unl \mid unl \mid a[0]$

$!lck.unl \mid a[unl]$

*passiv*

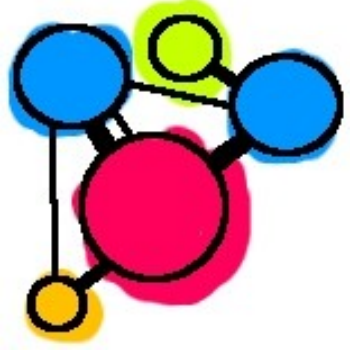
$!lck.unl \mid \textcolor{red}{unl} \mid 0$

$!lck.unl \mid 0$

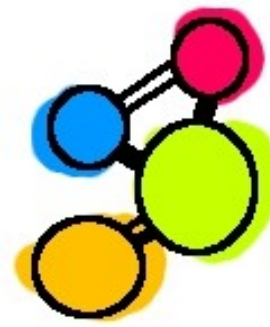
*input*

$!lck.unl \mid \textcolor{red}{0} \mid 0$

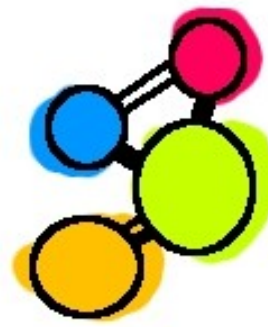
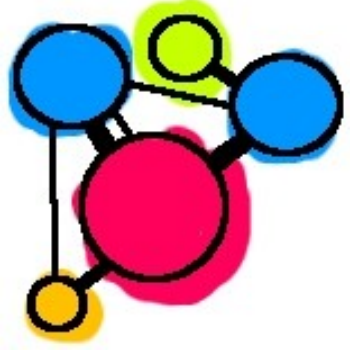
$\textcolor{red}{???}$



# EB and Spawn: Our Solution



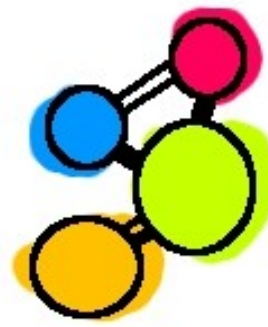
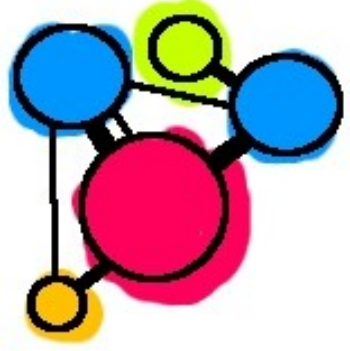
Whenever  $(\mathcal{E}, P, Q) \in \mathcal{X}$ , for all  $(M, N) \in \mathcal{E}$ ,  
 $(\mathcal{E}, P \mid a[M], Q \mid a[N]) \in \mathcal{X}$



# Summary

Environment bisimulation is an environmental relation preserved by

- Reductions
- Inputs of arguments composed from the environment
- Outputs (extending the environment)
- Spawning of related processes under a location

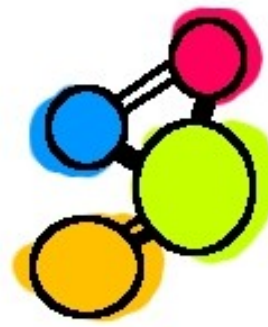
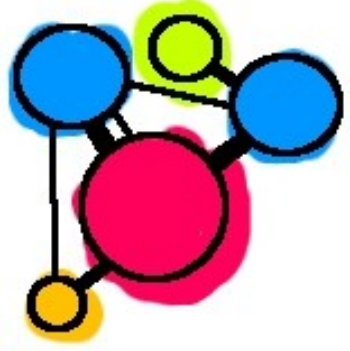


# EB: Improvements

Actual proofs become simpler when we use environmental bisimulations up-to

- structural congruence
- context
- environment, and
- restriction

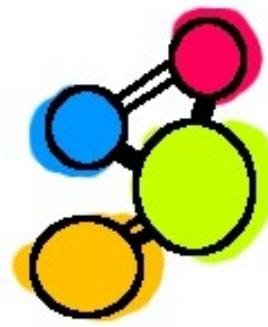
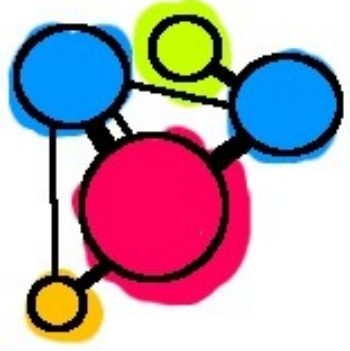
See [Piérard & Sumii, FoSSaCS '11] for details



# Soundness

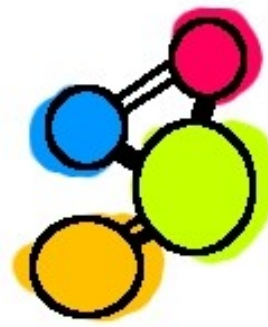
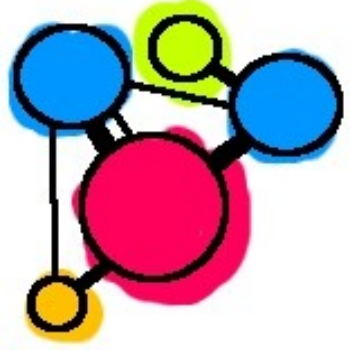
If  $(\emptyset, P, Q) \in \mathcal{X}$  then  $P \approx Q$

with a "simplicity" restriction on  $\mathcal{X}$  for technical reasons (fixing this is ongoing work)



# "Simplicity"

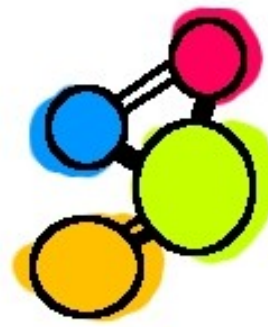
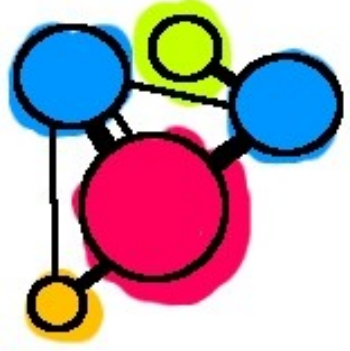
- Simple process: no subprocess has the form  $vx.P$  nor  $a(X).P$  with  $X \in \text{fv}(P)$
- Simple environment: made of simple processes
- Simple environmental relation: has only simple environments



# Soundness (bis)

Soundness holds for simple EBs

Thanks to up-to techniques, we can actually  
handle some non-simple (and non-trivial)  
processes

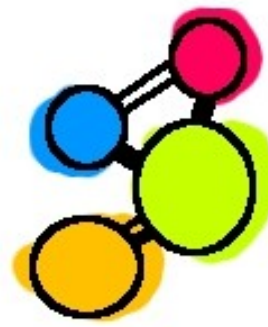
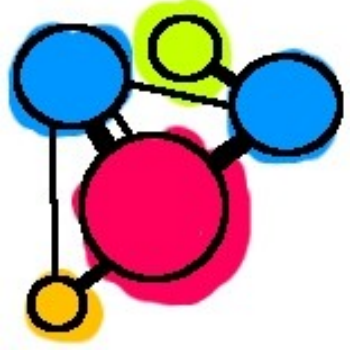


# Bonus Result

Reduction-closed barbed *congruence*  $\approx_c$   
(standard definition)

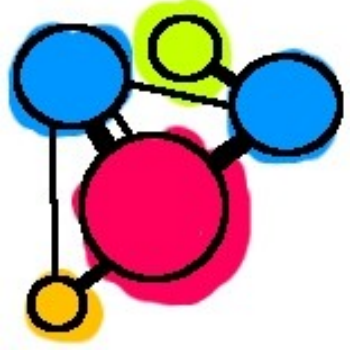
If  $(\emptyset, \bar{a}\langle P \rangle, \bar{a}\langle Q \rangle) \in \mathcal{X}$  then  $P \approx_c Q$

Compare with context bisimulations, where testing  $\bar{a}\langle P \rangle$  and  $\bar{a}\langle Q \rangle$  would imply testing  $P$  and  $Q$  in any context!

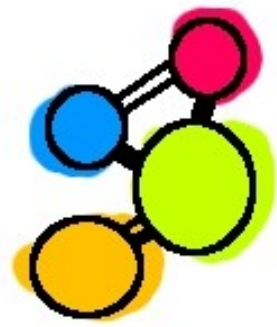


# Outline

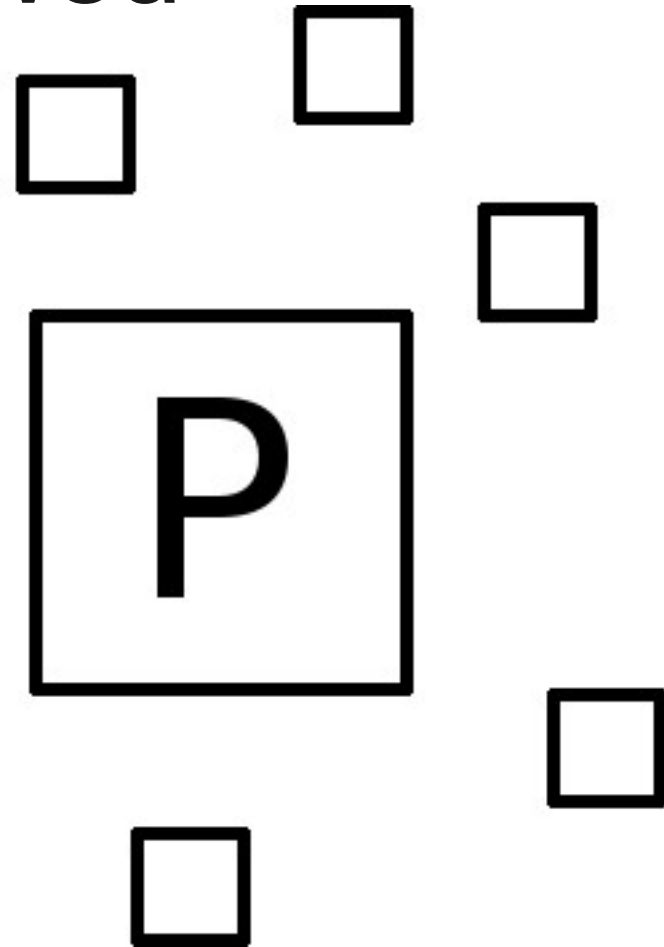
- Modeling higher-order and distribution
- Correctness as equivalence
- Environmental bisimulations
- **Example**
- Conclusion

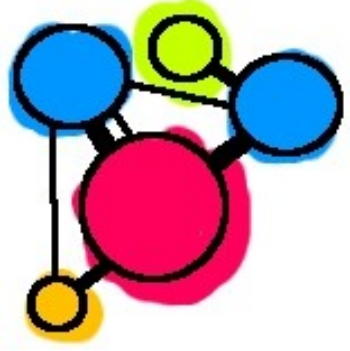


# Equivalence of Example, Reviewed

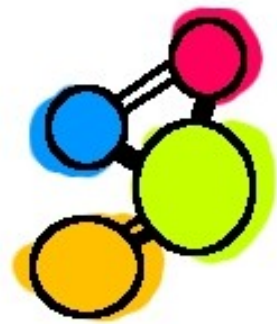


P

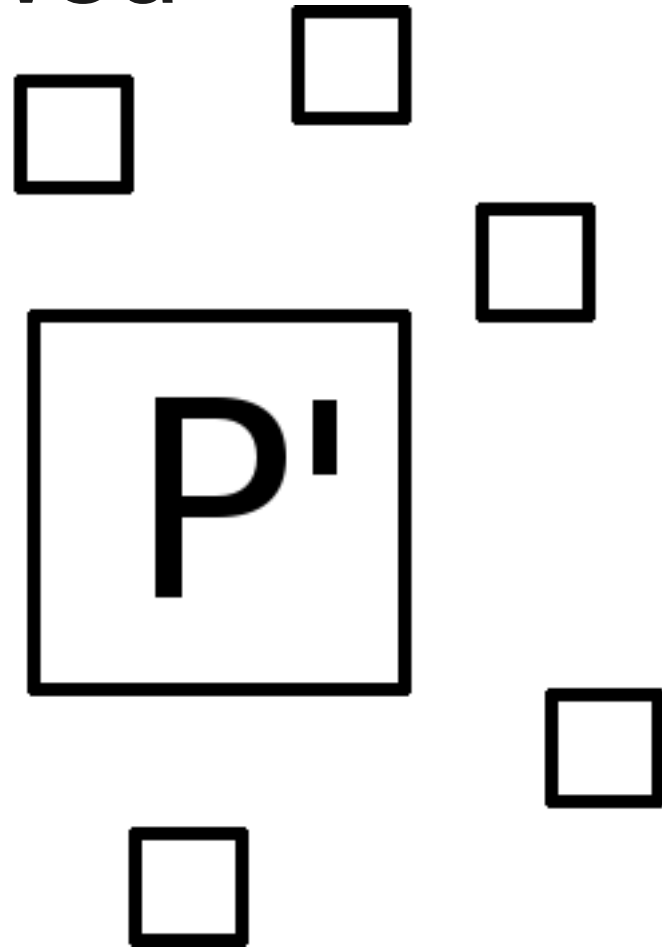


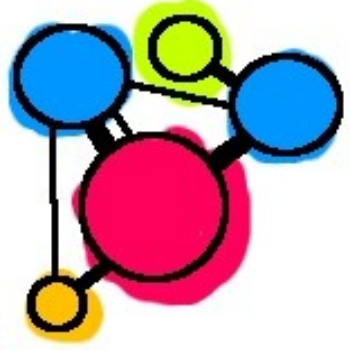


# Equivalence of Example, Reviewed

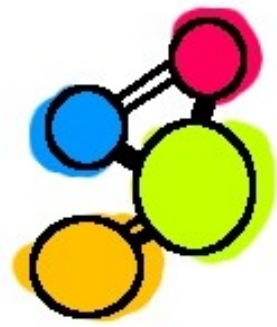


$P^i$

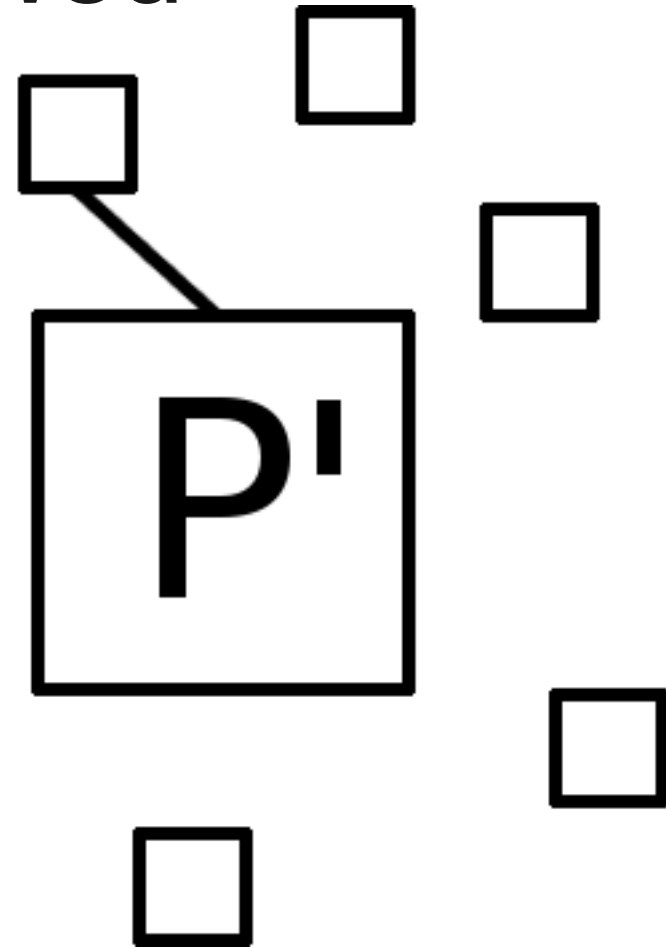


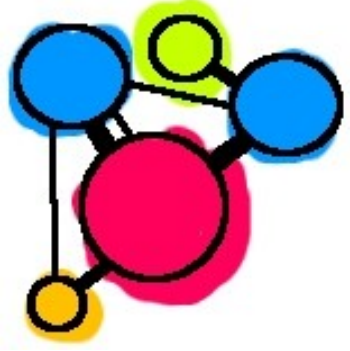


# Equivalence of Example, Reviewed

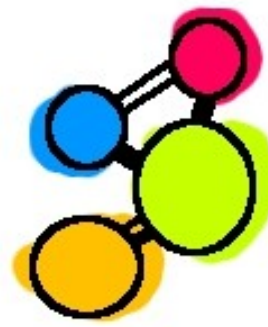


$P^i$

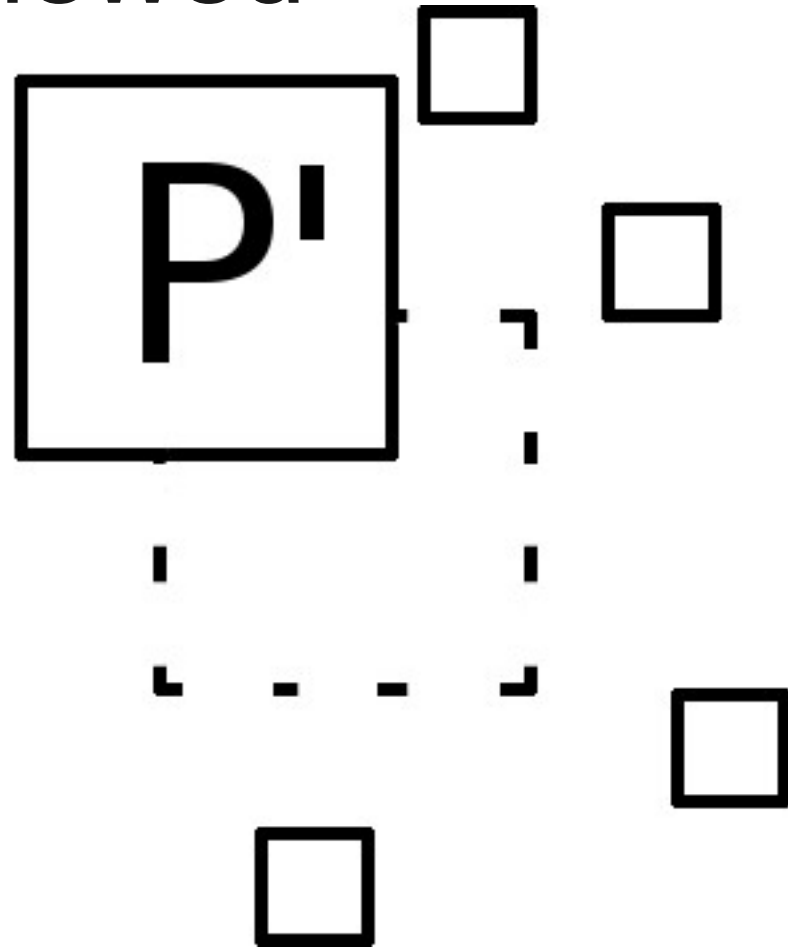


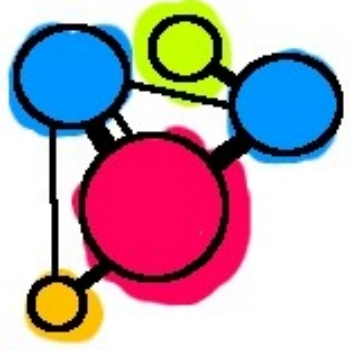


# Equivalence of Example, Reviewed

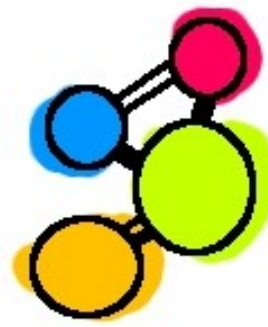


$P^i$



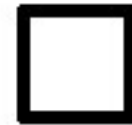
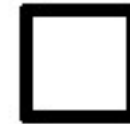


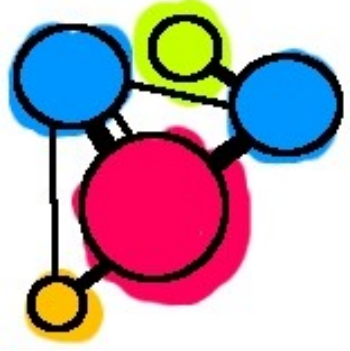
# Equivalence of Example, Reviewed



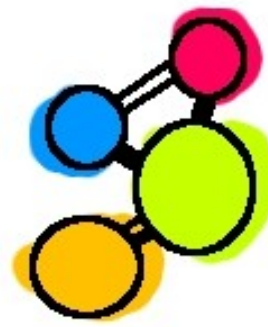
$P''$

$P''$





# Equivalence of Example with EB



$$P \approx \nu f. (f[P] \mid !f(X).f[X]) ?$$

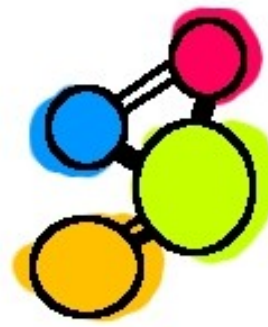
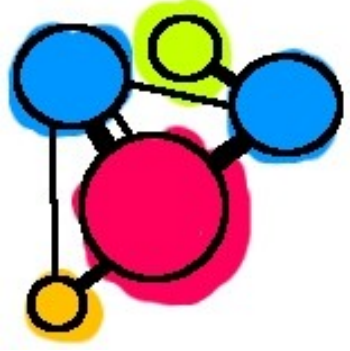
Proof: find an EB

$$X \ni (\emptyset, P, \nu f. (f[P] \mid !f(X).f[X]))$$

Take

$$X = \{(\emptyset, P, \nu f. (f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

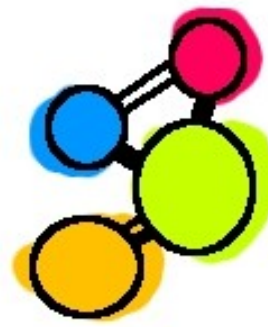
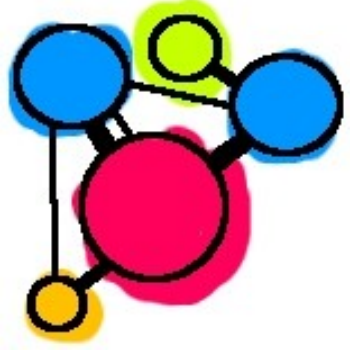
and check clauses of EB



# Transitions (Input)

$$X = \{(\emptyset, P, \nu f.(f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

$$\begin{array}{ccc} (\emptyset, & P, & \nu f.(f[P] \mid !f(X).f[X]) & \in \mathcal{X} \\ \downarrow \text{a(R)} & & \downarrow \text{a(R)} & \\ (\emptyset, & \textcolor{red}{P}', & \nu f.(f[\textcolor{red}{P}'] \mid !f(X).f[X]) & \in \mathcal{X} \end{array}$$

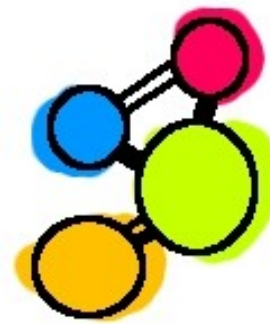
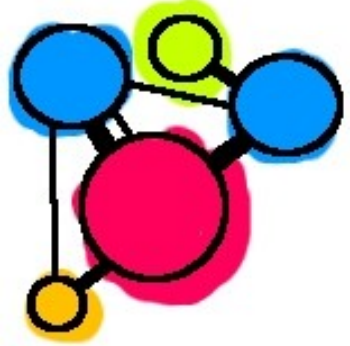


# Transitions (Output)

$$X = \{(\emptyset, P, \nu f.(f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

$$\begin{array}{ccc}
 (\emptyset, & P, & \nu f.(f[P] \mid !f(X).f[X]) \quad \in \mathcal{X} \\
 \downarrow \bar{a}\langle R \rangle & & \downarrow \bar{a}\langle R \rangle \\
 (\emptyset, & P', & \nu f.(f[P'] \mid !f(X).f[X]) \quad \in \mathcal{X}
 \end{array}$$

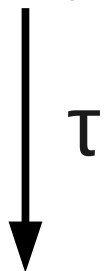
Identical outputs are cancelled up-to context



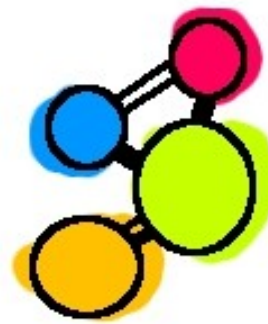
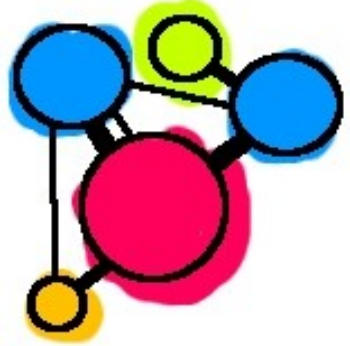
# Transitions (Reaction)

$$X = \{(\emptyset, P, \nu f.(f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

$$(\emptyset, P, \nu f.(f[P] \mid !f(X).f[X])) \in \mathcal{X}$$



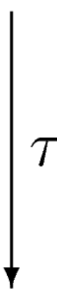
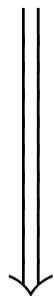
$$(\emptyset, \textcolor{red}{P'}, \nu f.(f[\textcolor{red}{P}'] \mid !f(X).f[X])) \in \mathcal{X}$$



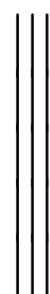
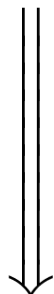
# Transitions (Reaction)

$$X = \{(\emptyset, P, \nu f.(f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

$$(\emptyset, P, \nu f.(\textcolor{red}{f}[P] \mid !\textcolor{red}{f}(X).f[X])) \in \mathcal{X}$$

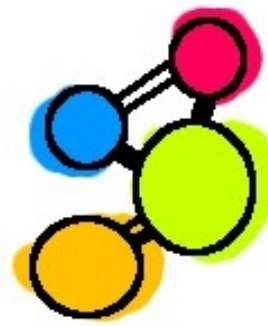
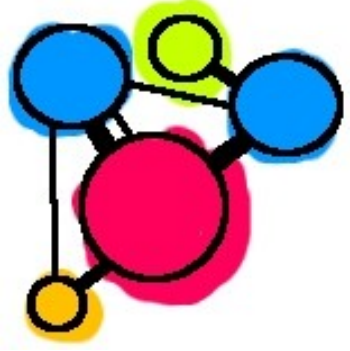


$$(\emptyset, P, \nu f.(\textcolor{red}{0} \mid !f(X).f[X] \mid \textcolor{red}{f}[P])) \in \mathcal{X}$$



$$(\emptyset, P, \nu f.(f[\textcolor{red}{P}] \mid !f(X).f[X])) \in \mathcal{X}$$

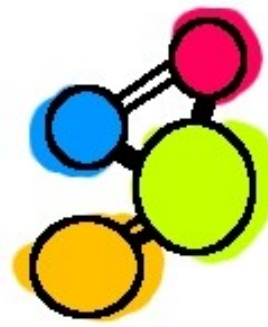
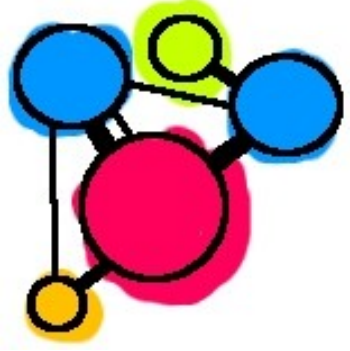
Use up-to structural congruence



# Spawn Clause

$$X = \{(\emptyset, P, \text{vf.}(f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

Vacuously satisfied as the environment is empty



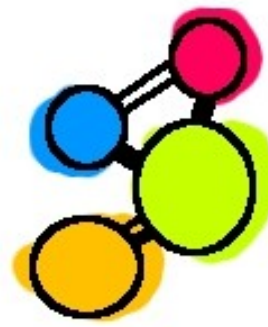
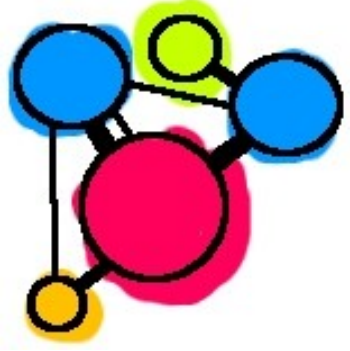
# Result

$$X = \{(\emptyset, P, \text{vf.}(f[P] \mid !f(X).f[X])) \mid f \notin \text{fn}(P)\}$$

$X$  is an EB (up-to context, etc)

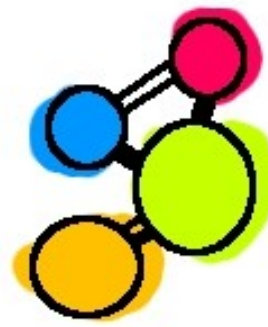
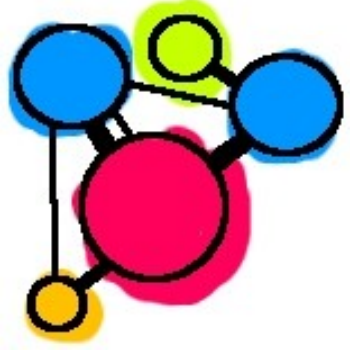
hence

$$P \approx \text{vf.}(f[P] \mid !f(X).f[X])$$



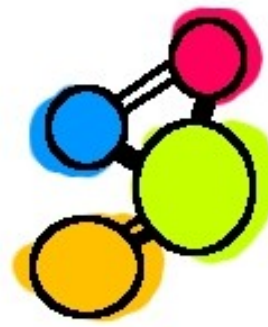
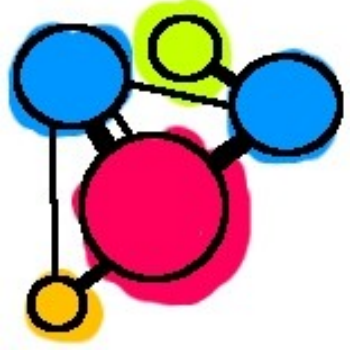
# Outline

- Modeling higher-order and distribution
- Proving equivalence
- Environmental bisimulations
- Example
- Conclusion



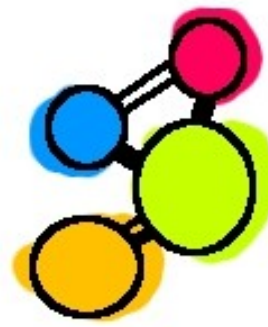
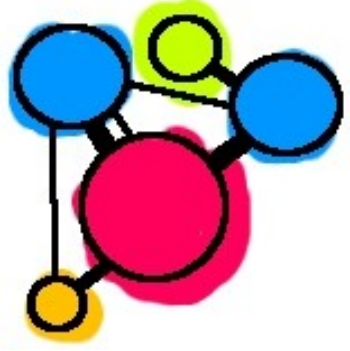
# Related Work

- The Kell calculus [Schmitt and Stefani '04]
  - Uses context bisimulations
- $\text{HO}\pi\text{P}$  [Lenglet et al. '09]
  - Uses context bisimulations
- Homer [Hildebrandt et al. '04]
  - Uses Howe's method

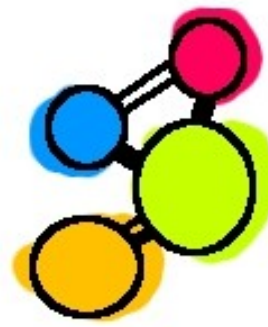
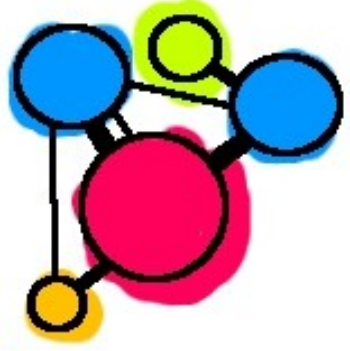


# Future Work

- Completeness
  - Improve the spawn clause (remove the simplicity restriction)
- More up-to techniques (eg. up-to bisimilarity)
- EBs for more expressive languages, for better modeling of realistic systems
  - Kell calculus?
  - Other?



Should you remember just one thing, let it be  
the following slide!

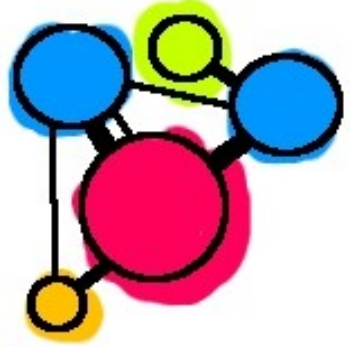


# Conclusion

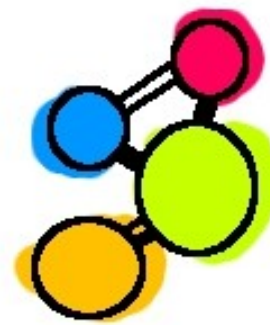
**Higher-order distributed  
computing is ubiquitous, but hard**

**Environmental bisimulations  
enable correctness proof  
(or disproofs)**

Thank you for your attention!



# The Spawn Clause: The Problems Strike Back



The spawn clause gives us for some

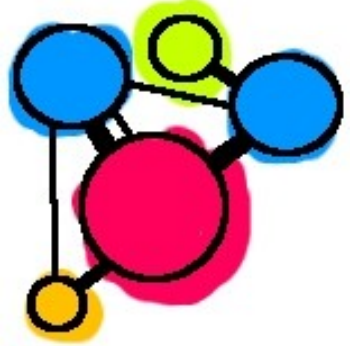
$(,P \mid a[A1] \mid b[A2], Q \mid a[B1] \mid b[B2]) \in X,$

$(,P \mid vx(a[A1'] \mid b[A2']), Q \mid vy(a[B1'] \mid b[B2'])) \in X$

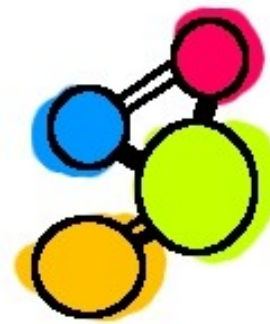
But it does not account for reactions of

$(,P \mid m[A1 \mid A2], Q \mid m[B1 \mid B2]) \in X,$  giving

$(,P \mid m[vx.(A1' \mid A2')], Q \mid m[vy.(B1' \mid B2')]) \in X$



# The Spawn Clause: The Problems Strike Back



$P \mid m[vx.(A1'|A2')] \times Q \mid m[vy.(B1'|B2')]$

Passivation of  $m[...]$  would keep the names  $x$ ,  $y$  bound in the environment

$P \mid vx(a[A1']|b[A2']) \times Q \mid vy(a[B1']|b[B2'])$

Passivations of  $a[...]$  and  $b[...]$  would extrude the names  $x$  and  $y$ ...

This poses technical problems in reductions in up-to context proofs, and doubt in our minds