# A Correct Abstract Machine for the Stochastic Pi-calculus

Andrew Phillips    Luca Cardelli

August 30th 2004

Microsoft Research
7 JJ Thomson Avenue
Cambridge, UK

# Introduction

- Ongoing Experiment:
  - Use process calculi to model biological systems
- Features of process calculi:
  - *Compositional* modelling, analysis and simulation of systems.
- Potential Benefits:
  - *Understand* complex systems by decomposing them into simpler subsystems.
  - *Analyse* properties of subsystems using established theory.
  - *Predict* behaviour of subsystems by running stochastic simulations.
  - Predict properties and behaviour of *composed* systems.
- Pi-calculus: one of the simplest and most well-studied calculi.

# Outline

- Graphical Pi-Calculus
- Gene Regulation by Positive Feedback [Priami et al., 2001]
- Abstract Machine for Stochastic Pi-Calculus
- Simulator for Stochastic Pi-Calculus

# Stochastic Pi-Calculus

- Syntax:

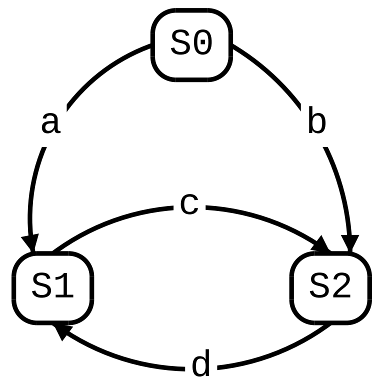
$$
\begin{array}{rcll}
P, Q ::= & \nu x\, P & \text{Restriction} \\
\mid & P \mid Q & \text{Parallel} \\
\mid & \Sigma & \text{Summation} \\
\mid & !\pi.P & \text{Replication}
\end{array}
\qquad
\begin{array}{rcll}
\Sigma ::= & \mathbf{0} & \text{Null} \\
\mid & \pi.P + \Sigma & \text{Action} \\
\pi ::= & x\langle n \rangle & \text{Output} \\
\mid & x(m) & \text{Input}
\end{array}
$$

- Semantics:

$$
\begin{array}{rcl}
(x\langle n \rangle.P + \Sigma) \mid (x(m).Q + \Sigma') & \xrightarrow{rate(x)} & P \mid Q_{\{n/m\}} \\
P \xrightarrow{r} P' \;\Rightarrow\; P \mid Q & \xrightarrow{r} & P' \mid Q \\
P \xrightarrow{r} P' \;\Rightarrow\; \nu x\, P & \xrightarrow{r} & \nu x\, P' \\
Q \equiv P \wedge P \xrightarrow{r} P' \wedge P' \equiv Q' \;\Rightarrow\; Q & \xrightarrow{r} & Q'
\end{array}
$$

# Graphical Pi-Calculus

➢ An intuitive representation for pi-calculus. Like FSMs...

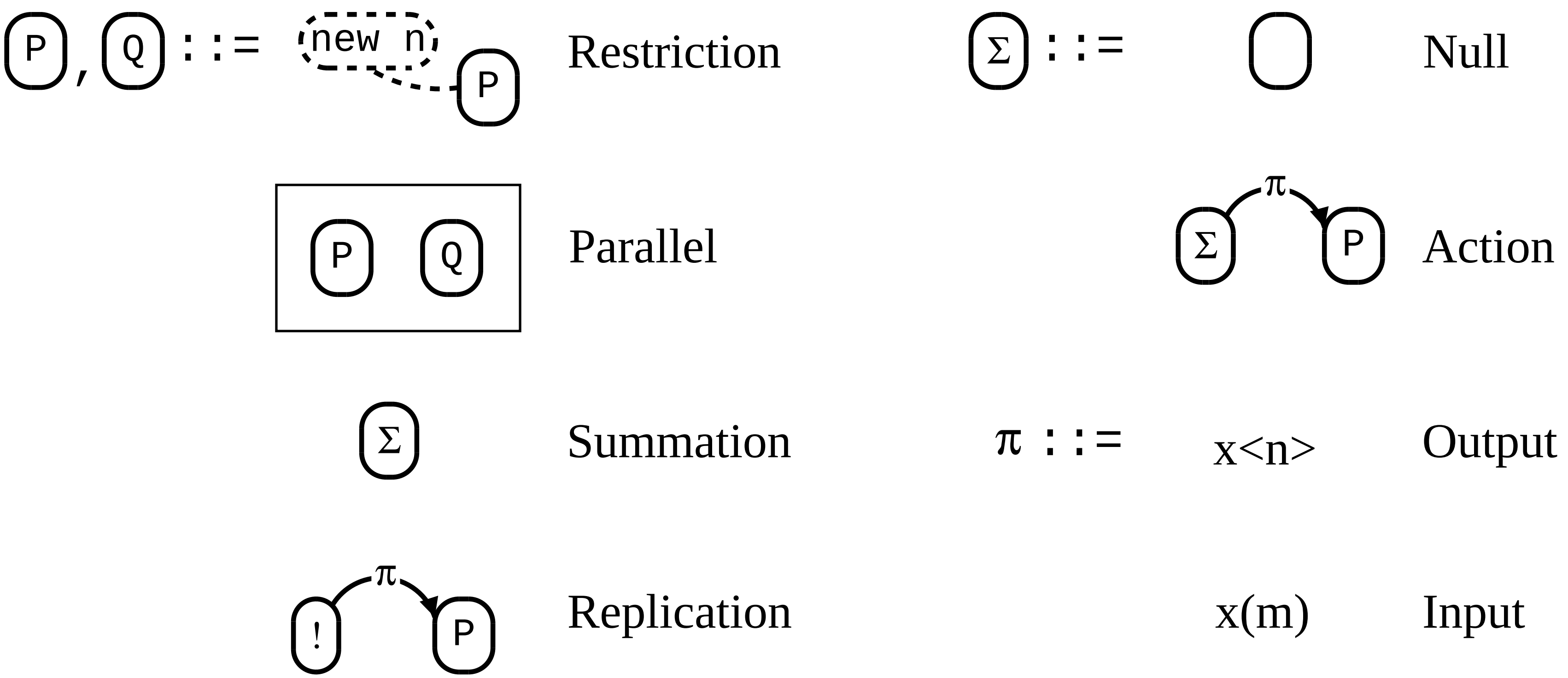



➢ But with all the features of pi: compositionality, restriction, communication, replication.

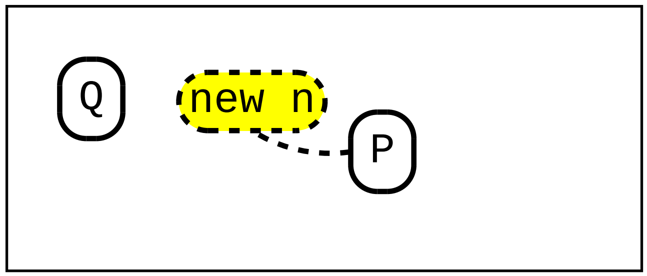
➢ Should be a 1-1 correspondence between graphics and text

➢ NO NEW THEORY

# Graphical Syntax

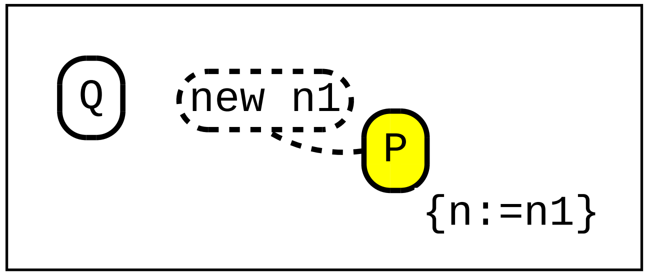


# Graphical Semantics: Restriction



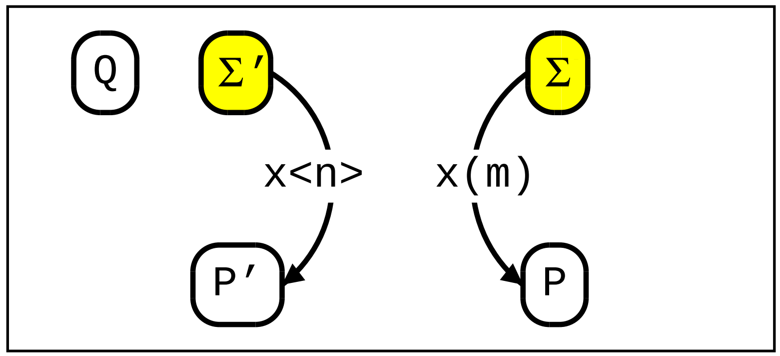

➢ Restriction creates a fresh name inside a given process.

# Graphical Semantics: Restriction



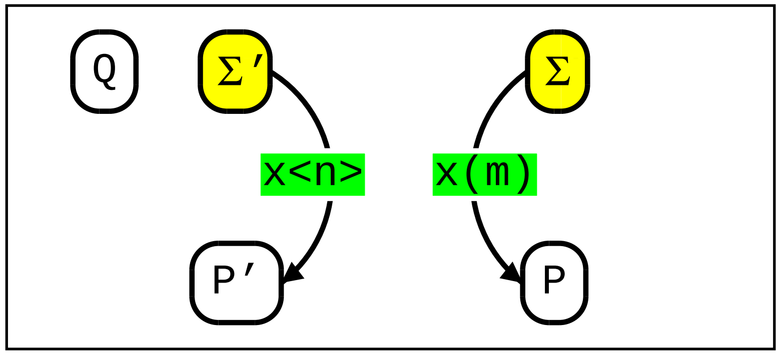

➢ The name $n$ is replaced with a fresh name $n1$ that is unknown to $Q$.

# Graphical Semantics: Communication



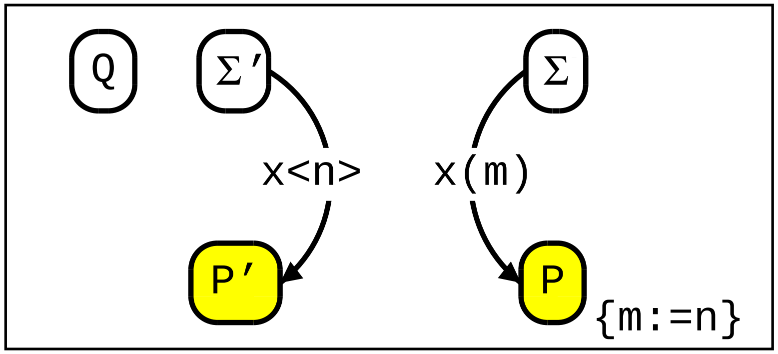

➢ Two parallel summations can interact on a common channel.

# Graphical Semantics: Communication



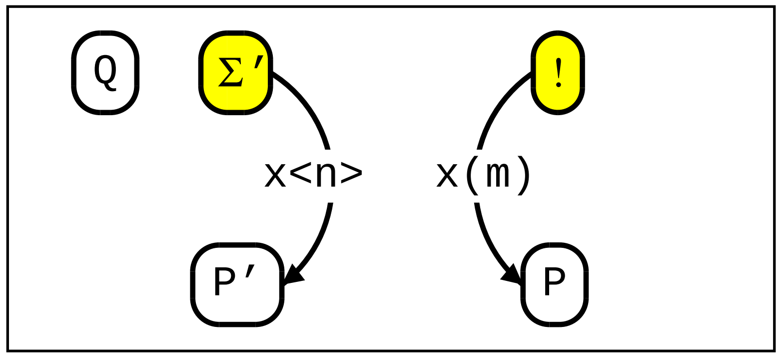

➢ An output $x\langle n\rangle$ can send a message $n$ on channel $x$ to an input $x(m)$.

# Graphical Semantics: Communication



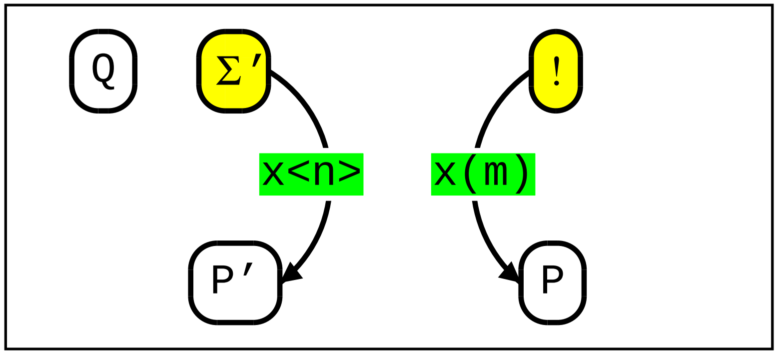

➢ Message $n$ is assigned to $m$ in process $P'$.

# Graphical Semantics: Replication



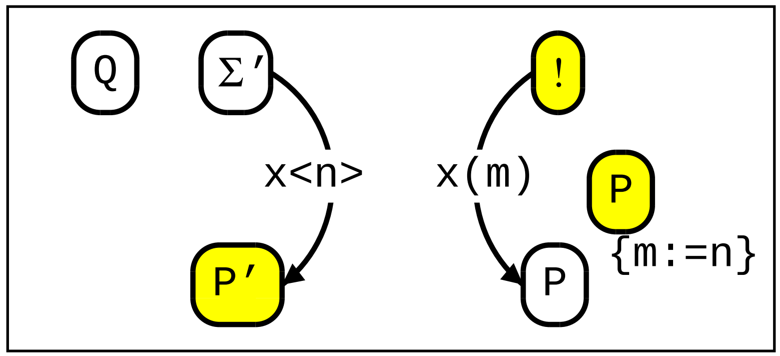

➢ A replicated input can spawn a clone of a process.

# Graphical Semantics: Replication




➢ An output $x\langle n\rangle$ can send a message $n$ to a replicated input $!x(m)$.

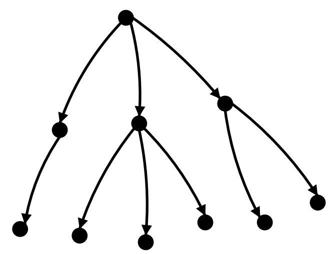
# Graphical Semantics: Replication



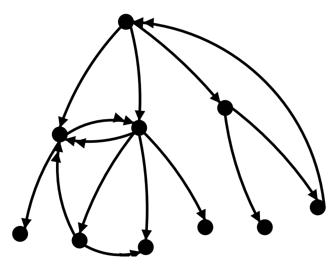

➢ A clone of $P$ is spawned and message $n$ is assigned to $m$ in the clone.

# Trees vs Graphs

➢ By definition, a graphical pi process is a *tree* of nodes:

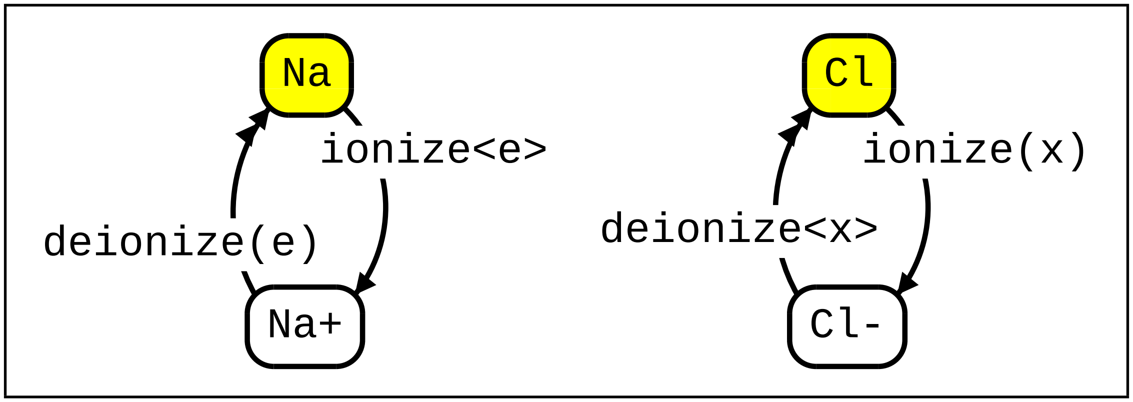


➢ *Links* between nodes in the tree can be encoded to represent recursive processes:



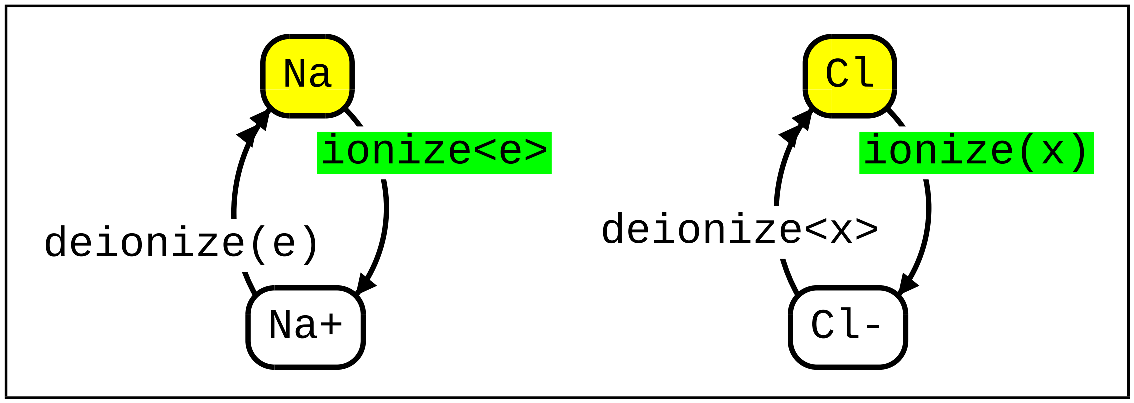
➢ The result is an arbitrary graph with two kinds of edges.

# Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$



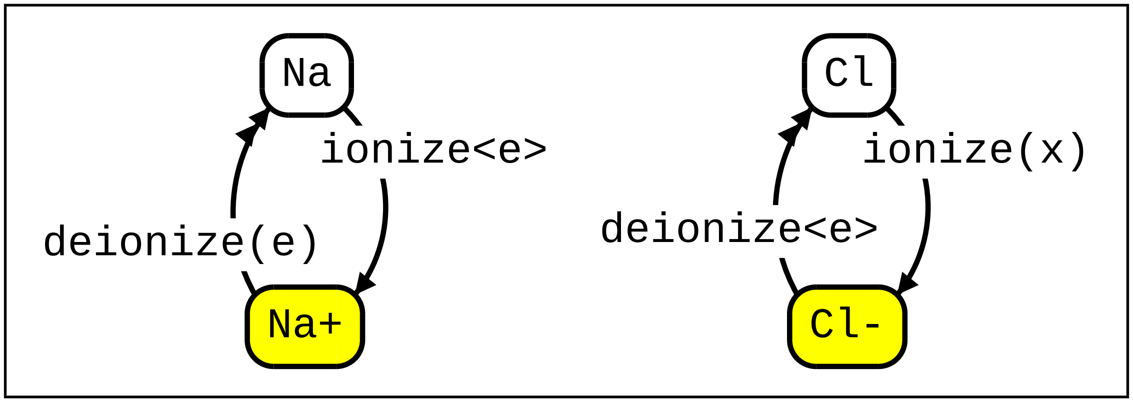

- $Na$ can ionize $Cl$ by sending its electron, with rate $100s^{-1}$
- $Cl^-$ can deionize $Na^+$ by sending its electron, with rate $10s^{-1}$
- State names are merely *annotations*

# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



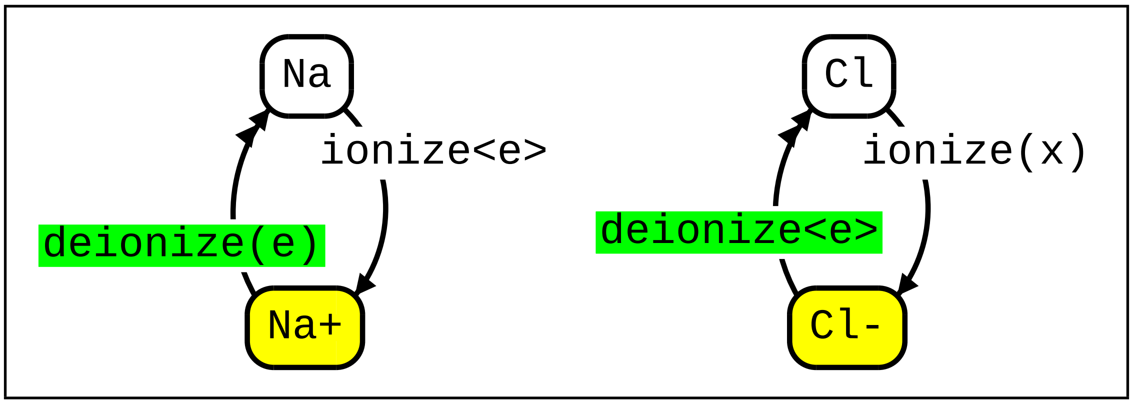

- ➢ $Na$ can ionize $Cl$ by sending its electron on the $ionize$ channel

# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



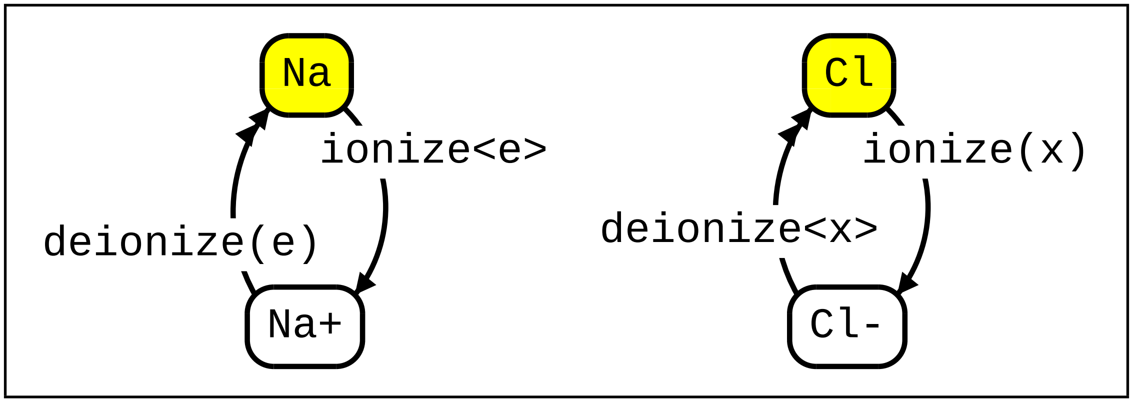

- $Na^{+}$ is positively charged and $Cl^{-}$ is negatively charged

# Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$



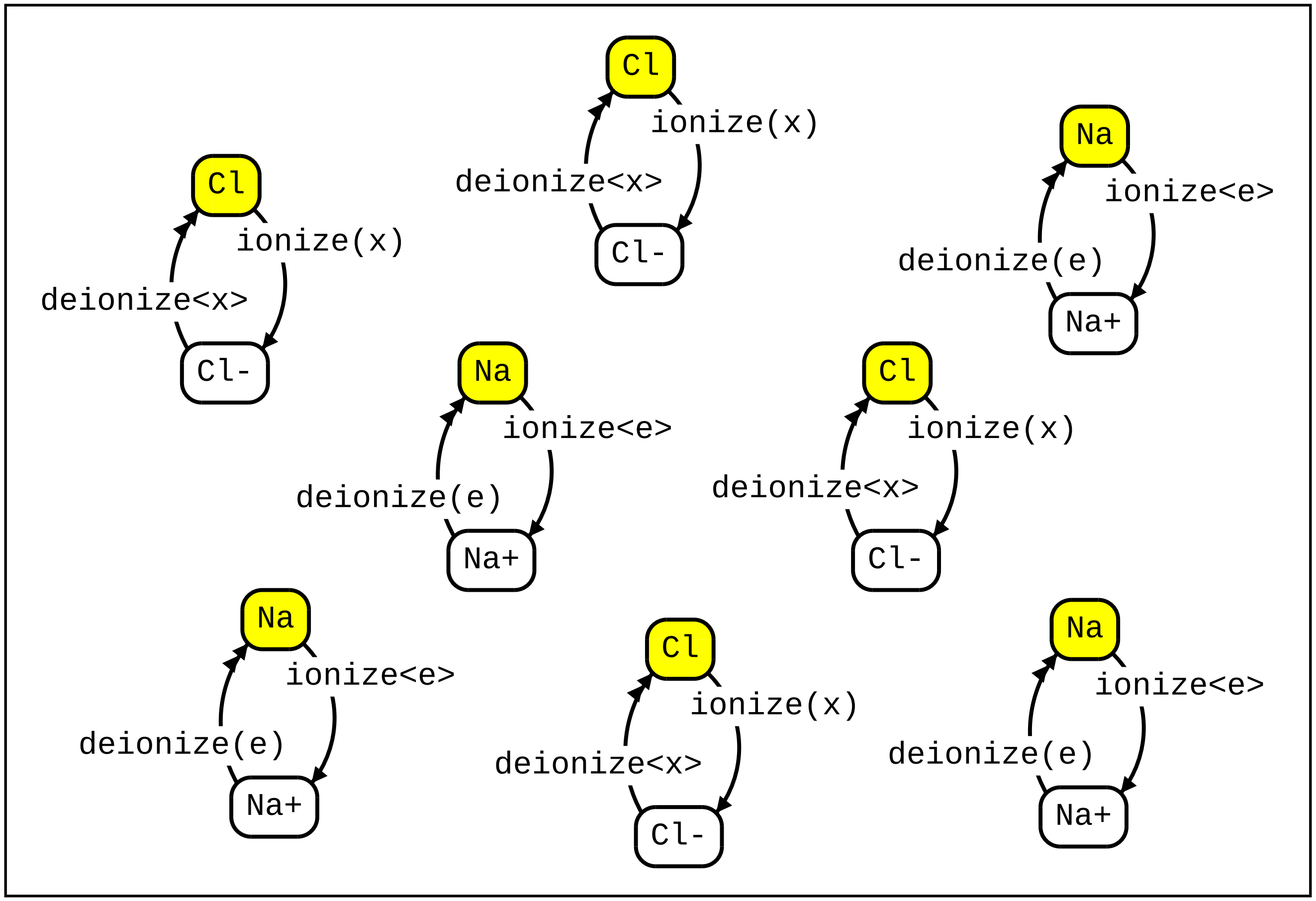

➢ $Cl^-$ can deionize $Na^+$ by sending its electron on the *deionize* channel

# Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$

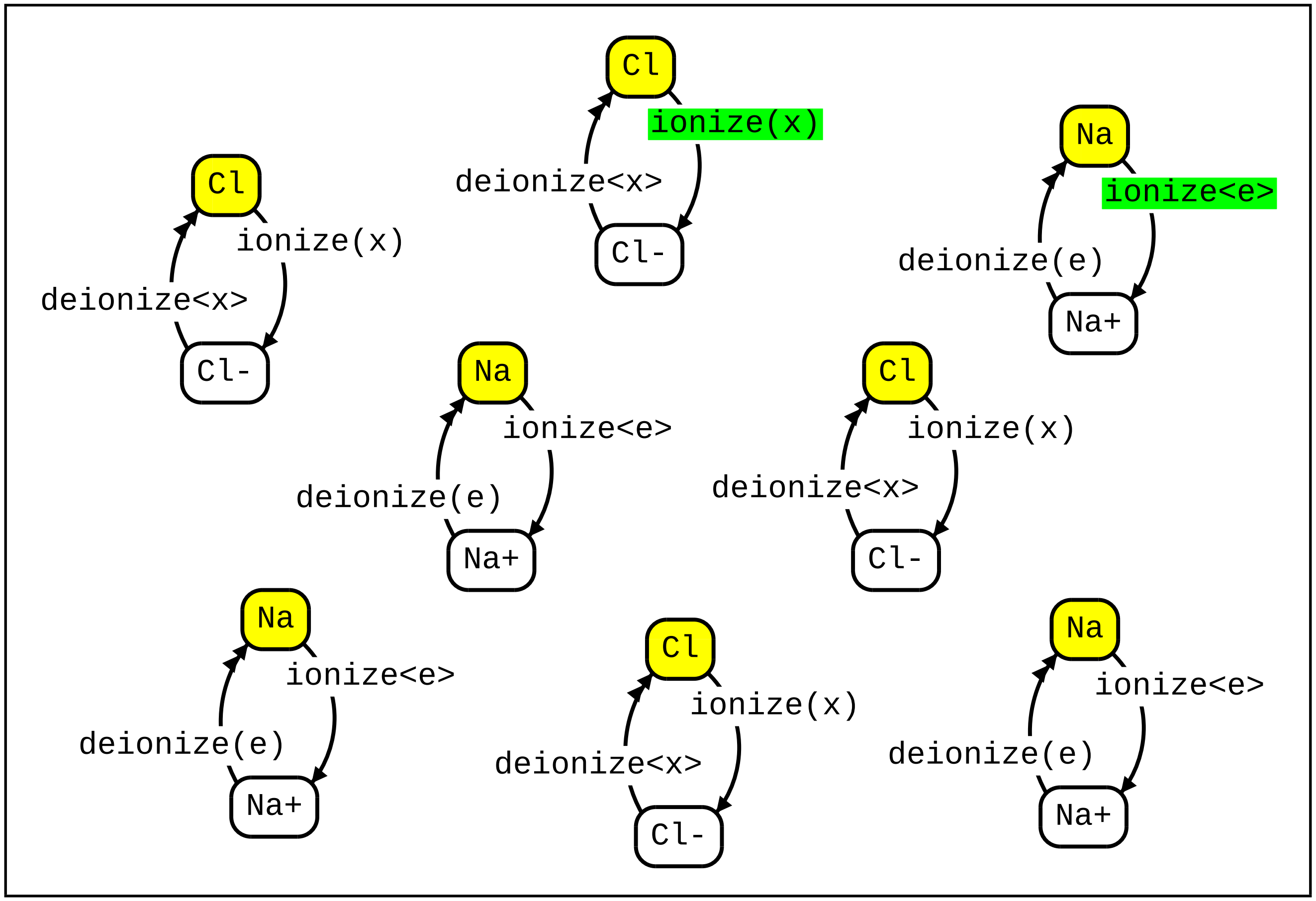



- $Na$ and $Cl$ are no longer charged

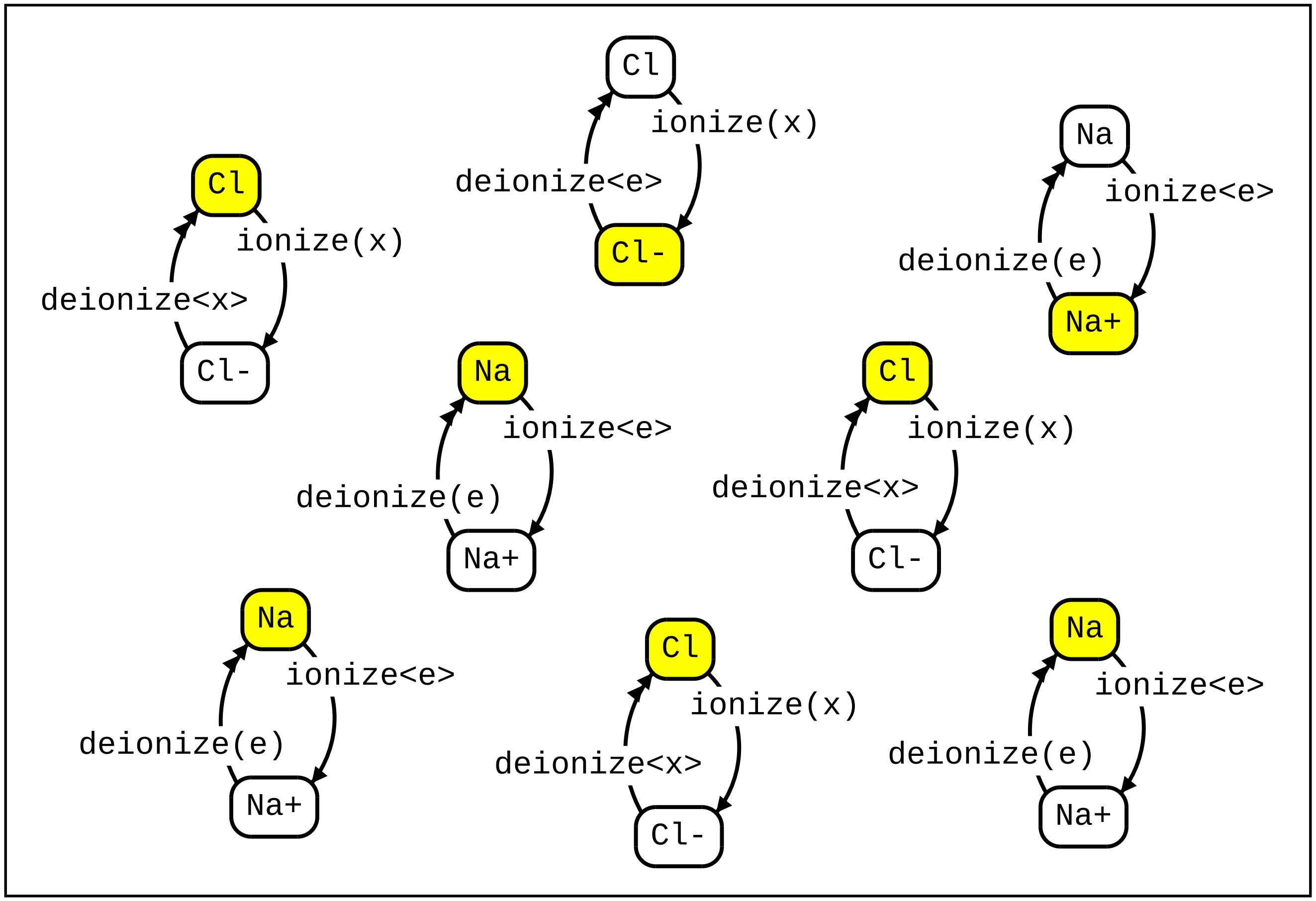
# Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$



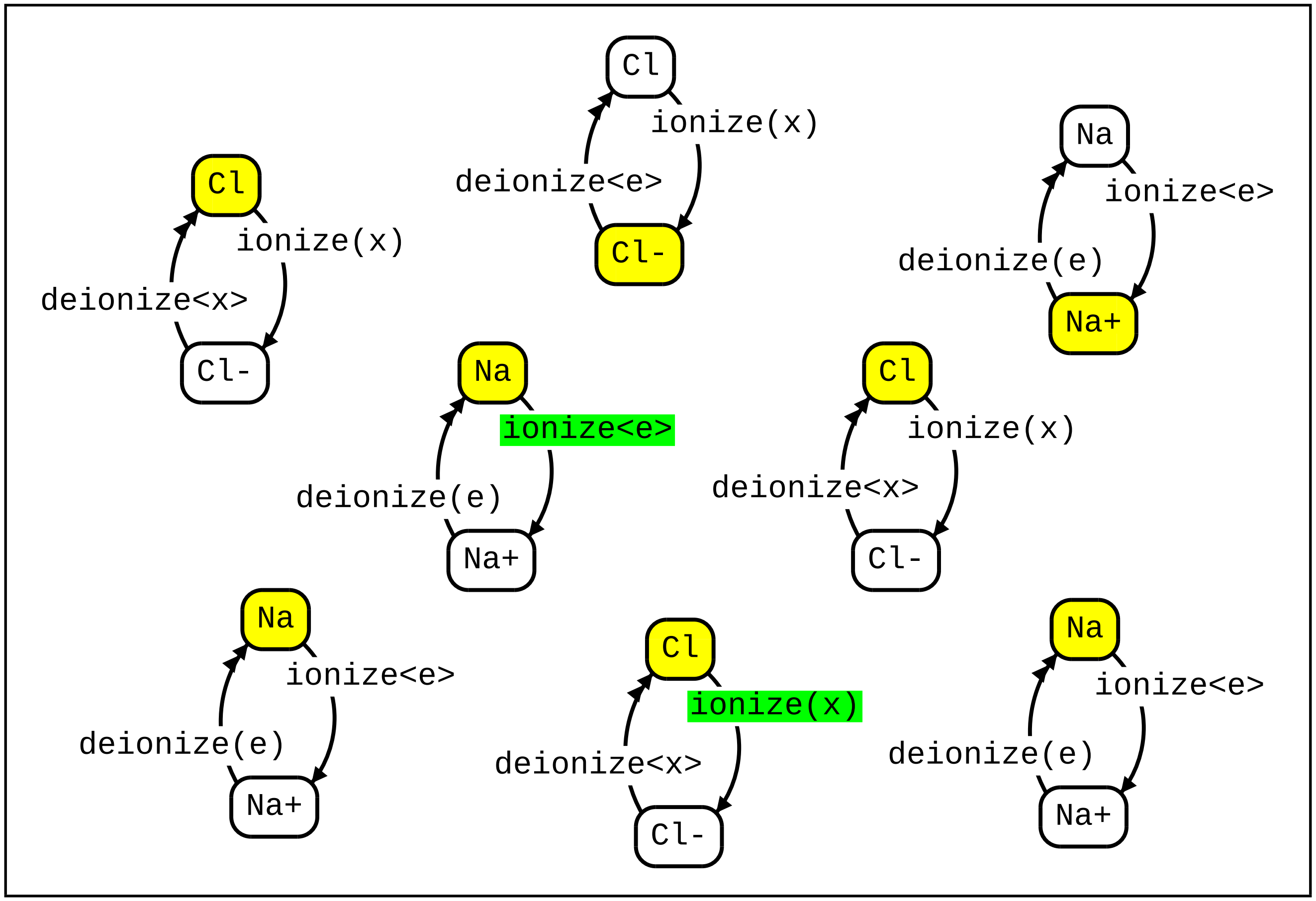
# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



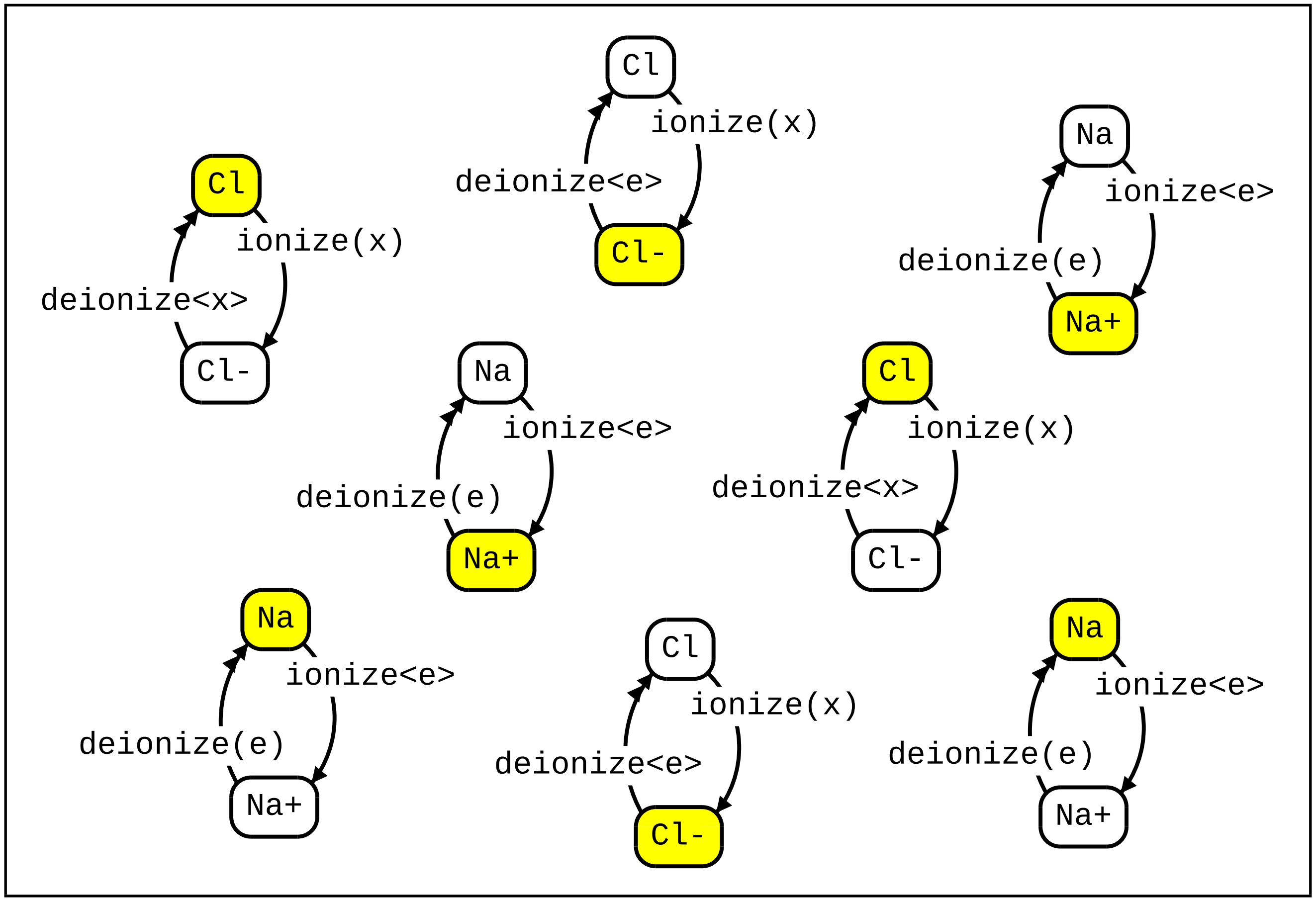
# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



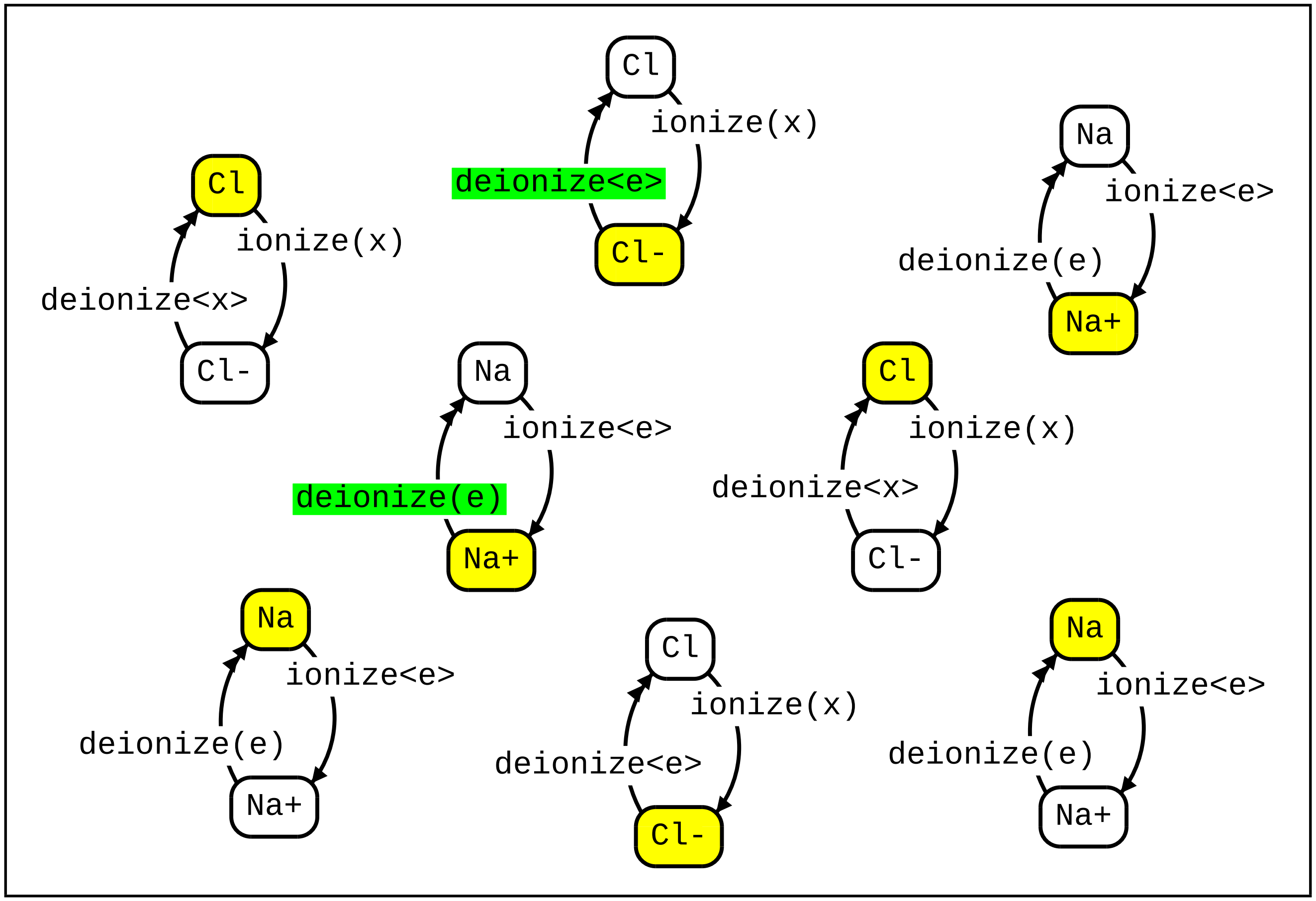
# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



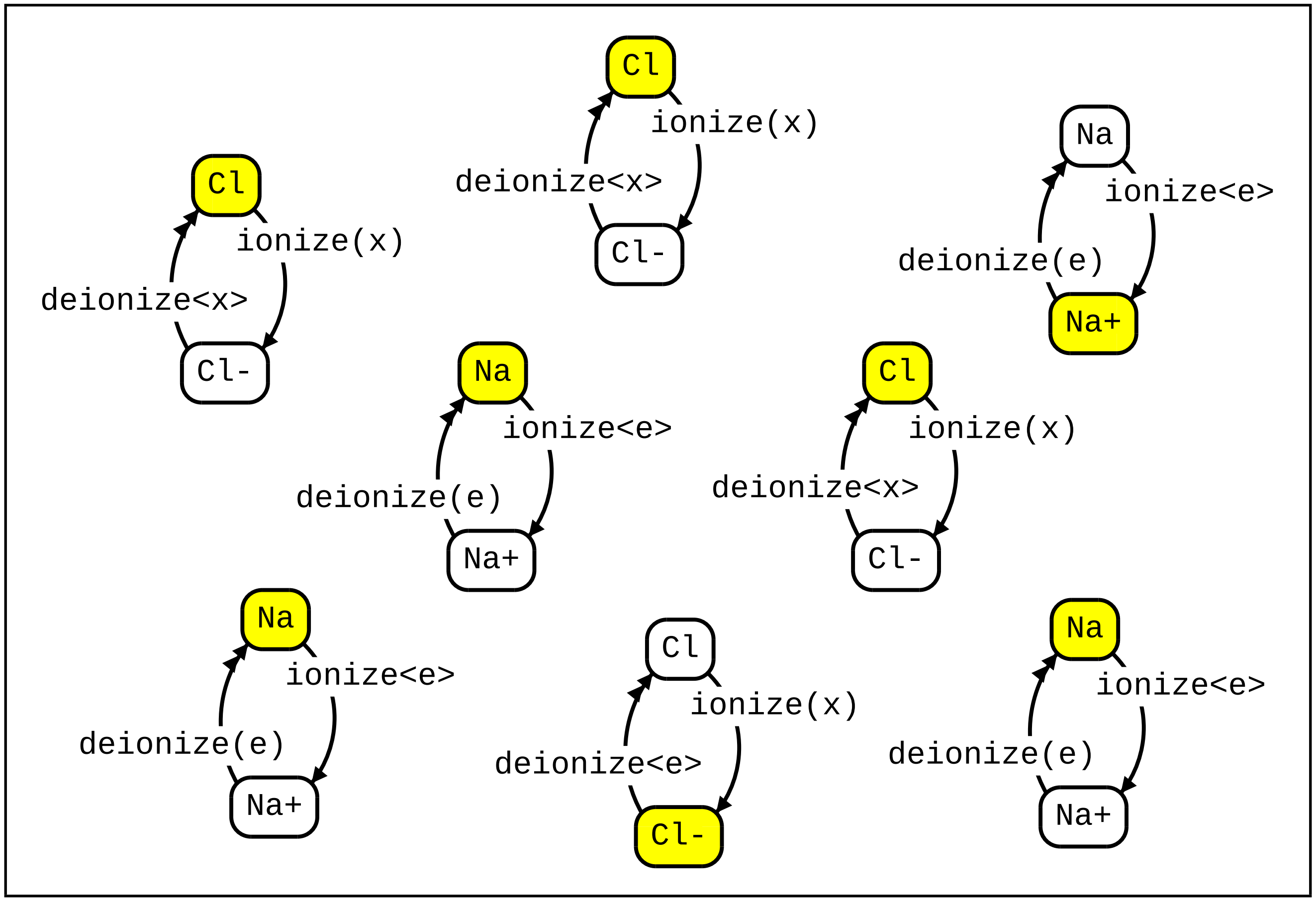
# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



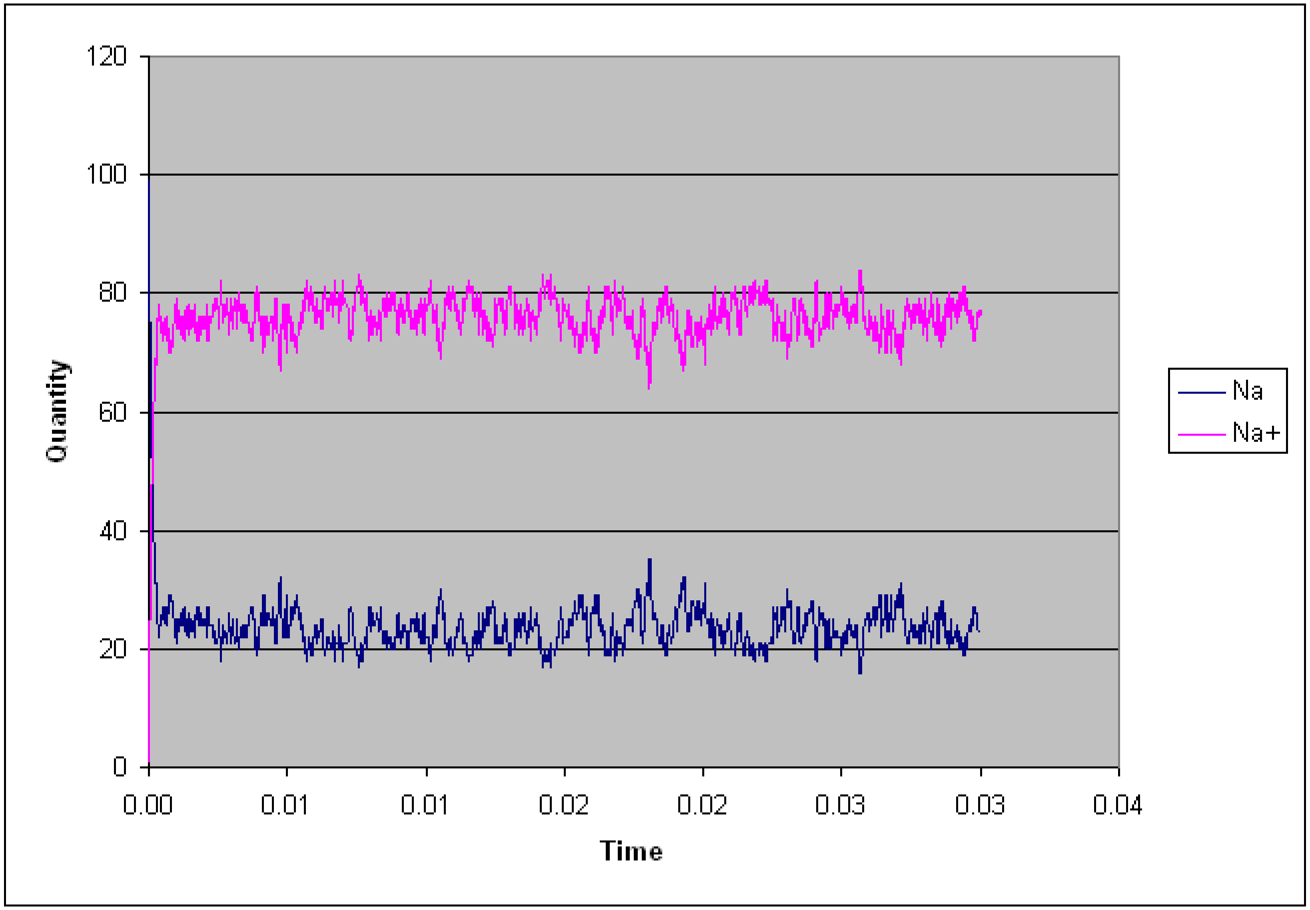
# Ionization: $Na + Cl \rightleftharpoons Na^{+} + Cl^{-}$



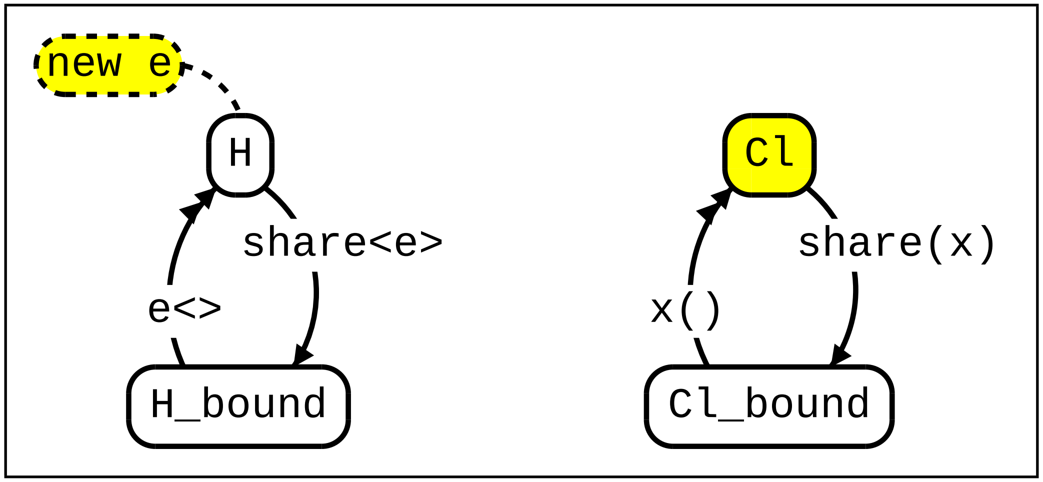
# Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$



# Virtual Experiment: $Na + Cl \rightleftharpoons Na^+ + Cl^-$

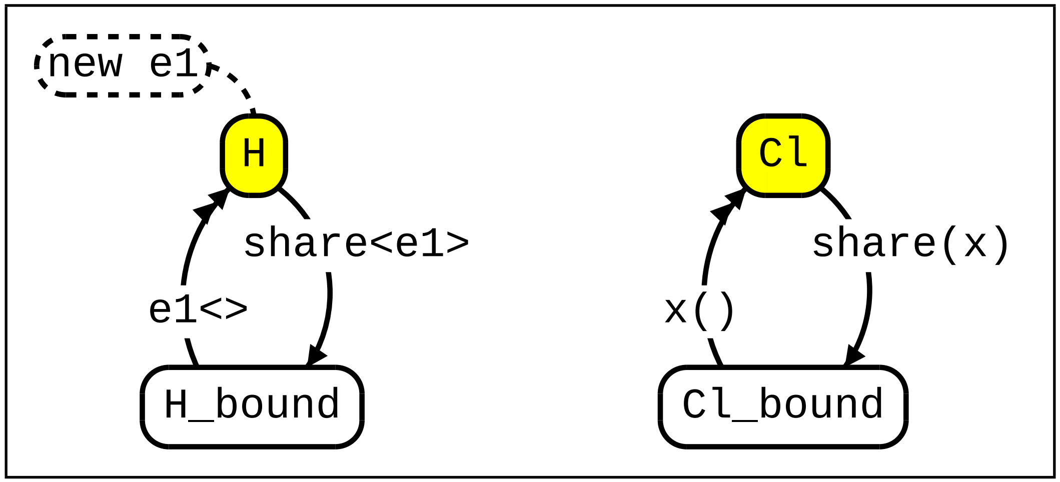


# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



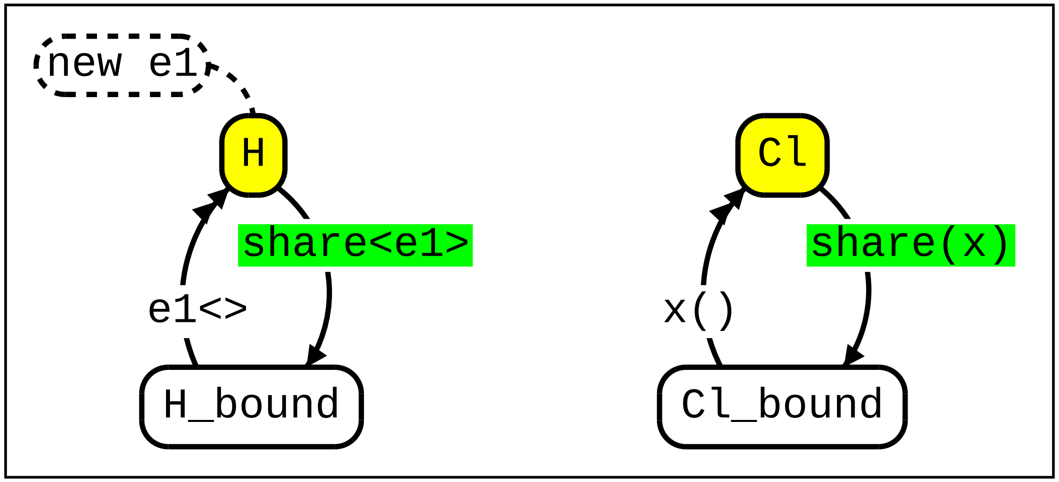

- $H$ has a *private* electron.
- $H$ can share its electron with $Cl$ to form a covalent bond with rate $100s^{-1}$
- $HCl$ can break its private bond with rate $10s^{-1}$

# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



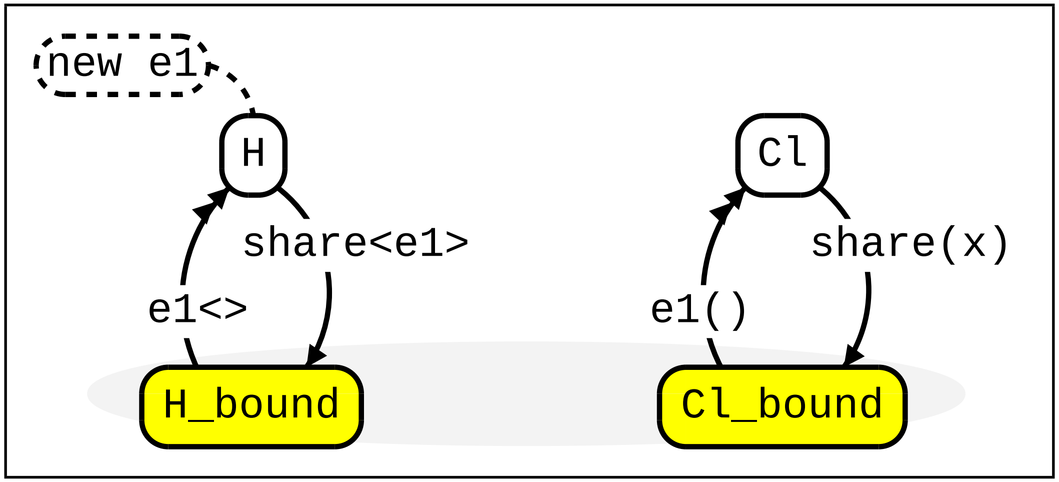

- $H$ has a private electron $e1$ that is not accessible from outside.

# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



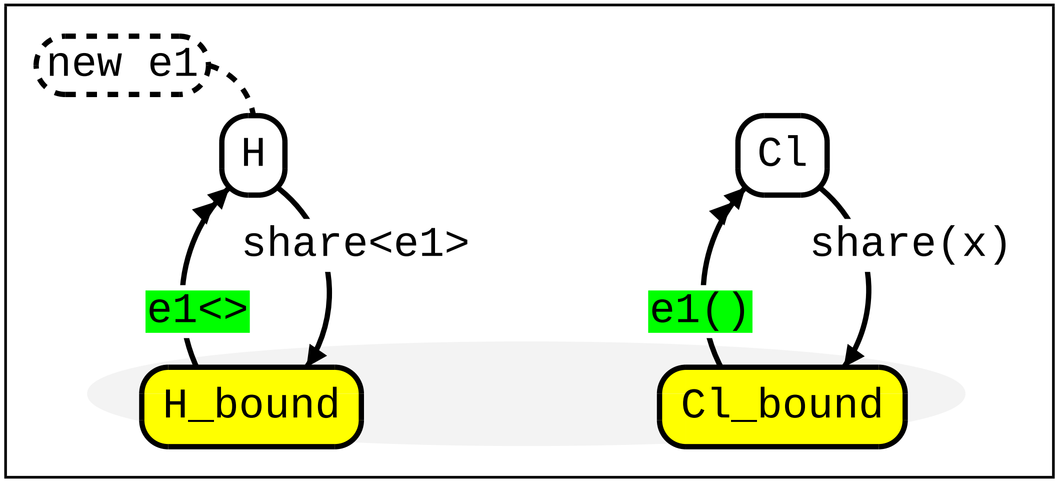

- $H$ can share its electron with $Cl$ on the $share$ channel.

## Covalent Bonding: $H + Cl \rightleftharpoons HCl$



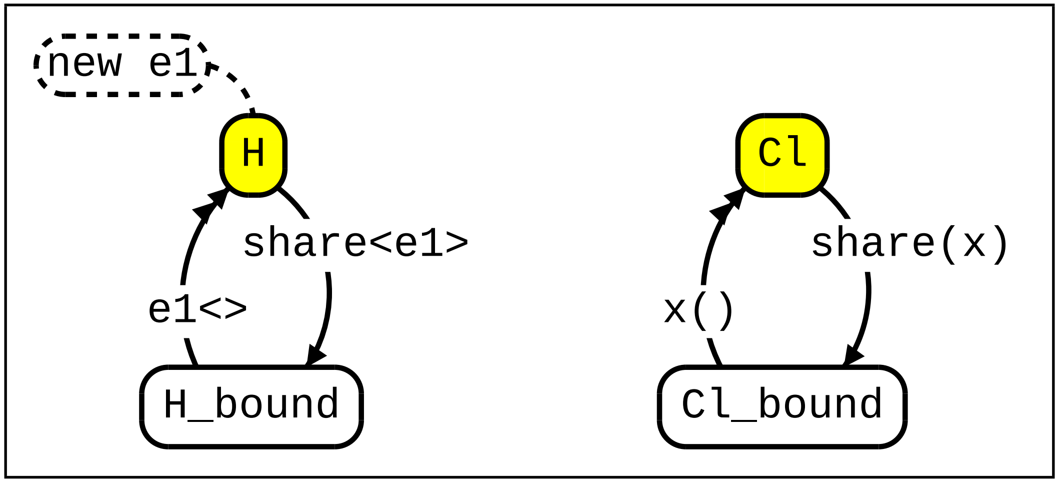

- $H$ and $Cl$ share a private electron, to form $HCl$.

# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



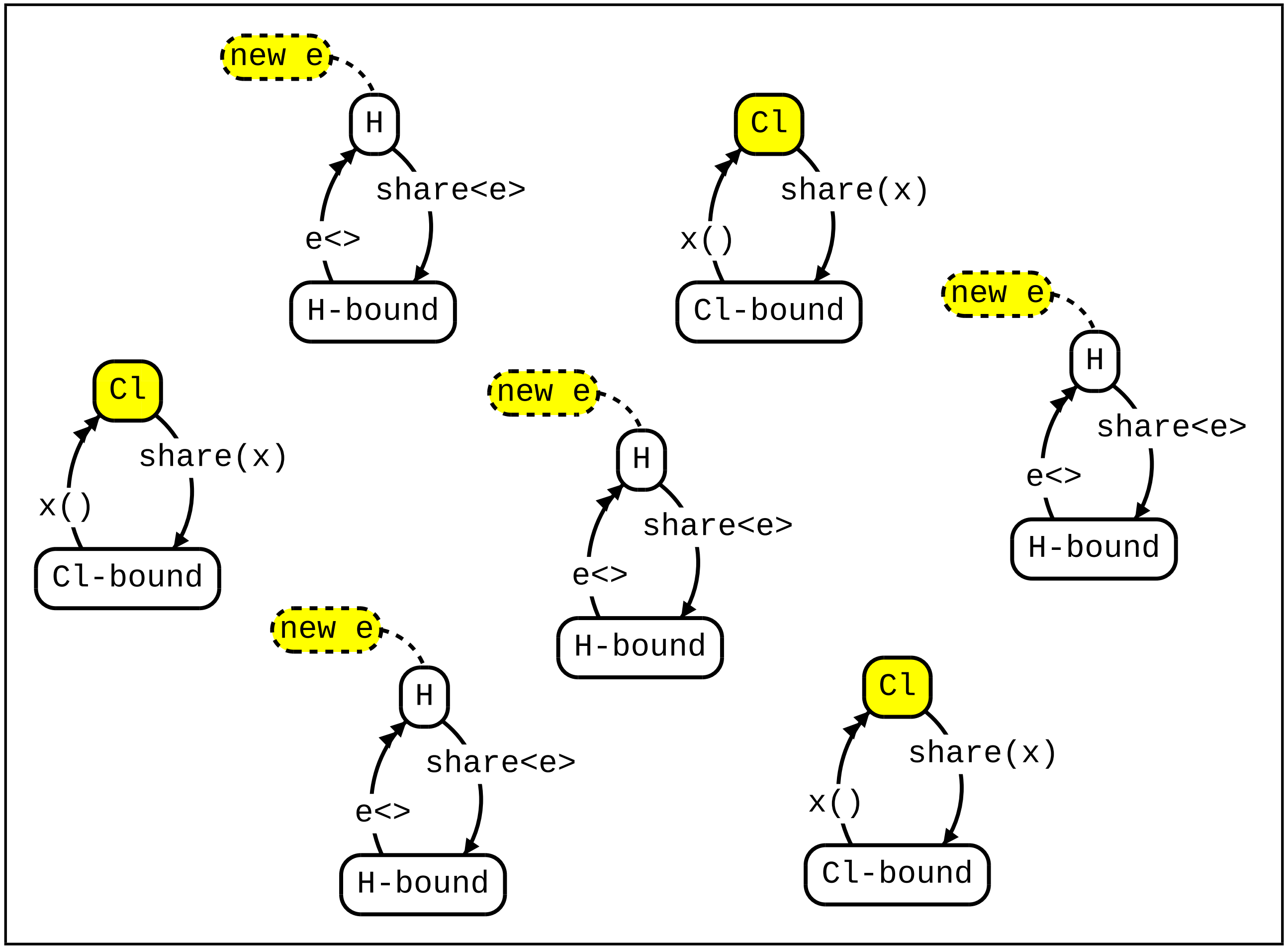

➢ $HCl$ can break its private bond by synchronising on channel $e1$.

# Covalent Bonding: $H + Cl \rightleftharpoons HCl$

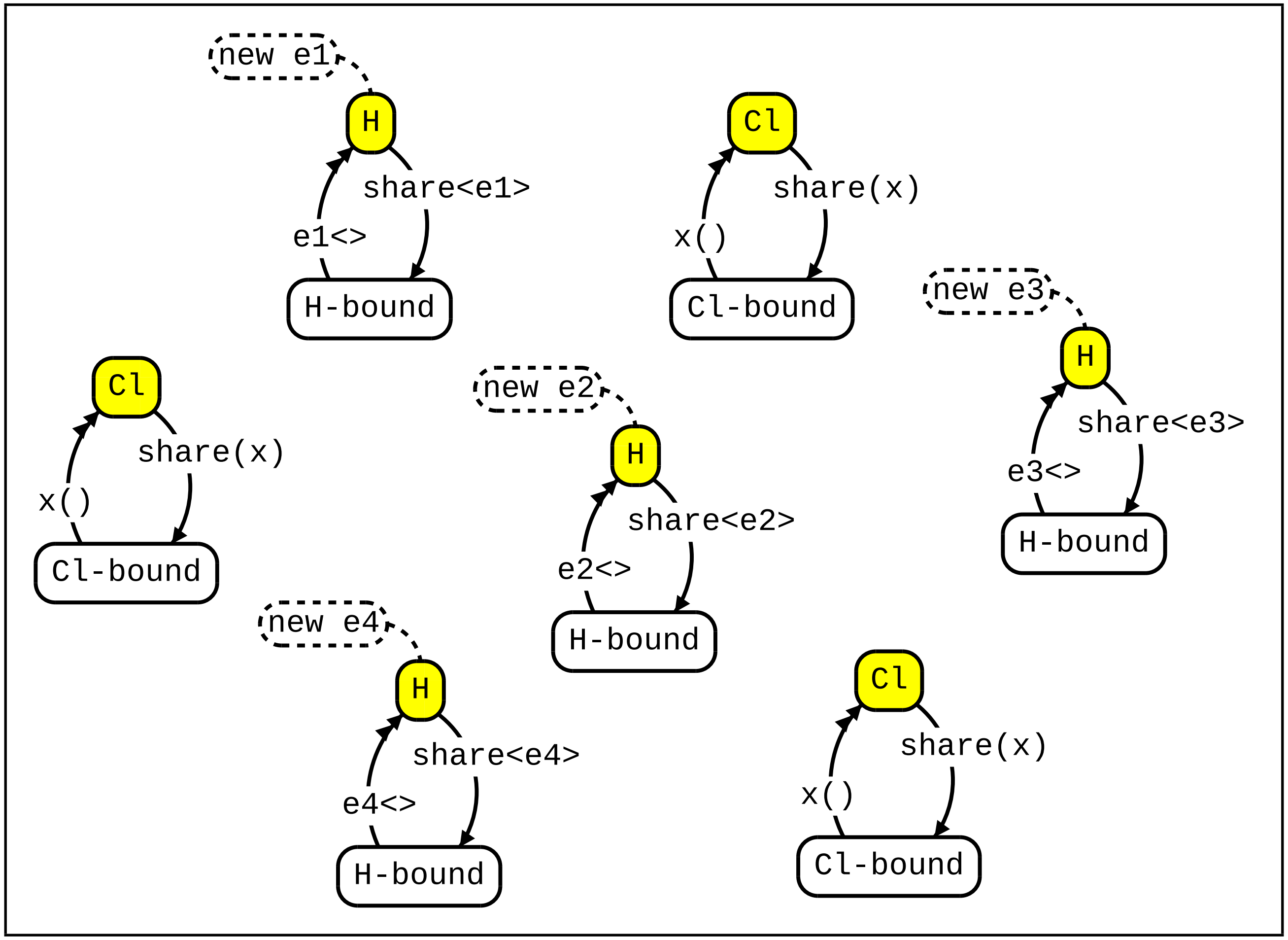



- $H$ and $Cl$ are no longer bound

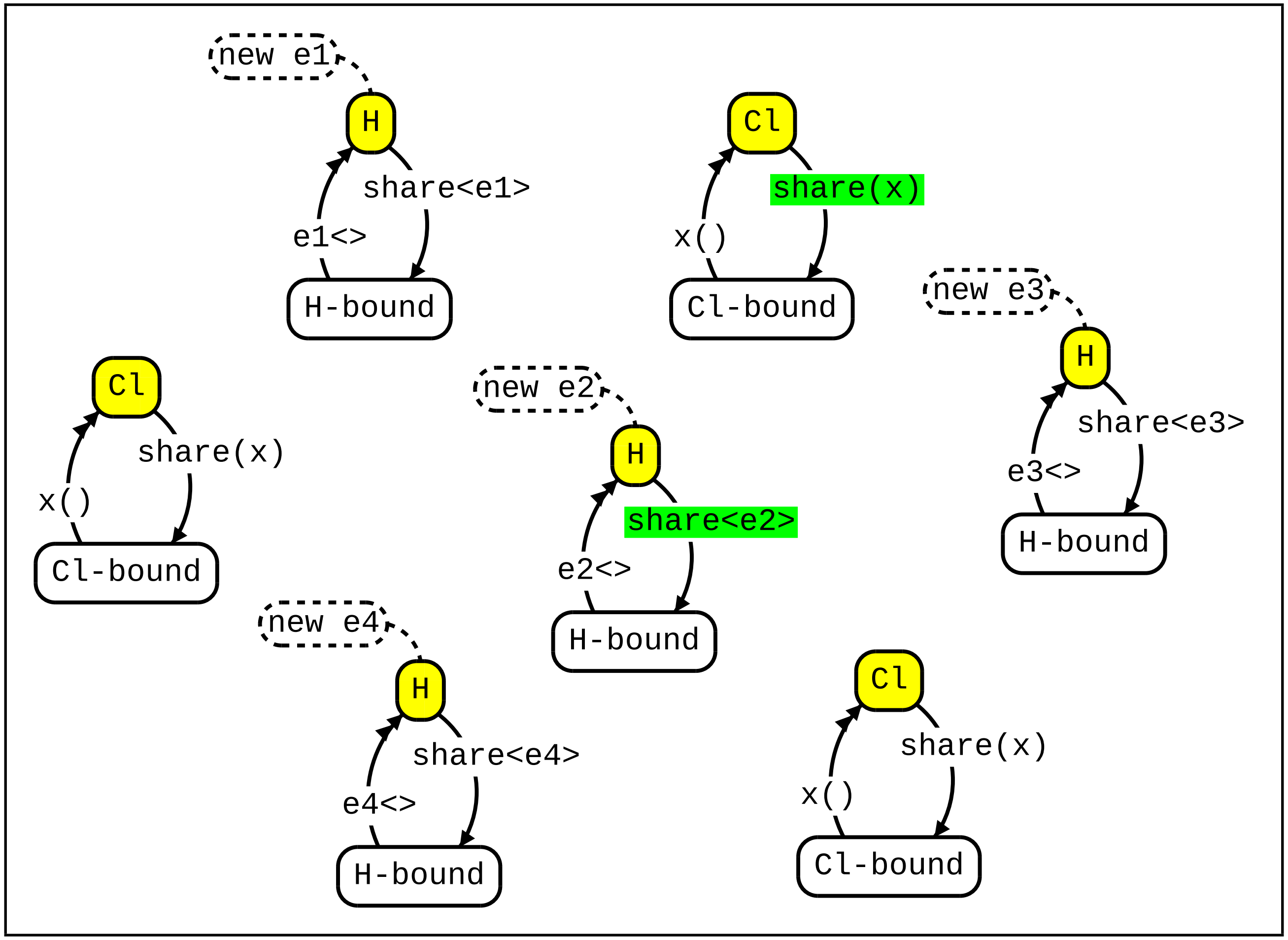
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



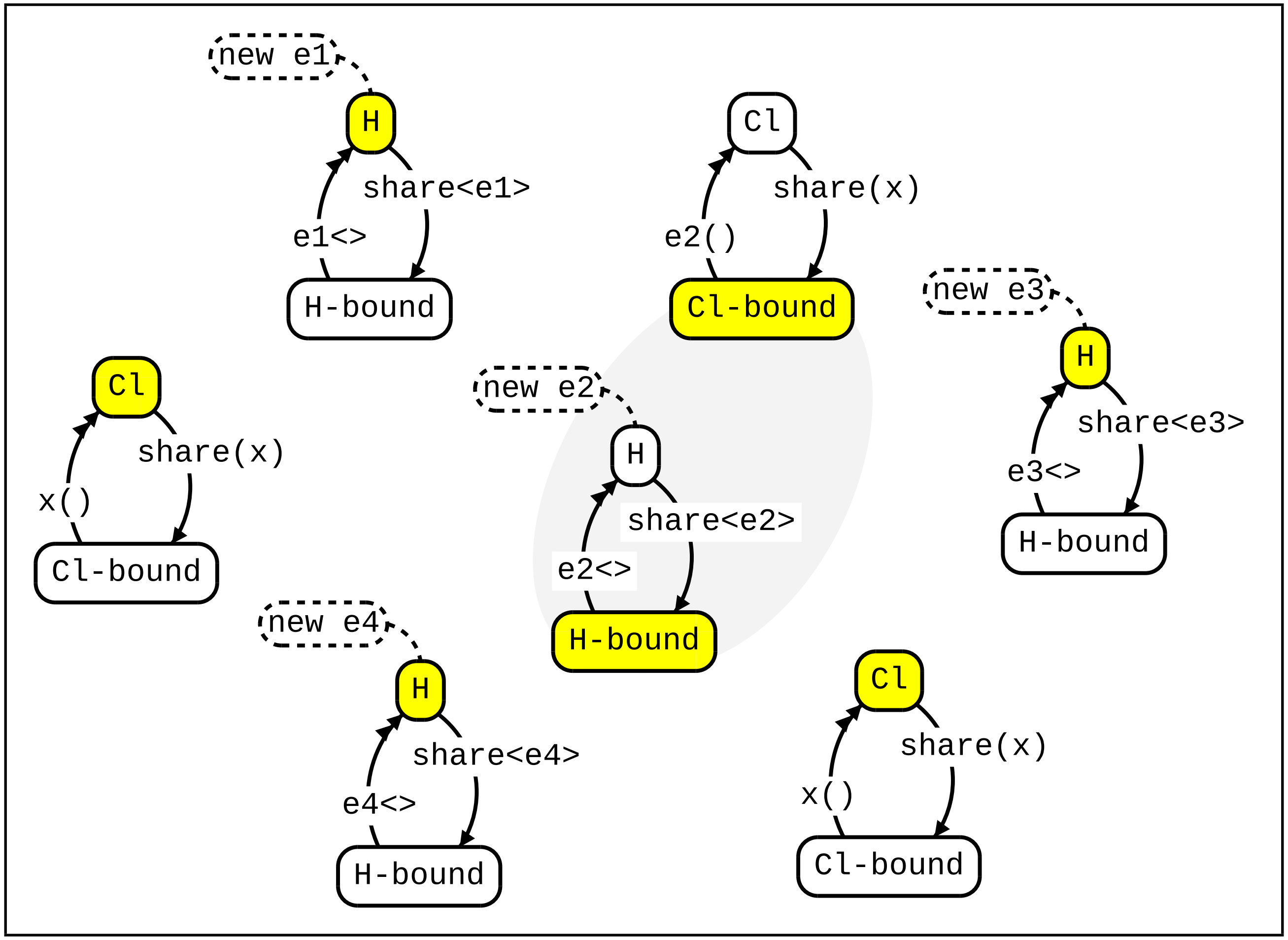
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



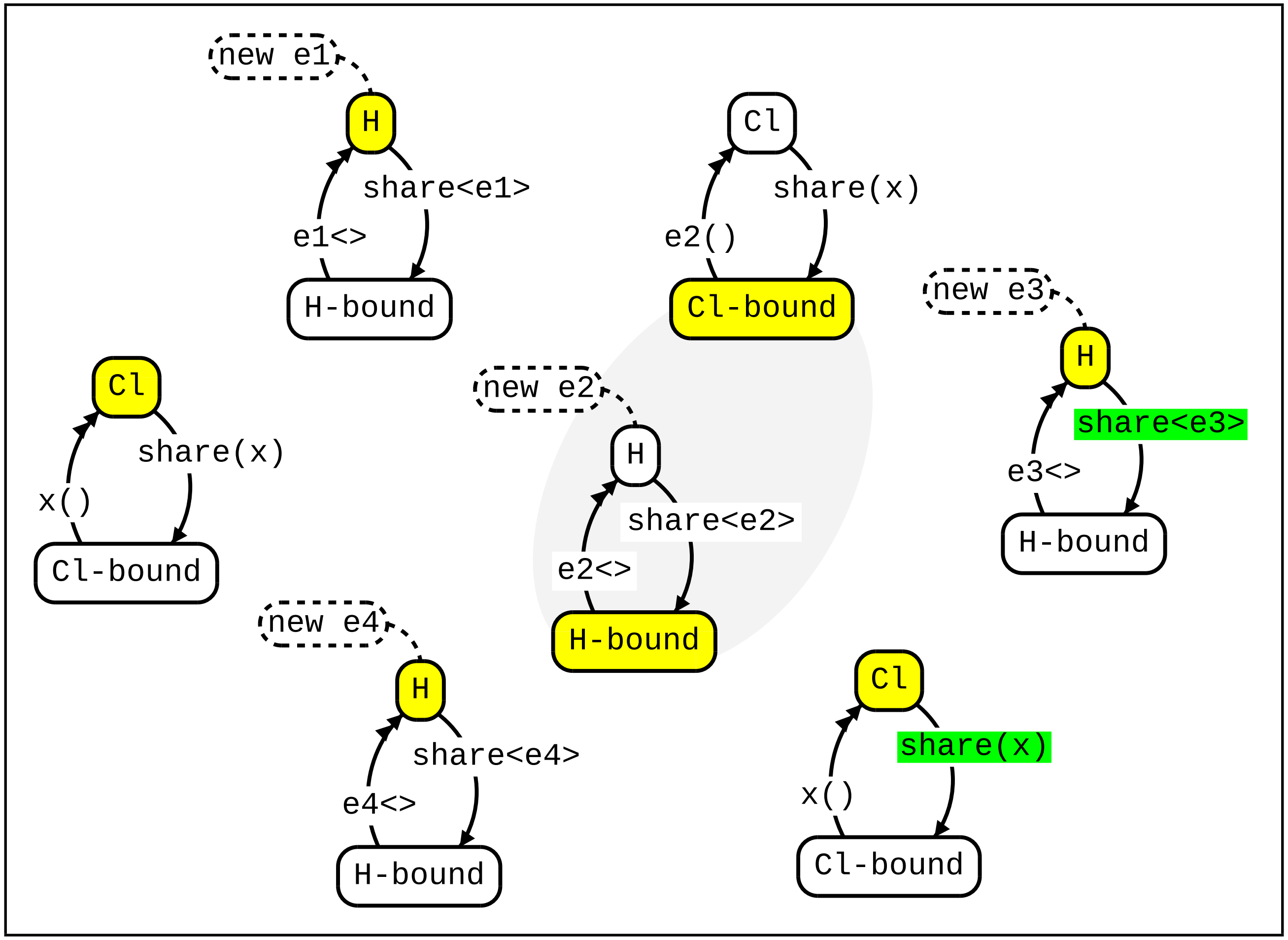
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



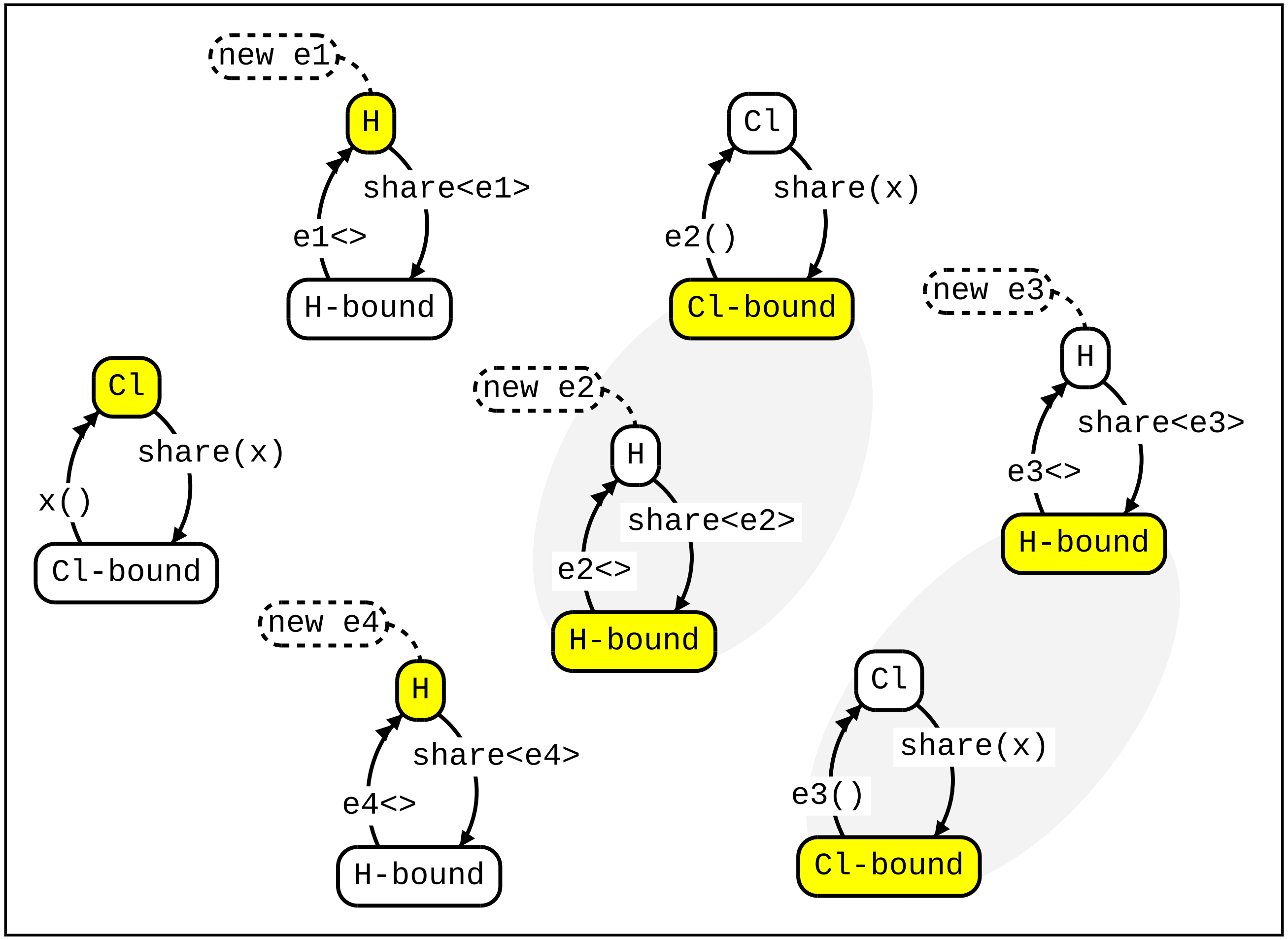
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



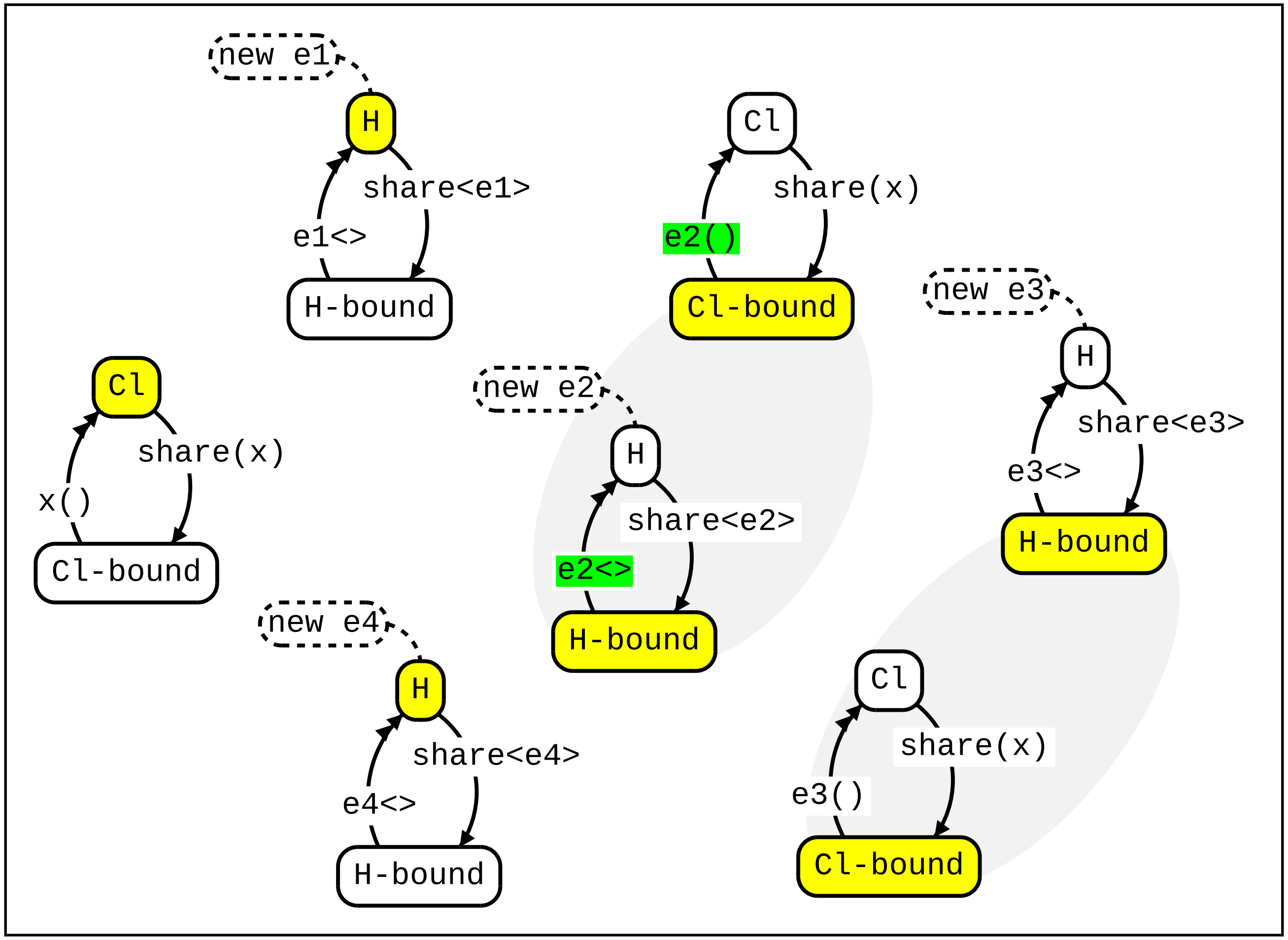
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



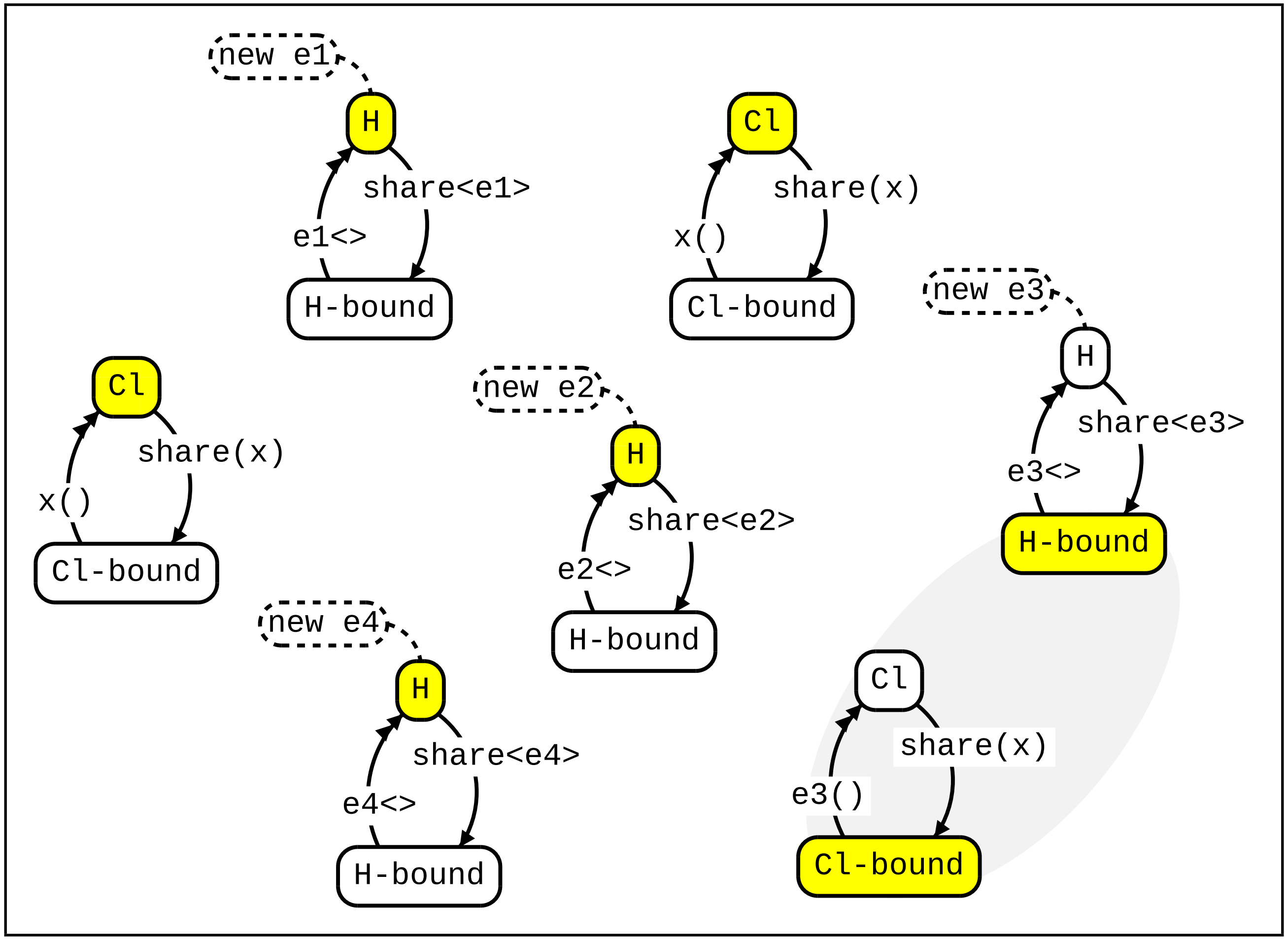
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



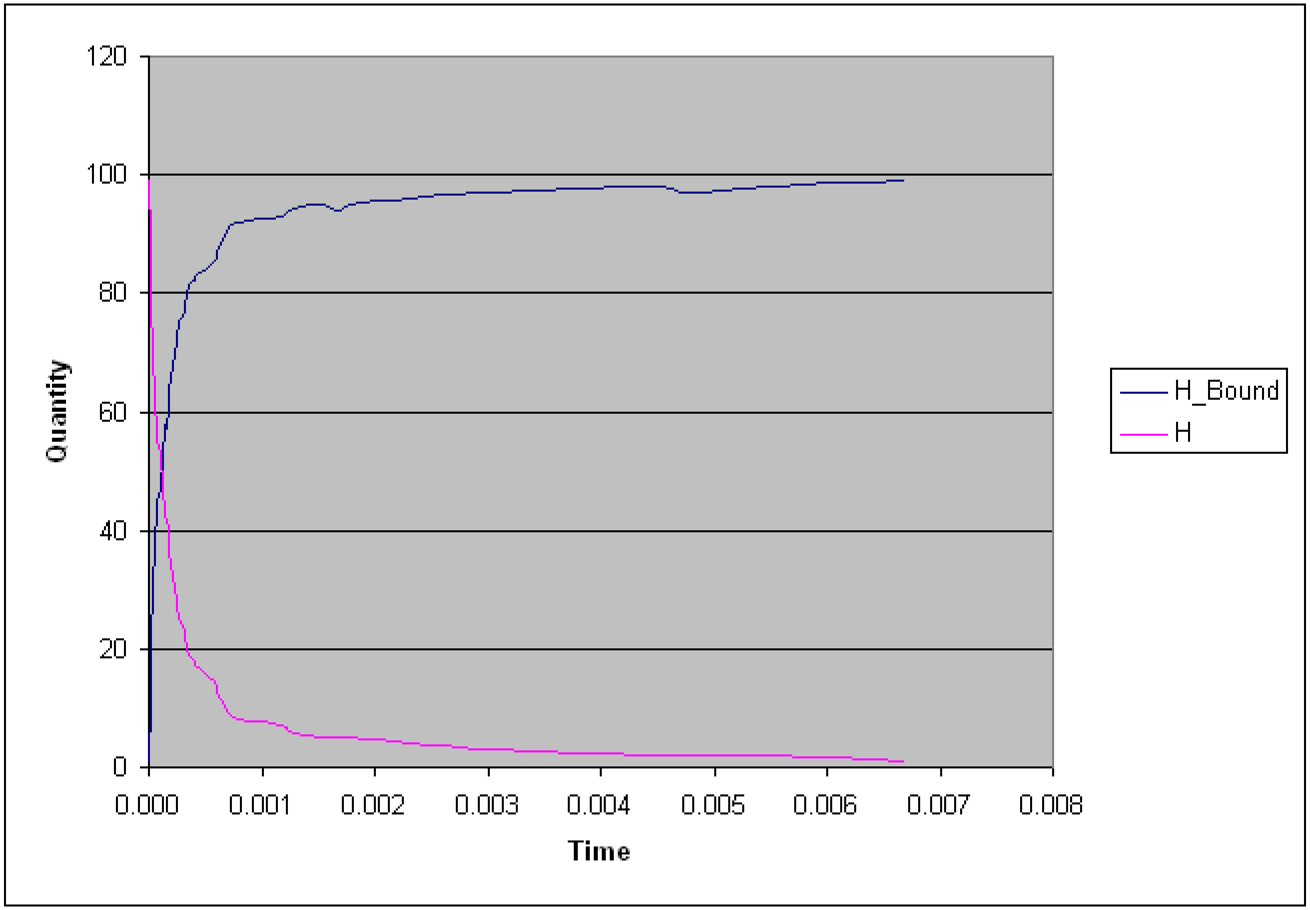
# Covalent Bonding: $H + Cl \rightleftharpoons HCl$



# Covalent Bonding: $H + Cl \rightleftharpoons HCl$

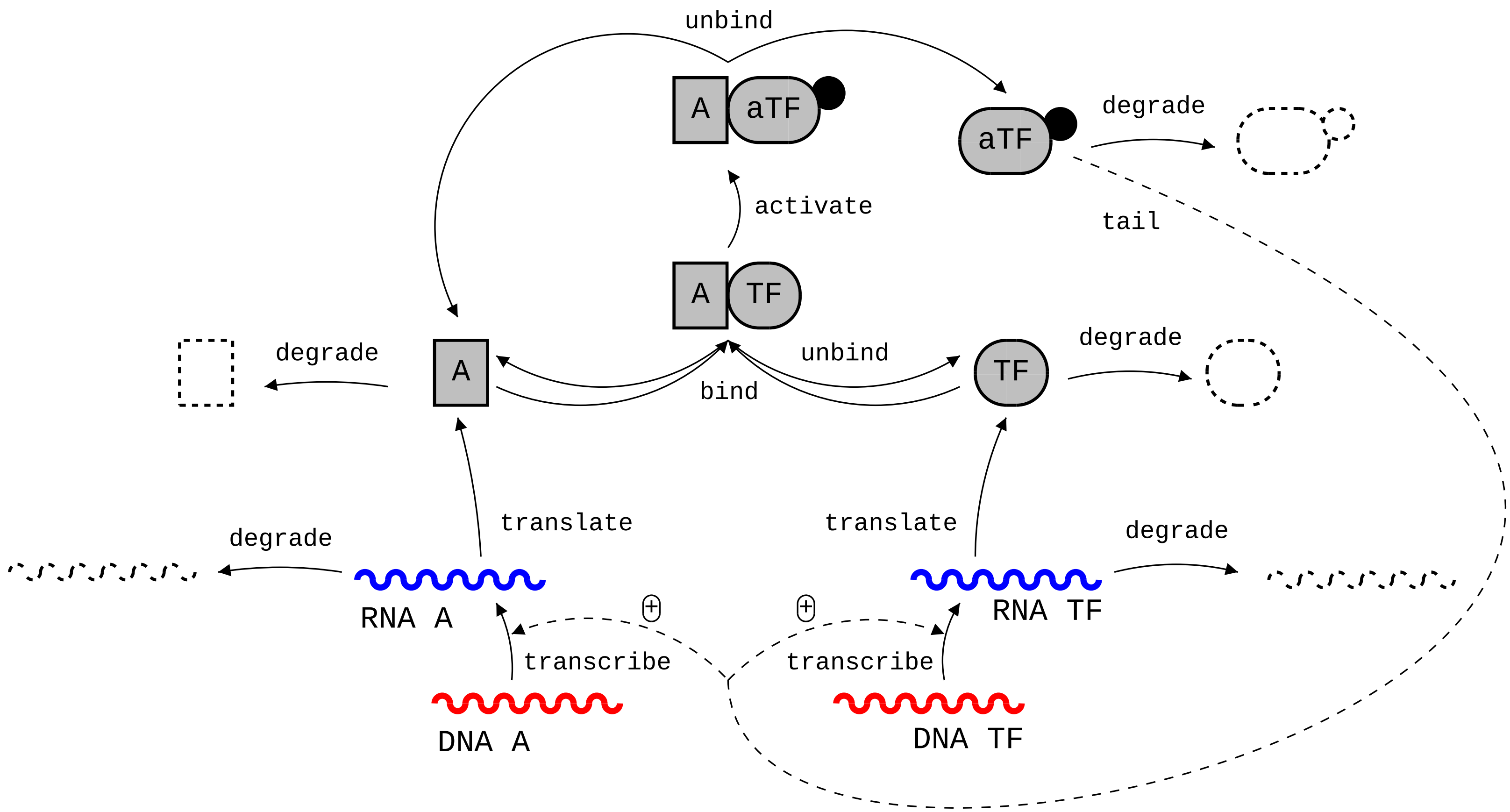


# Virtual Experiment: $H + Cl \rightleftharpoons HCl$

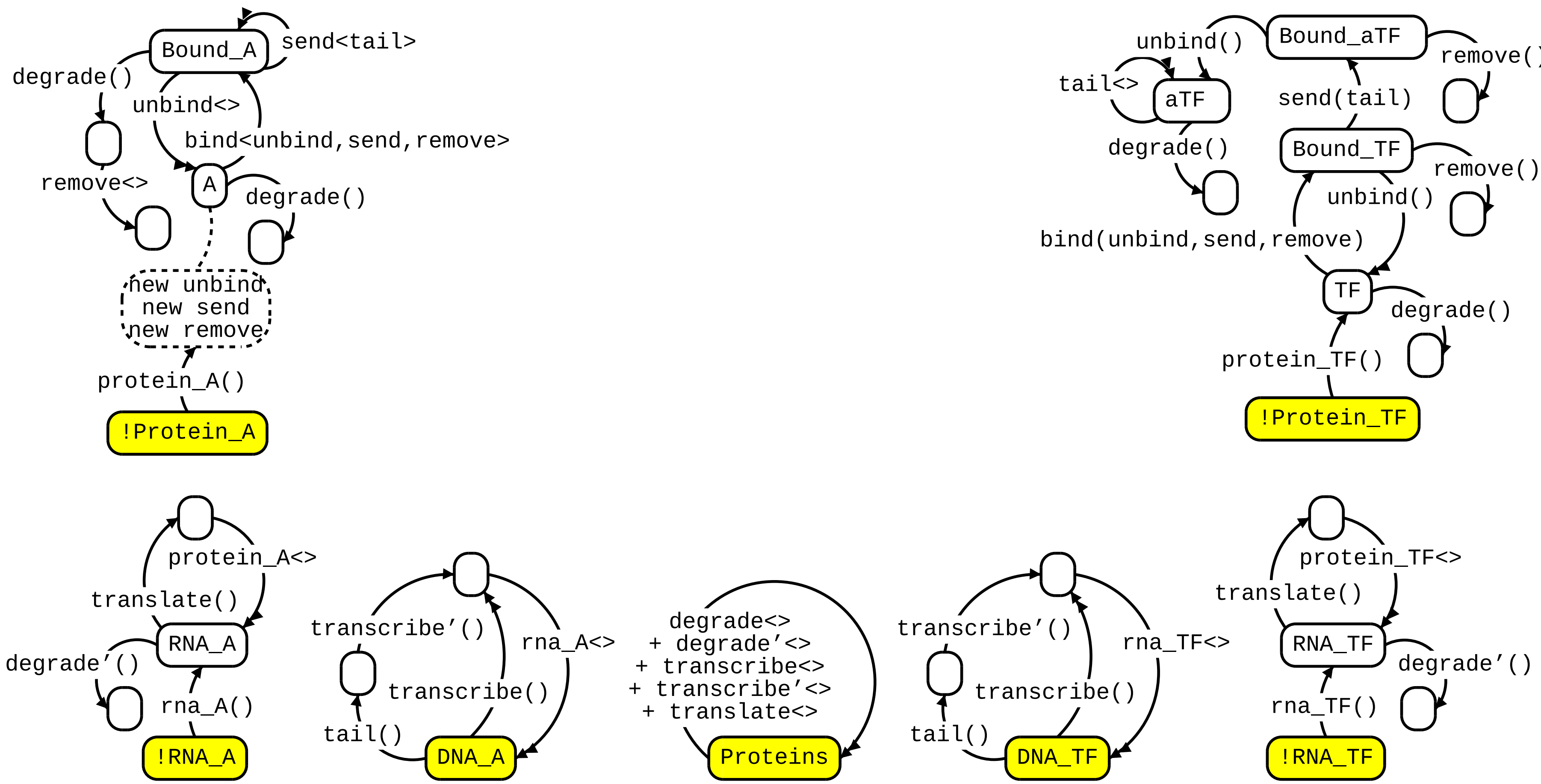


# Gene Regulation by Positive Feedback

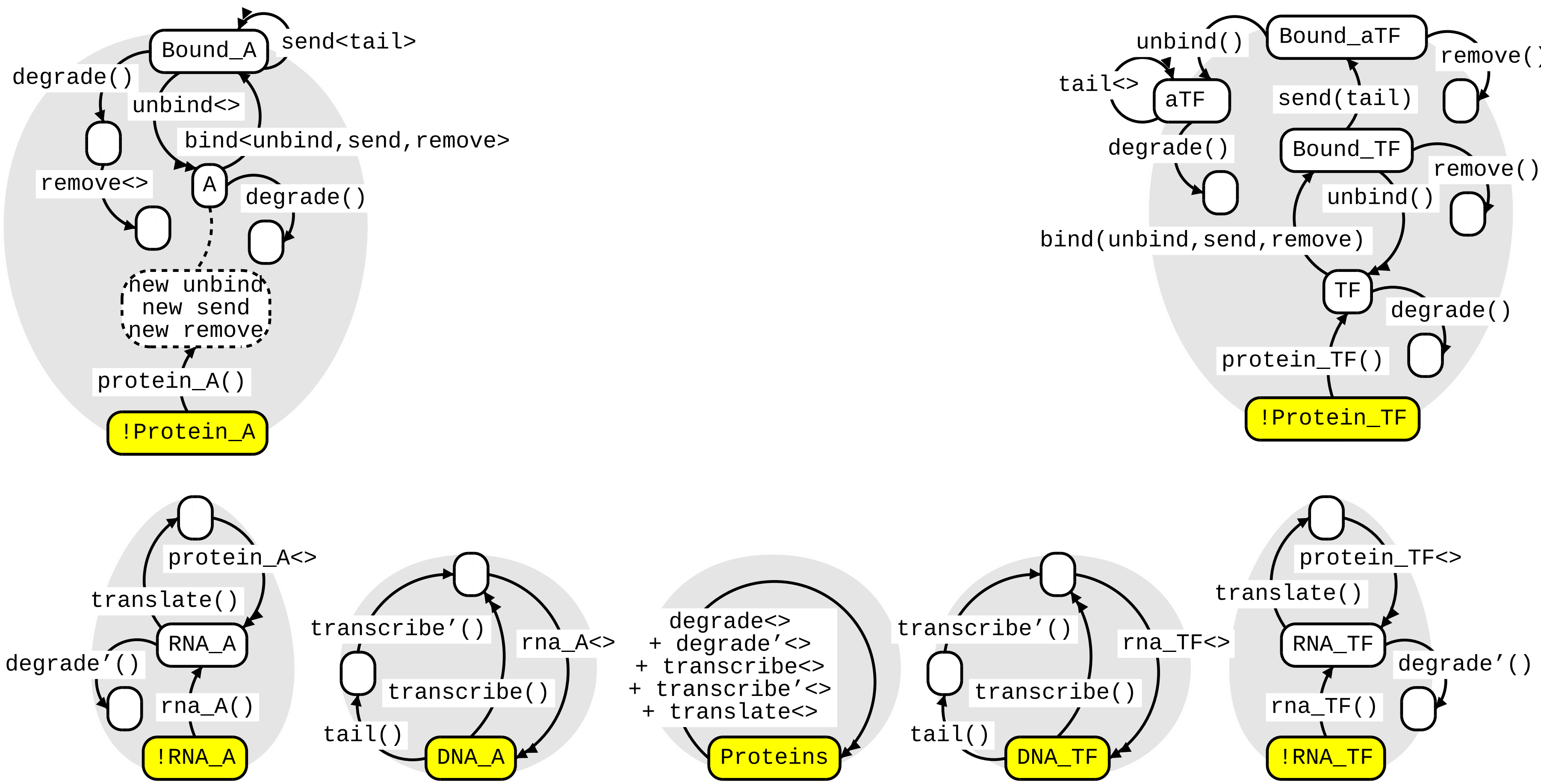
[Priami et al., 2001]



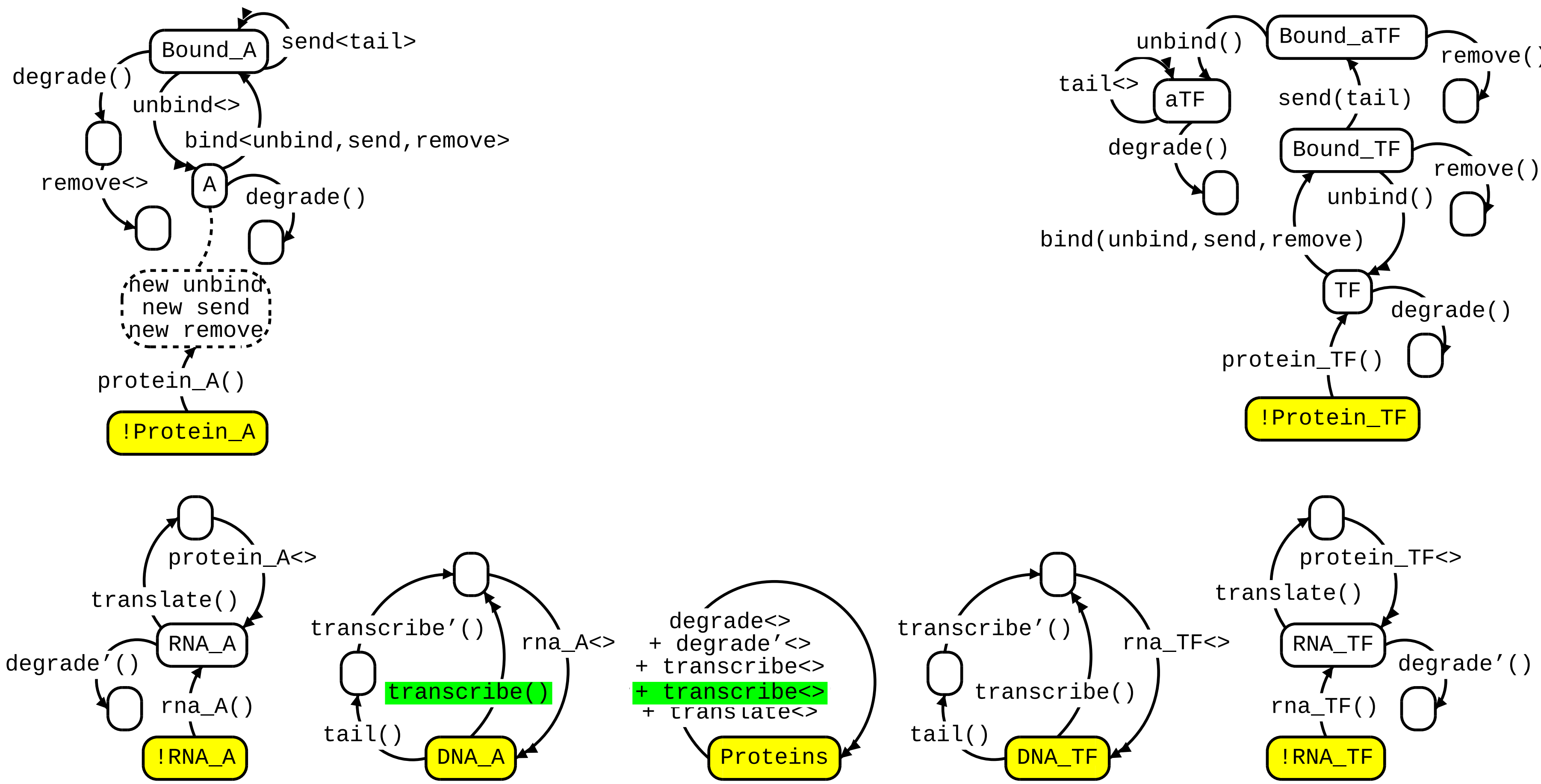
# Gene Regulation by Positive Feedback



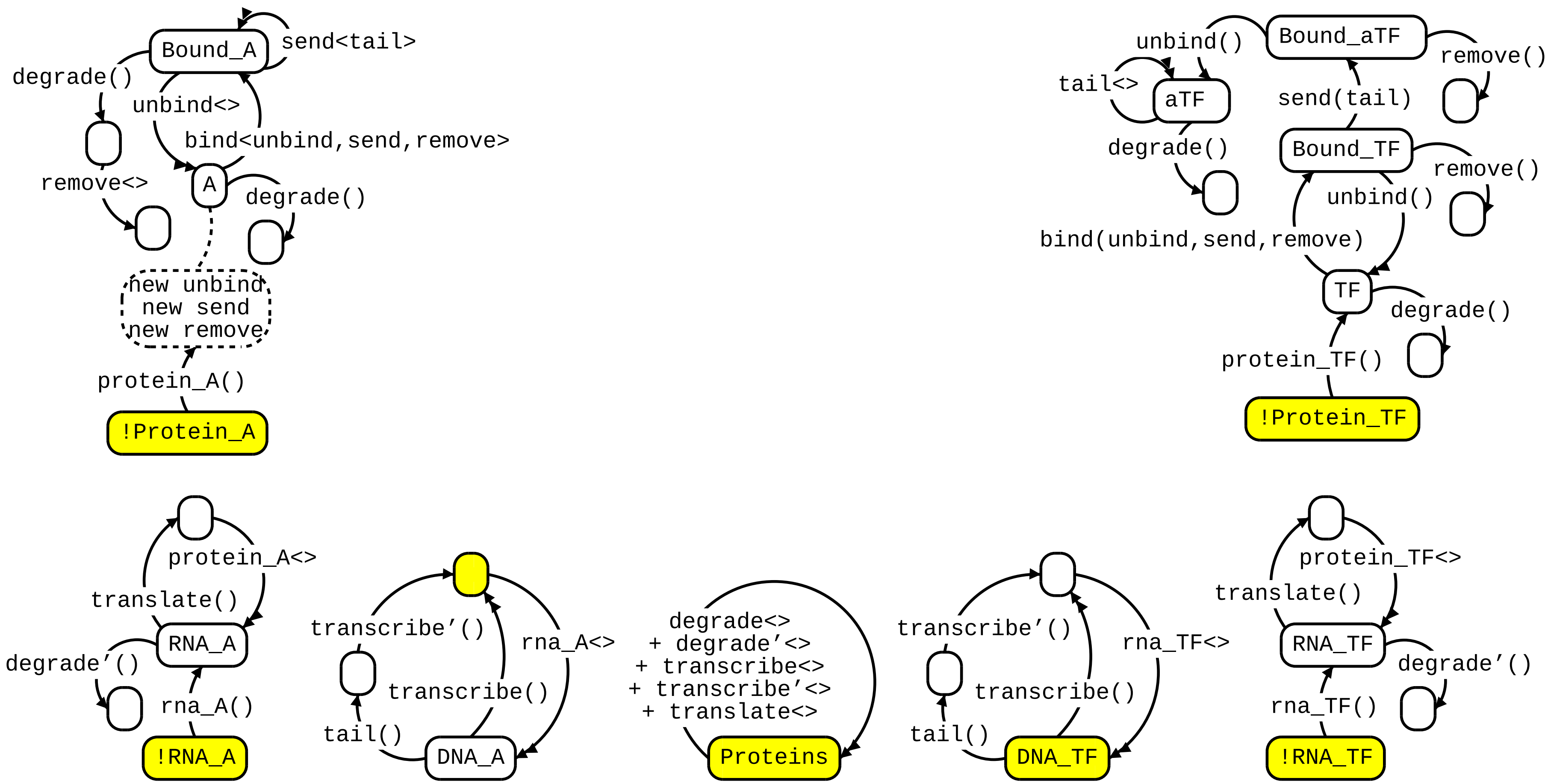
# Gene Regulation by Positive Feedback



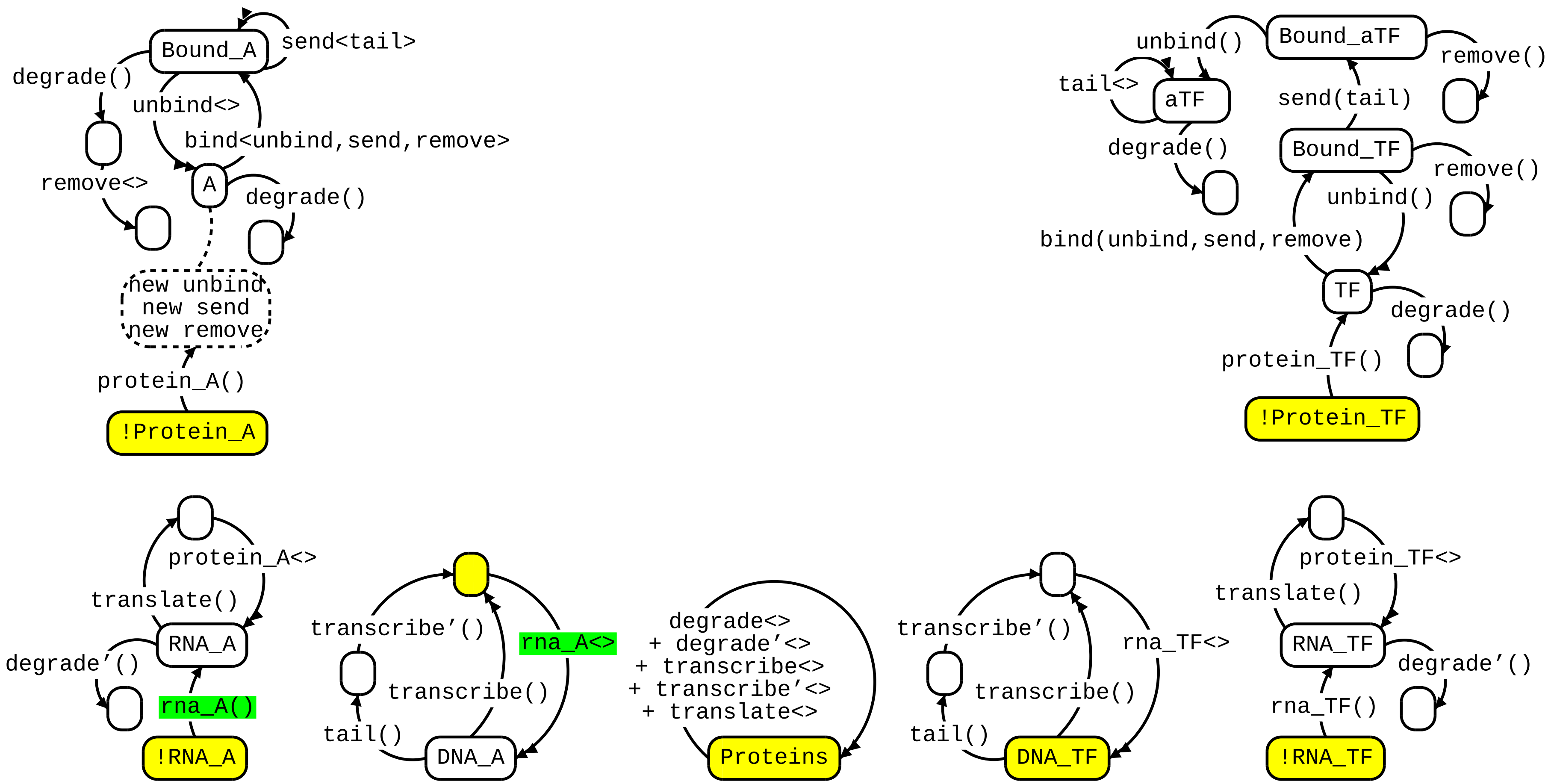
# Gene Regulation by Positive Feedback



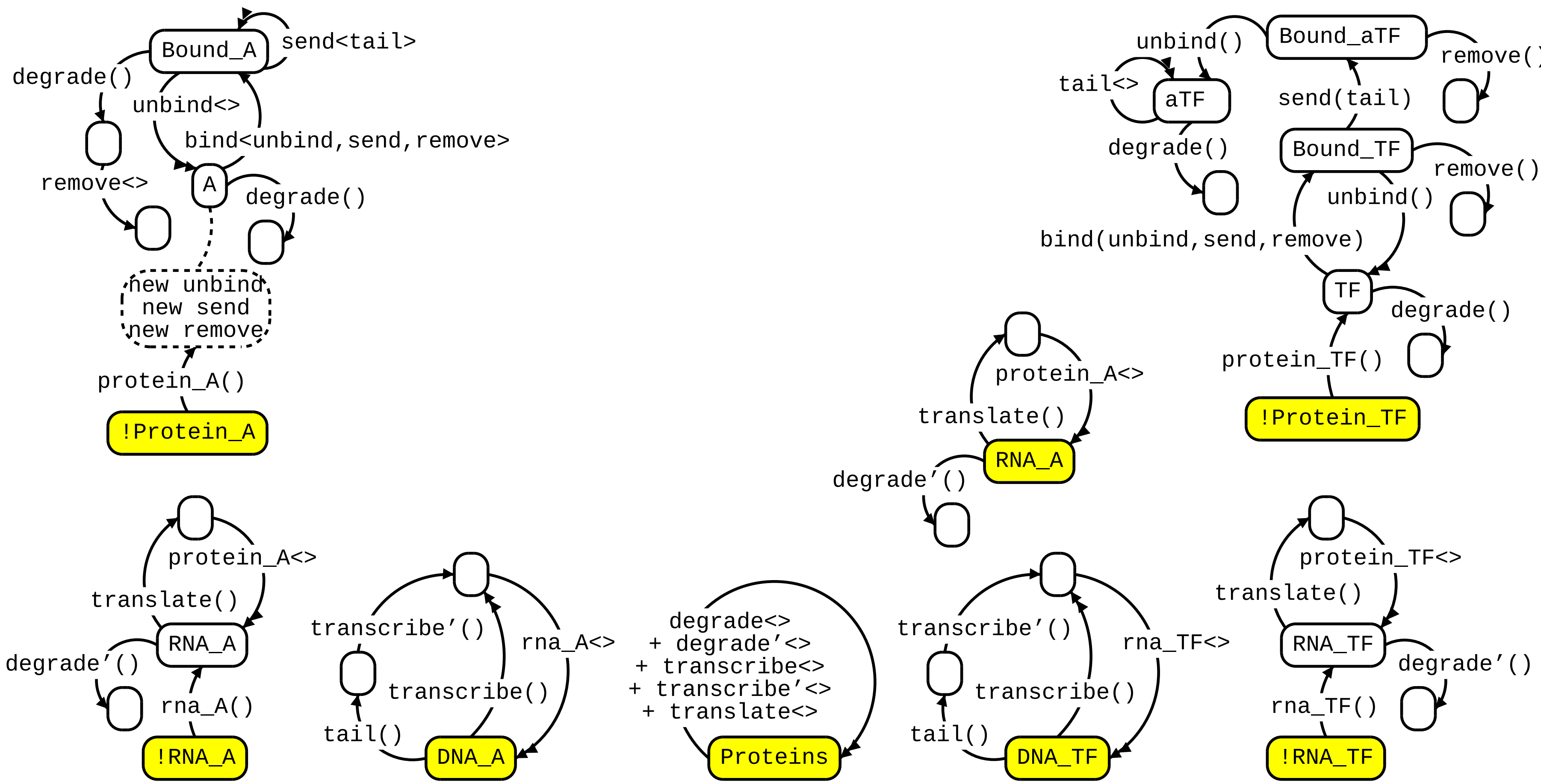
# Gene Regulation by Positive Feedback



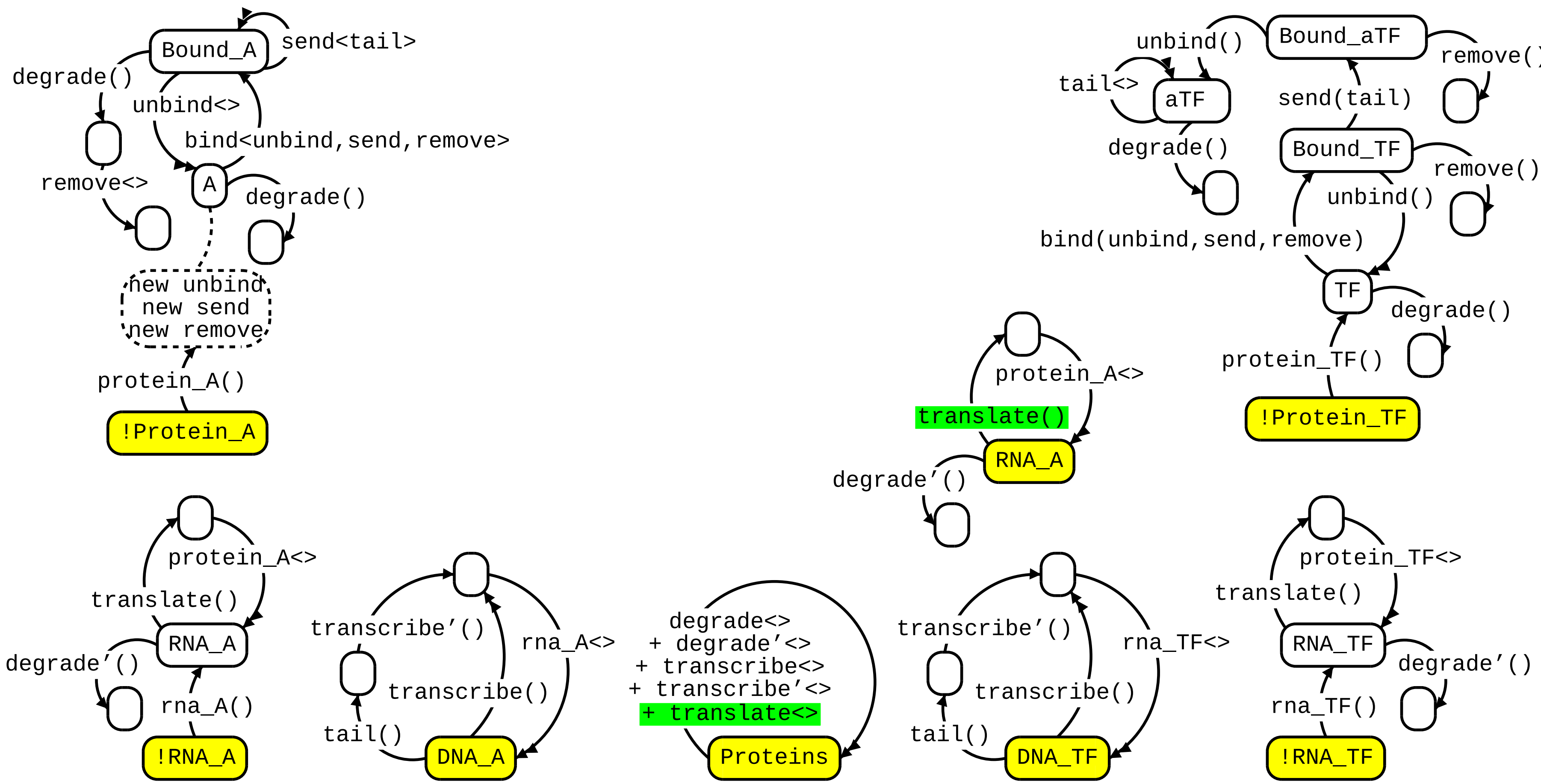
# Gene Regulation by Positive Feedback



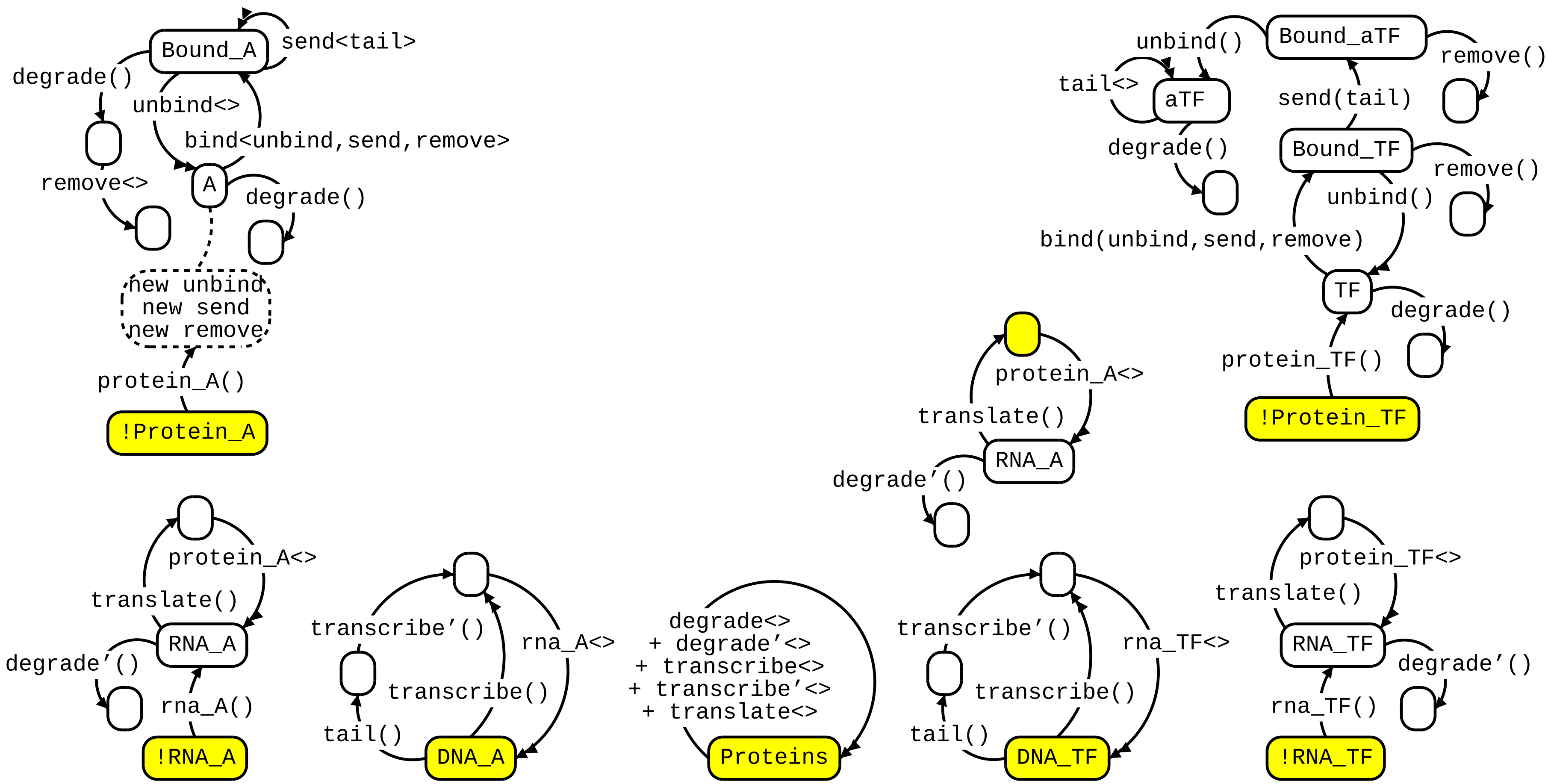
# Gene Regulation by Positive Feedback



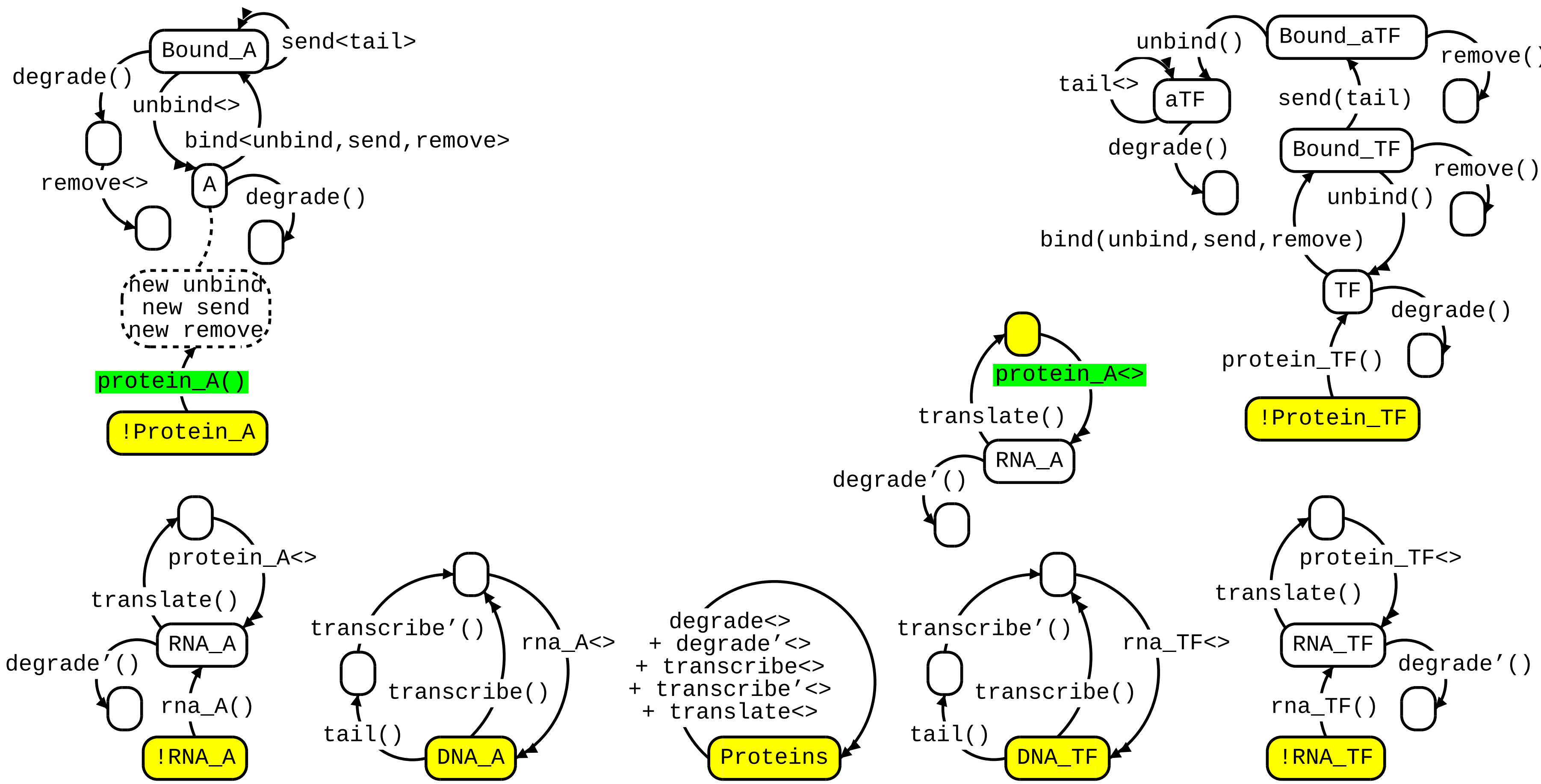
# Gene Regulation by Positive Feedback



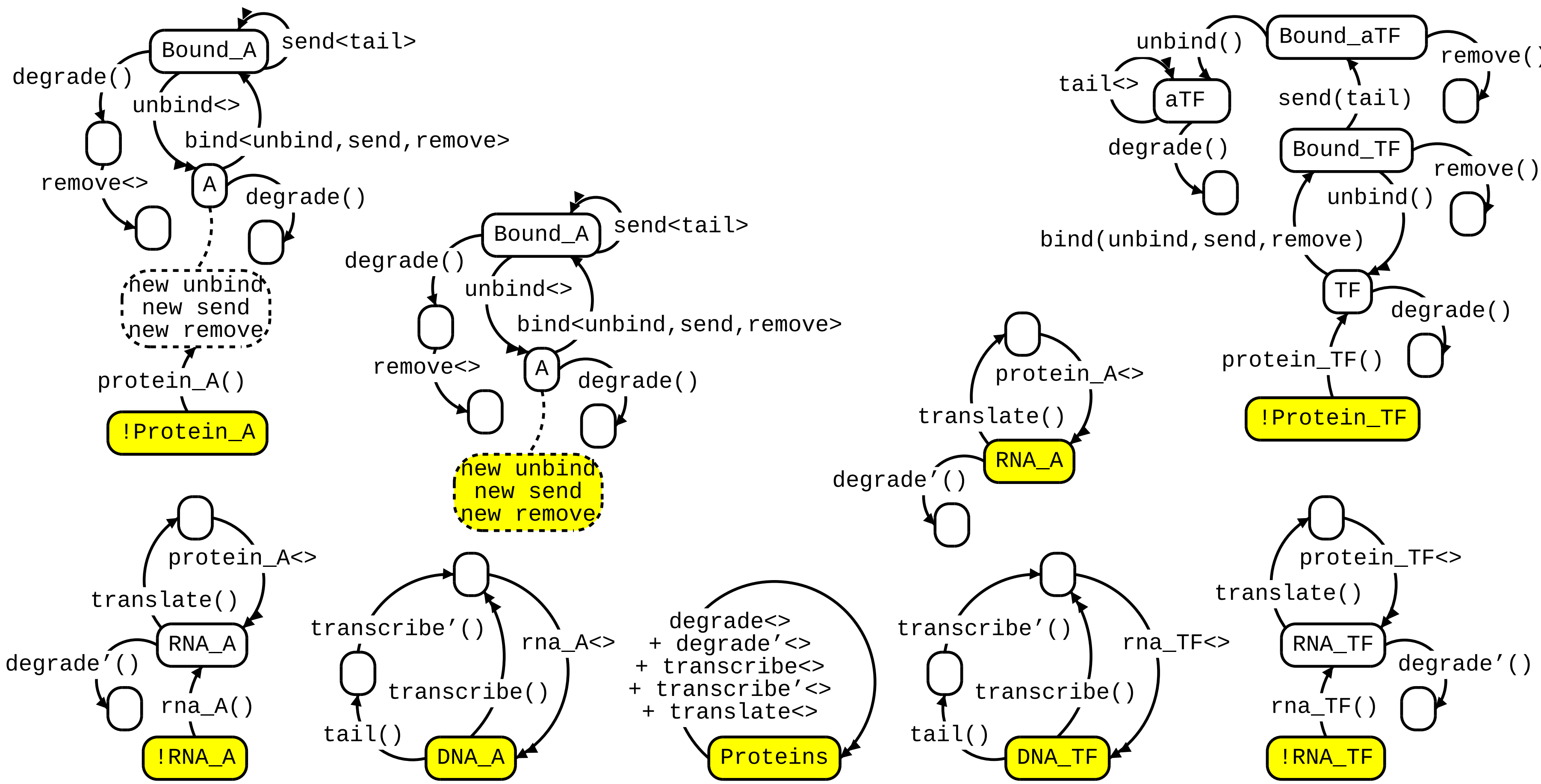
# Gene Regulation by Positive Feedback



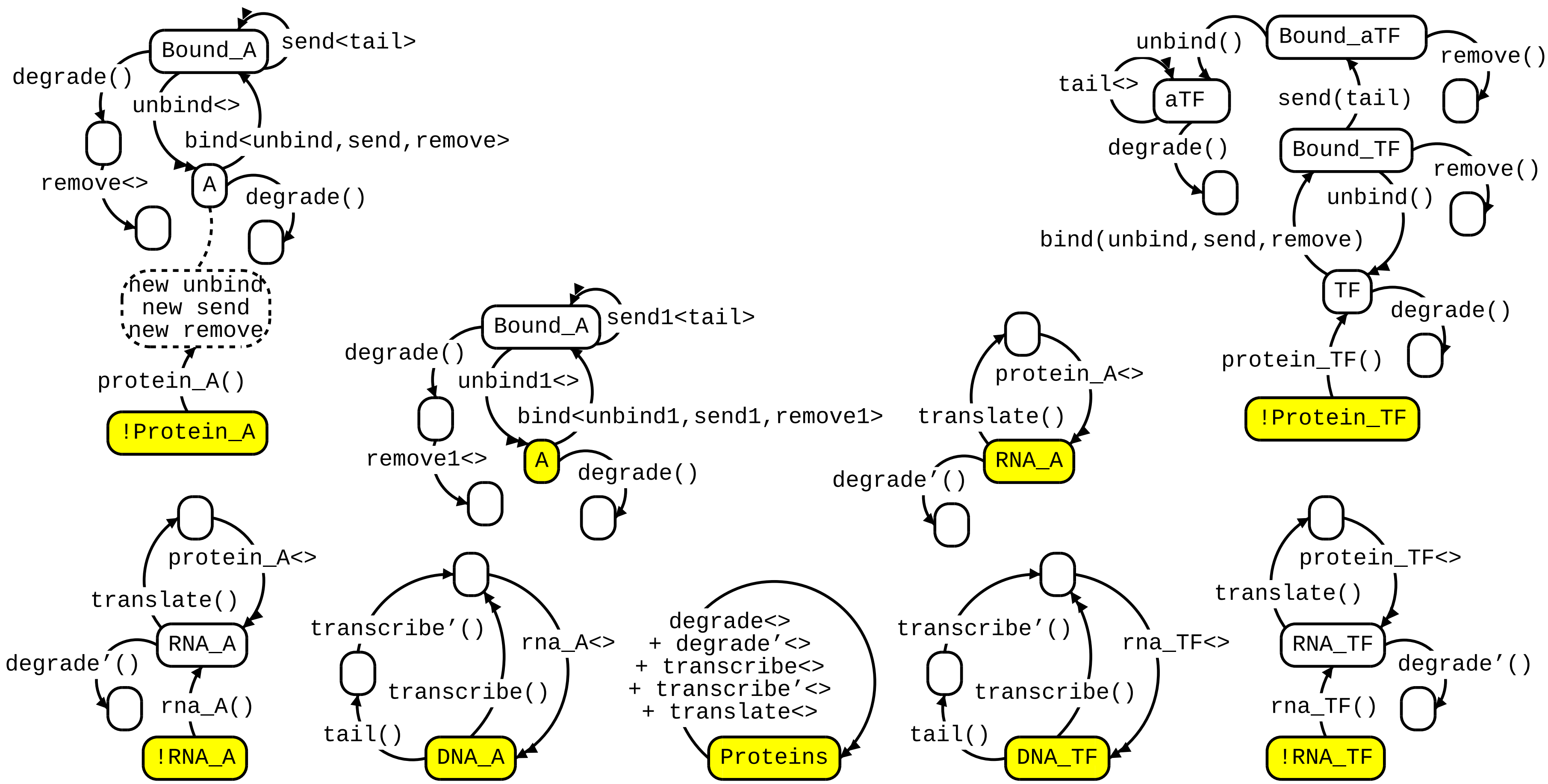
# Gene Regulation by Positive Feedback



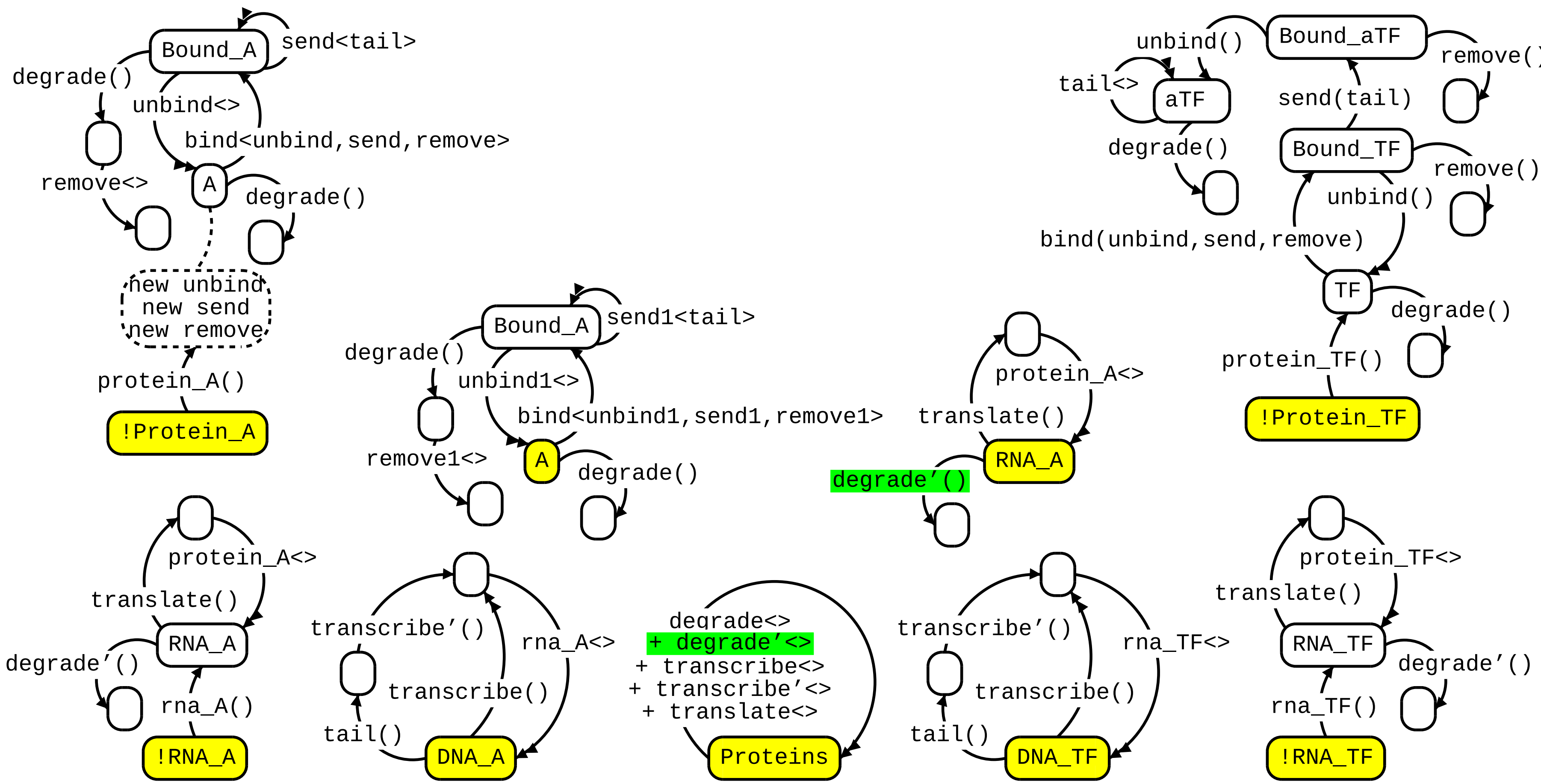
# Gene Regulation by Positive Feedback



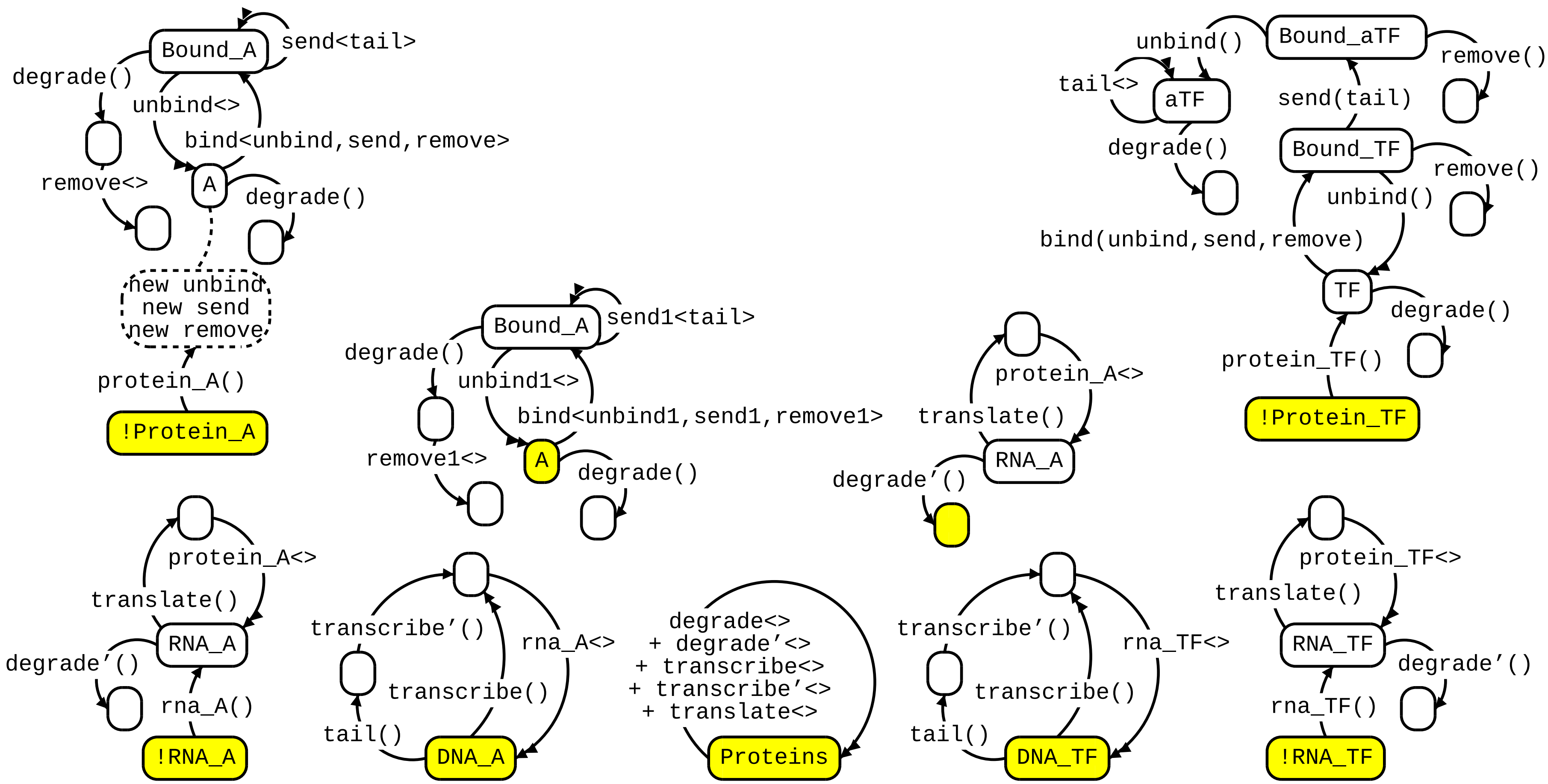
# Gene Regulation by Positive Feedback



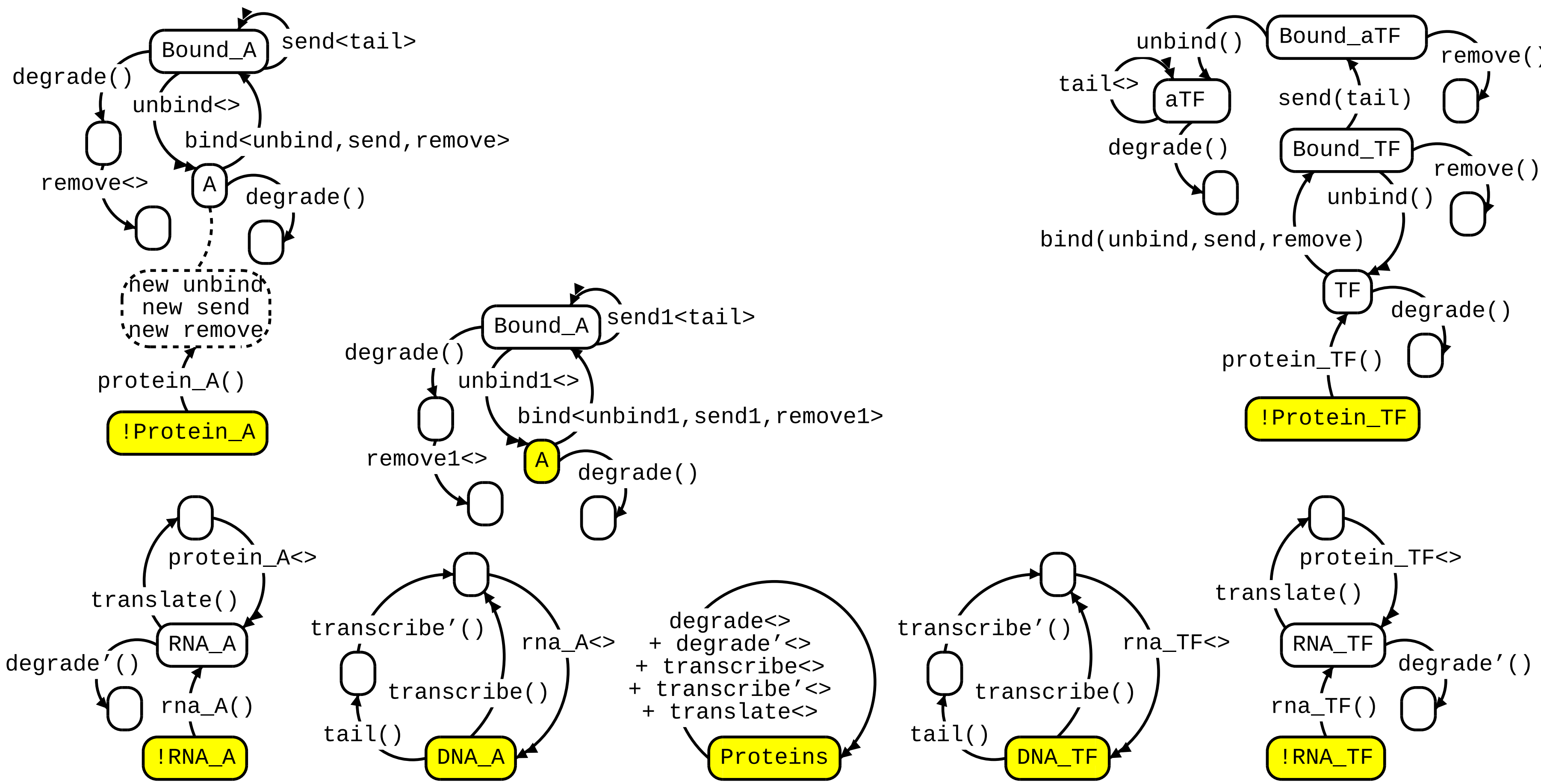
# Gene Regulation by Positive Feedback



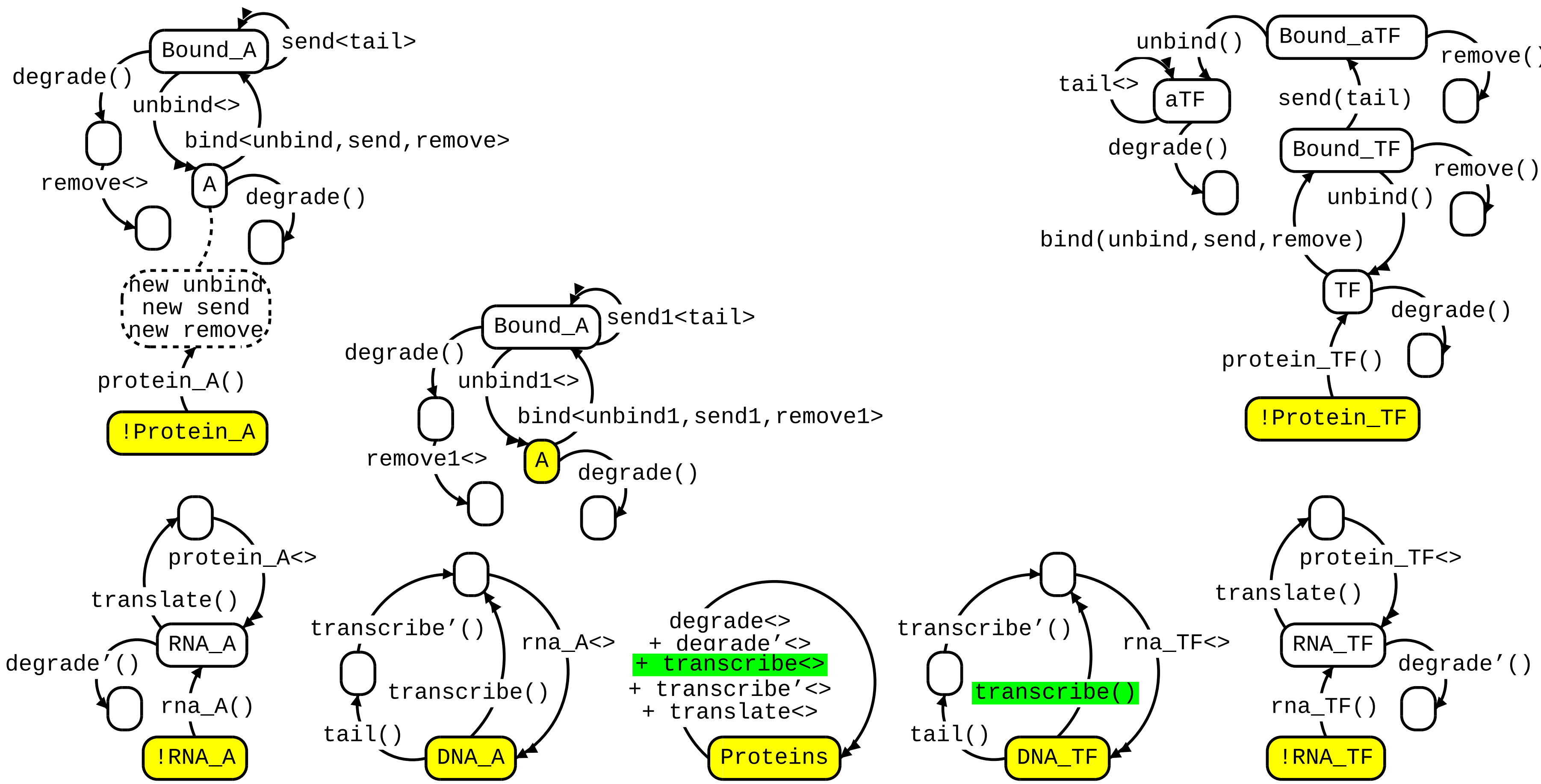
# Gene Regulation by Positive Feedback



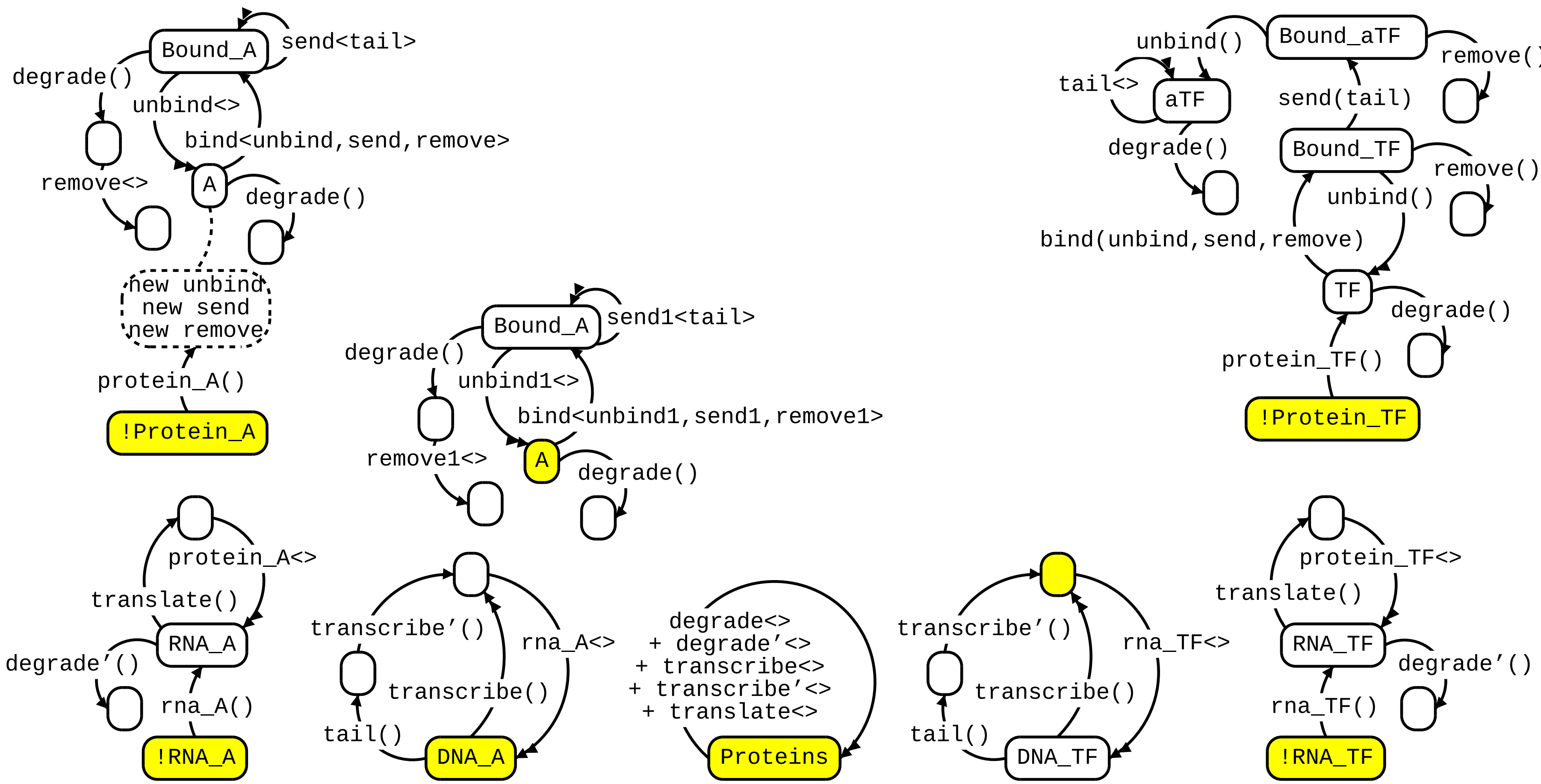
# Gene Regulation by Positive Feedback



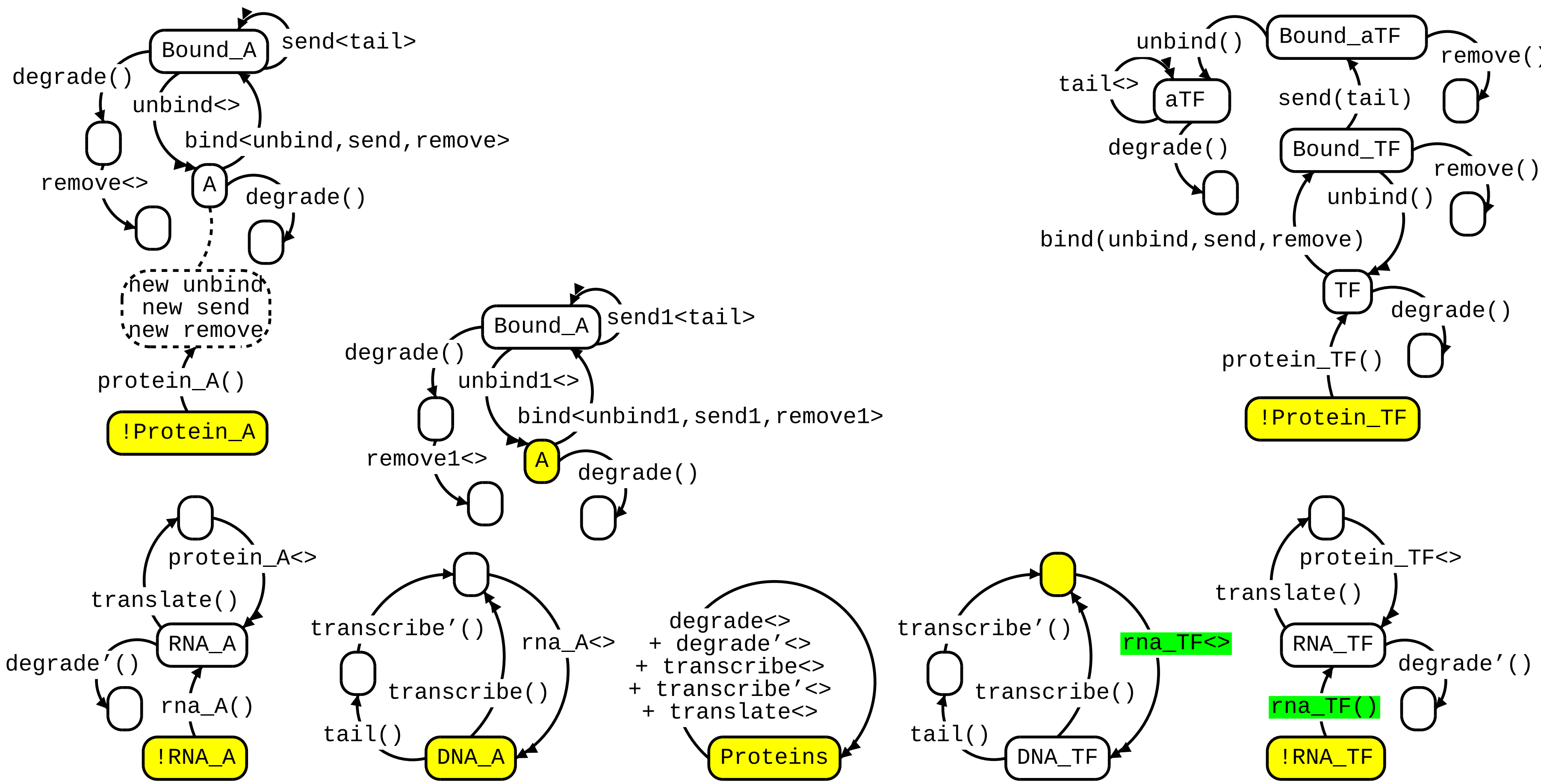
# Gene Regulation by Positive Feedback



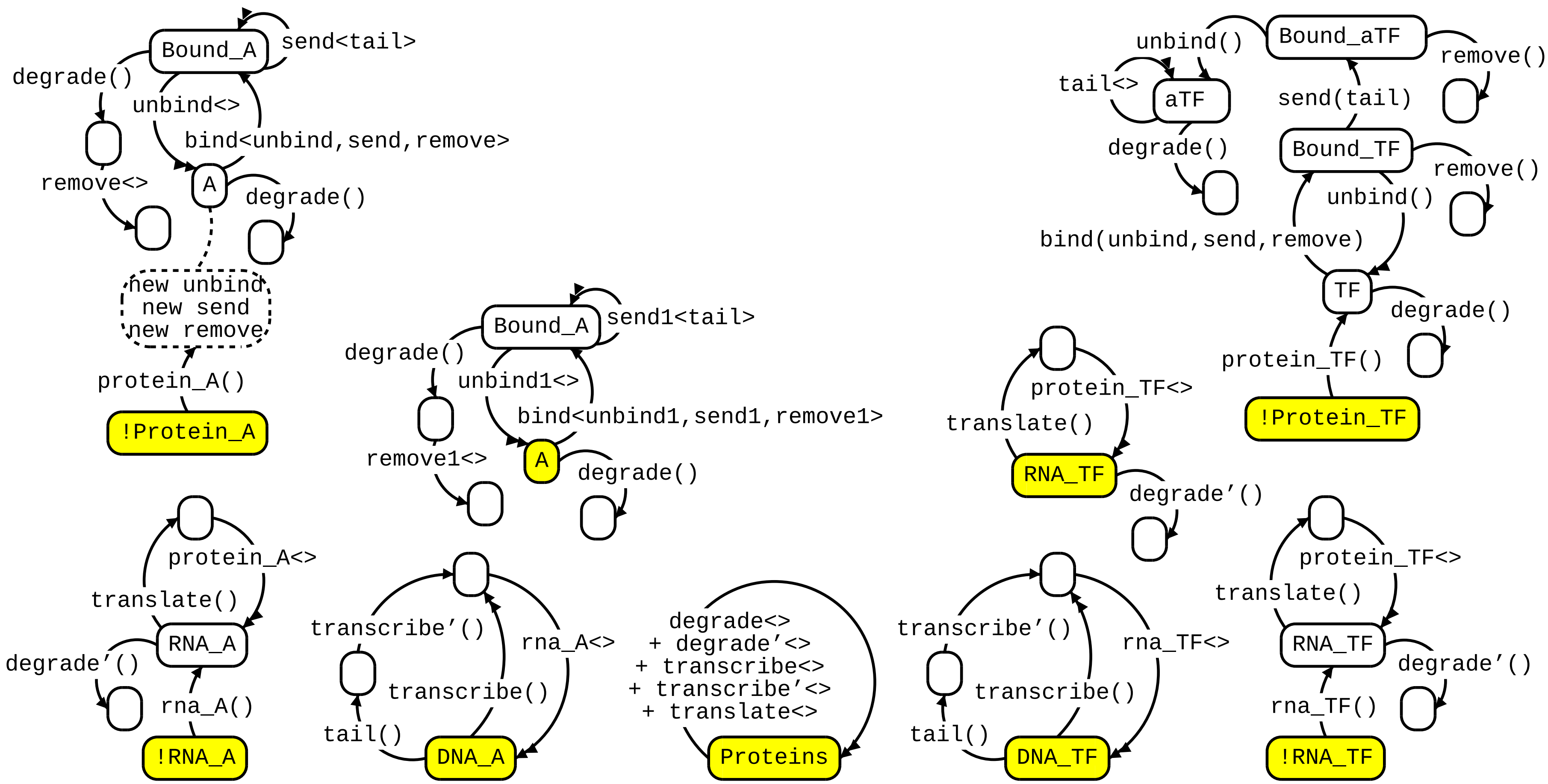
# Gene Regulation by Positive Feedback



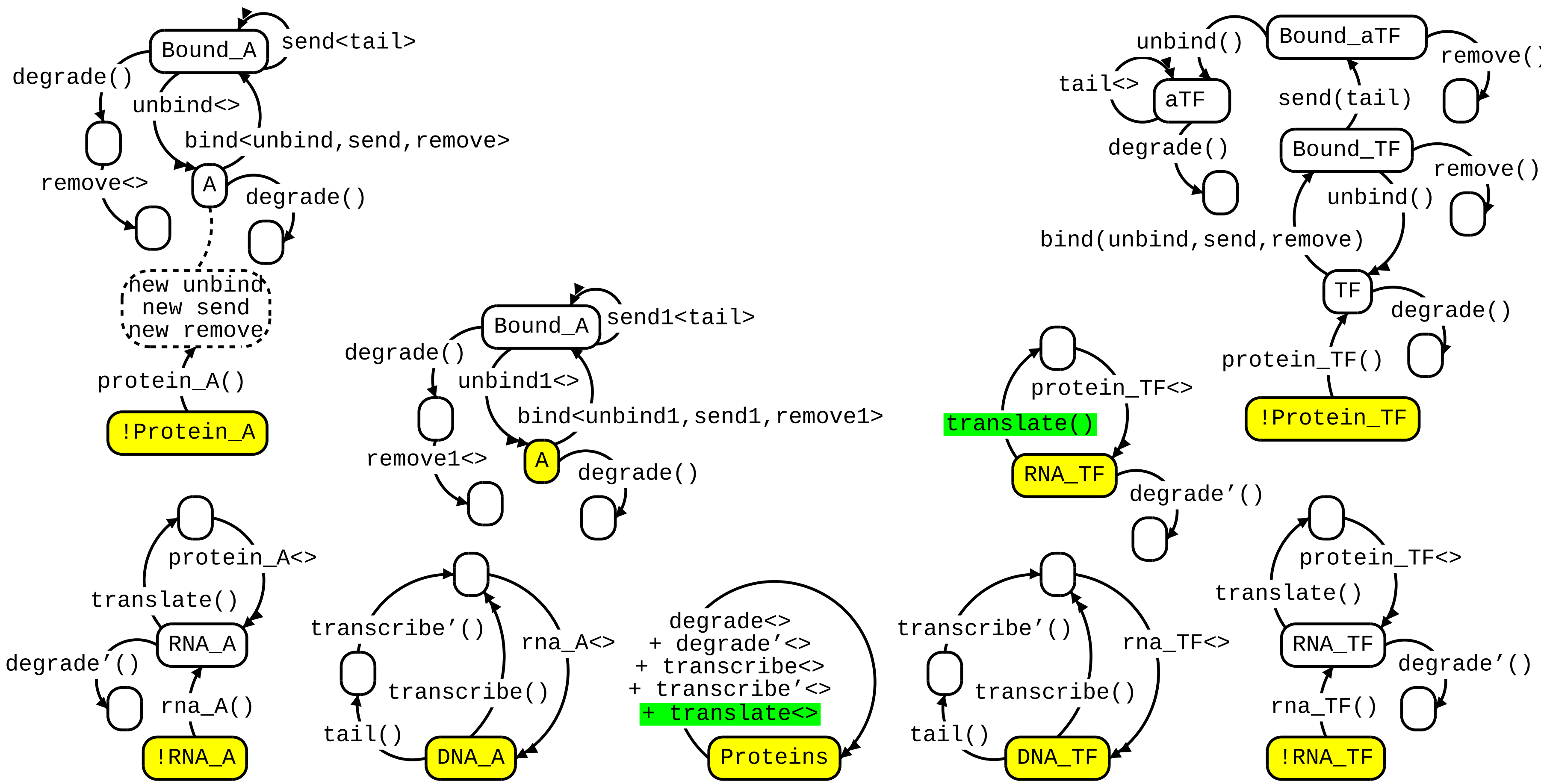
# Gene Regulation by Positive Feedback



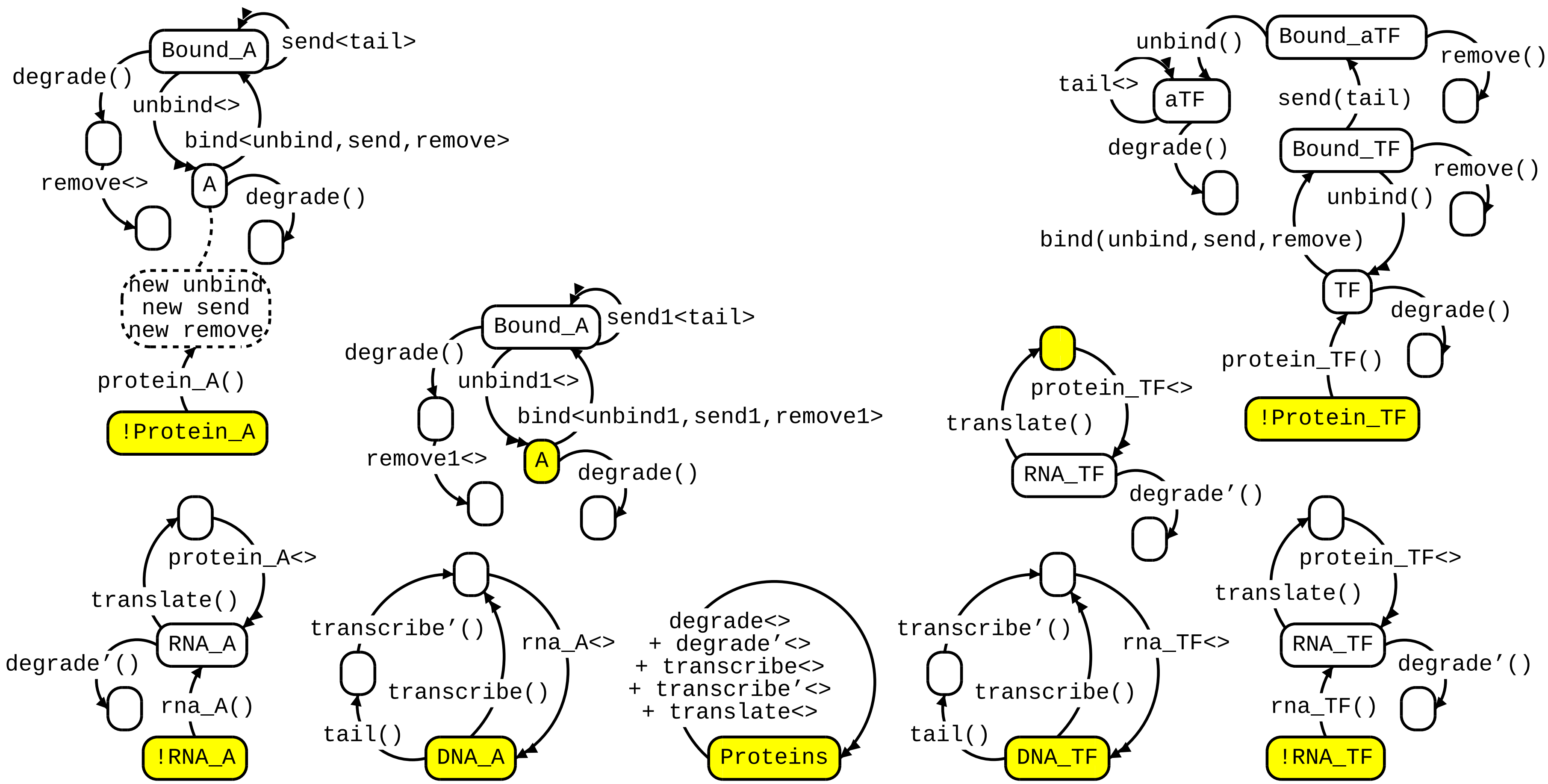
# Gene Regulation by Positive Feedback



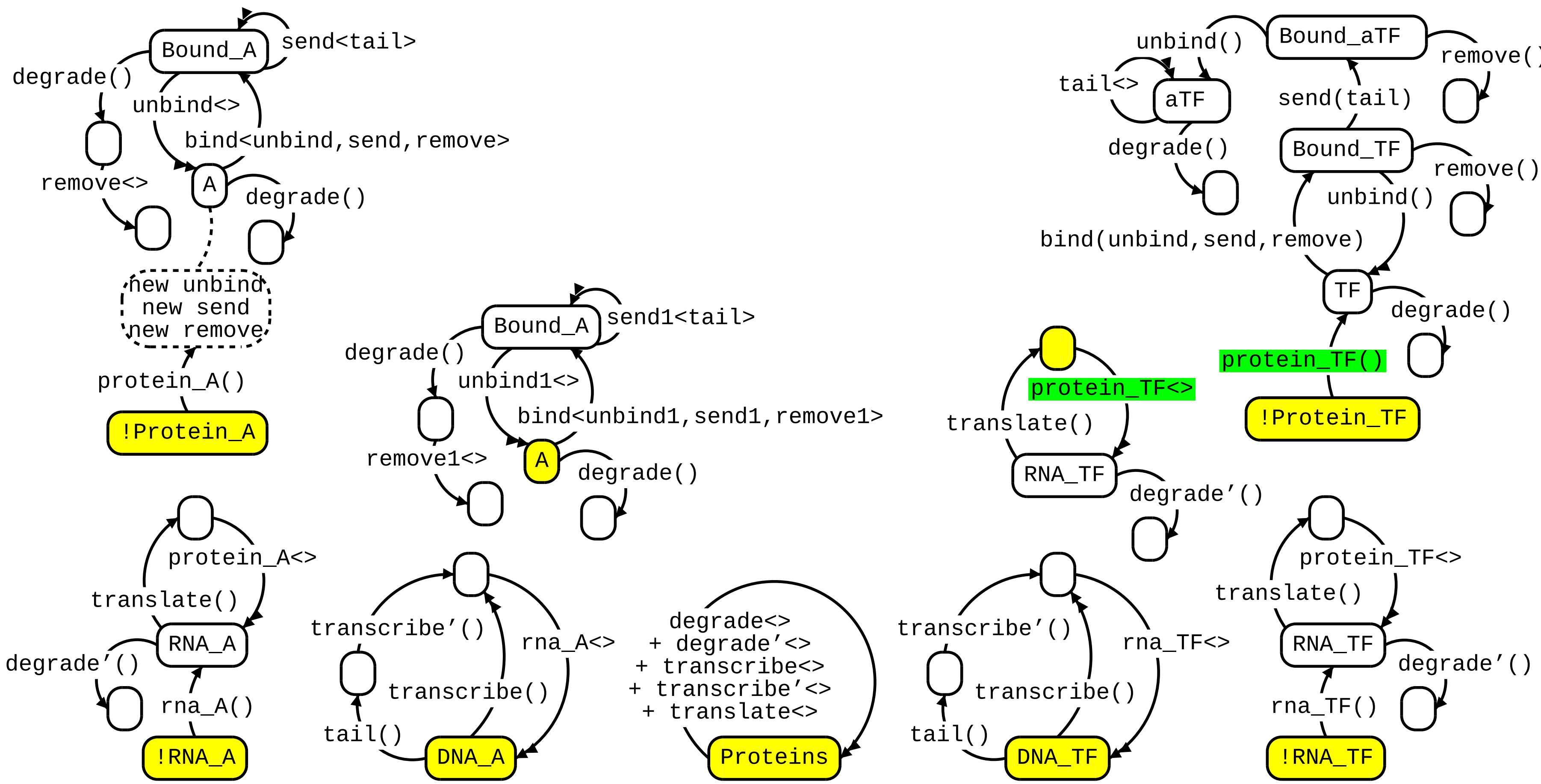
# Gene Regulation by Positive Feedback



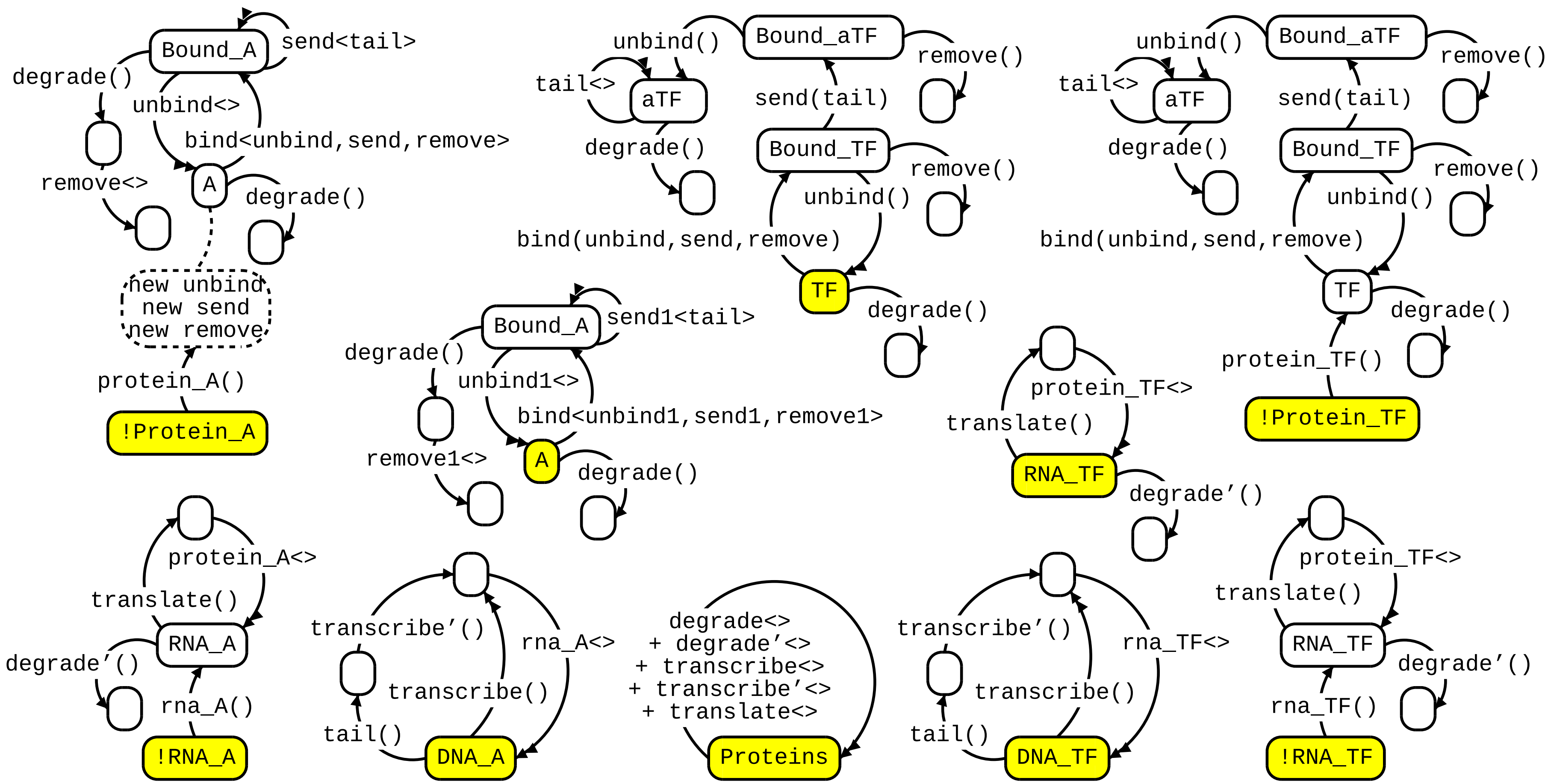
# Gene Regulation by Positive Feedback



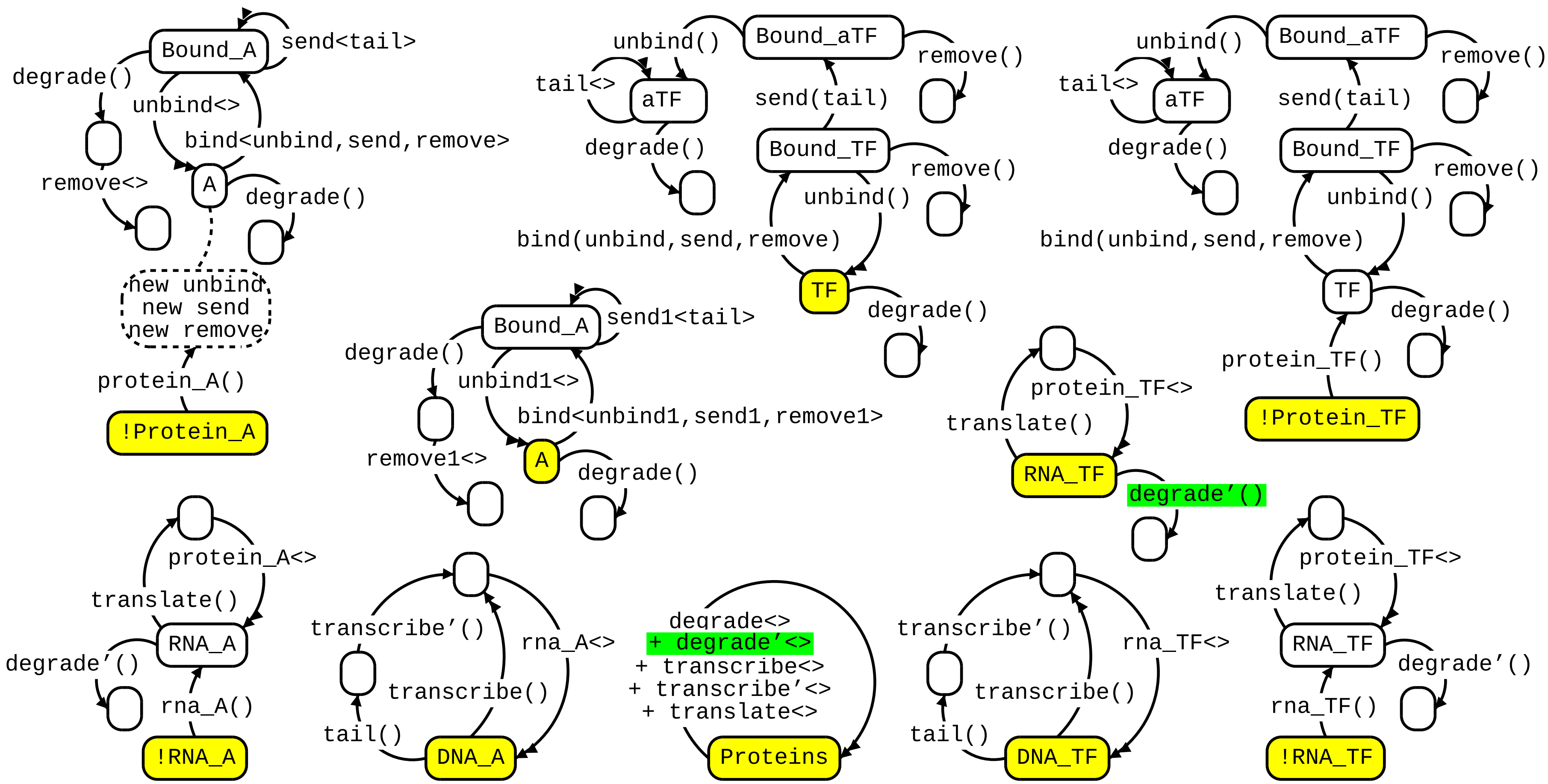
# Gene Regulation by Positive Feedback



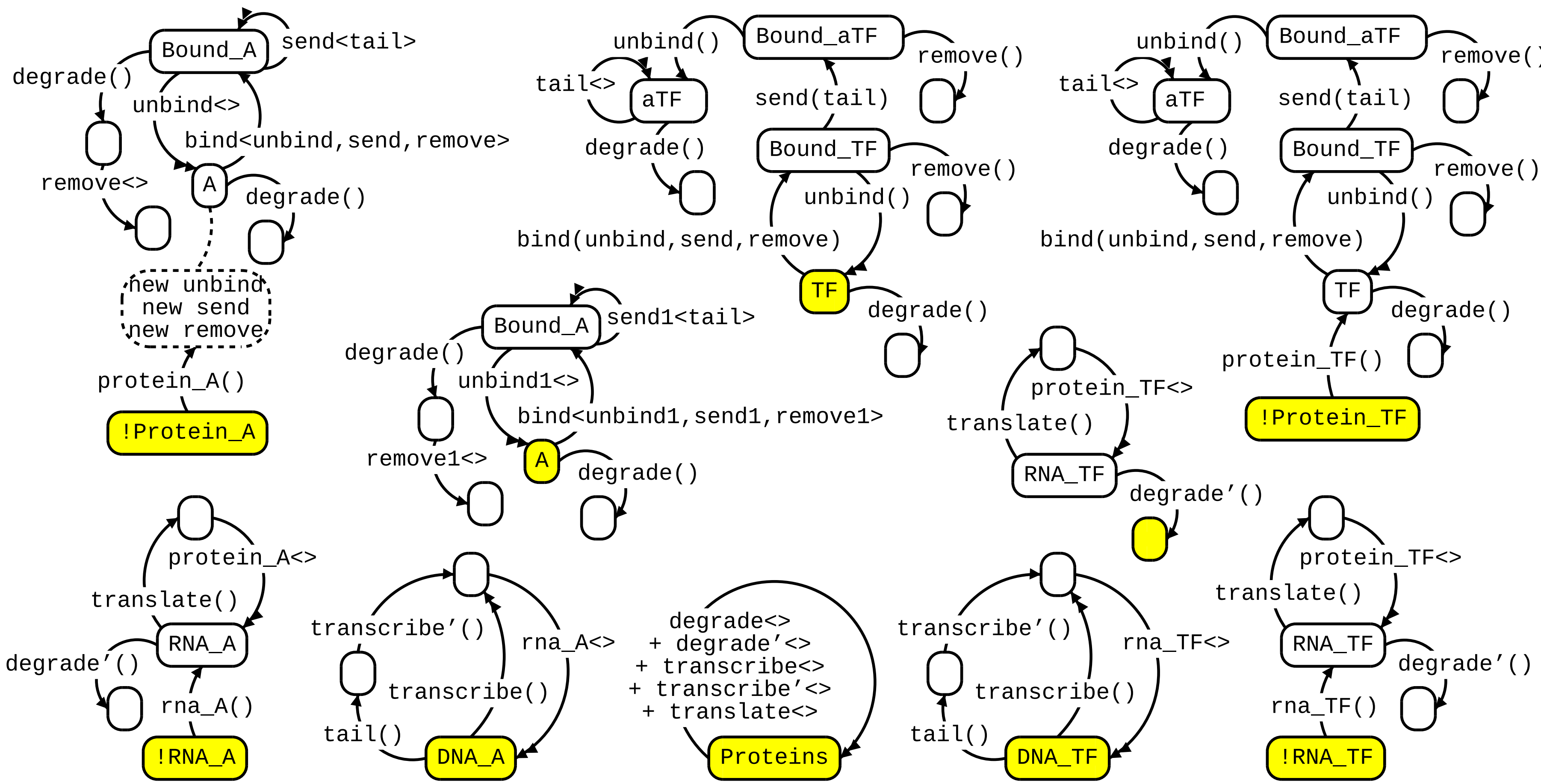
# Gene Regulation by Positive Feedback



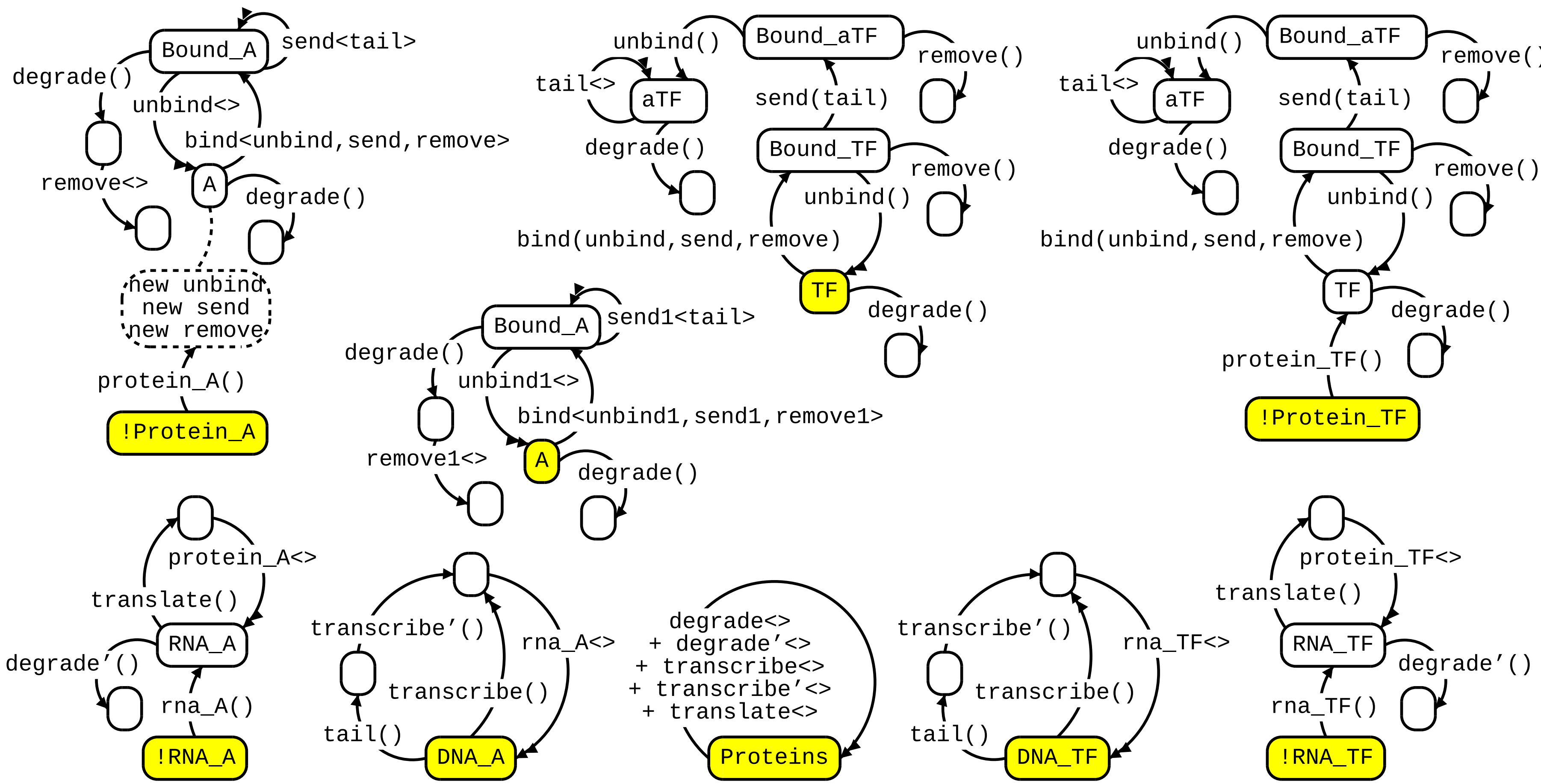
# Gene Regulation by Positive Feedback



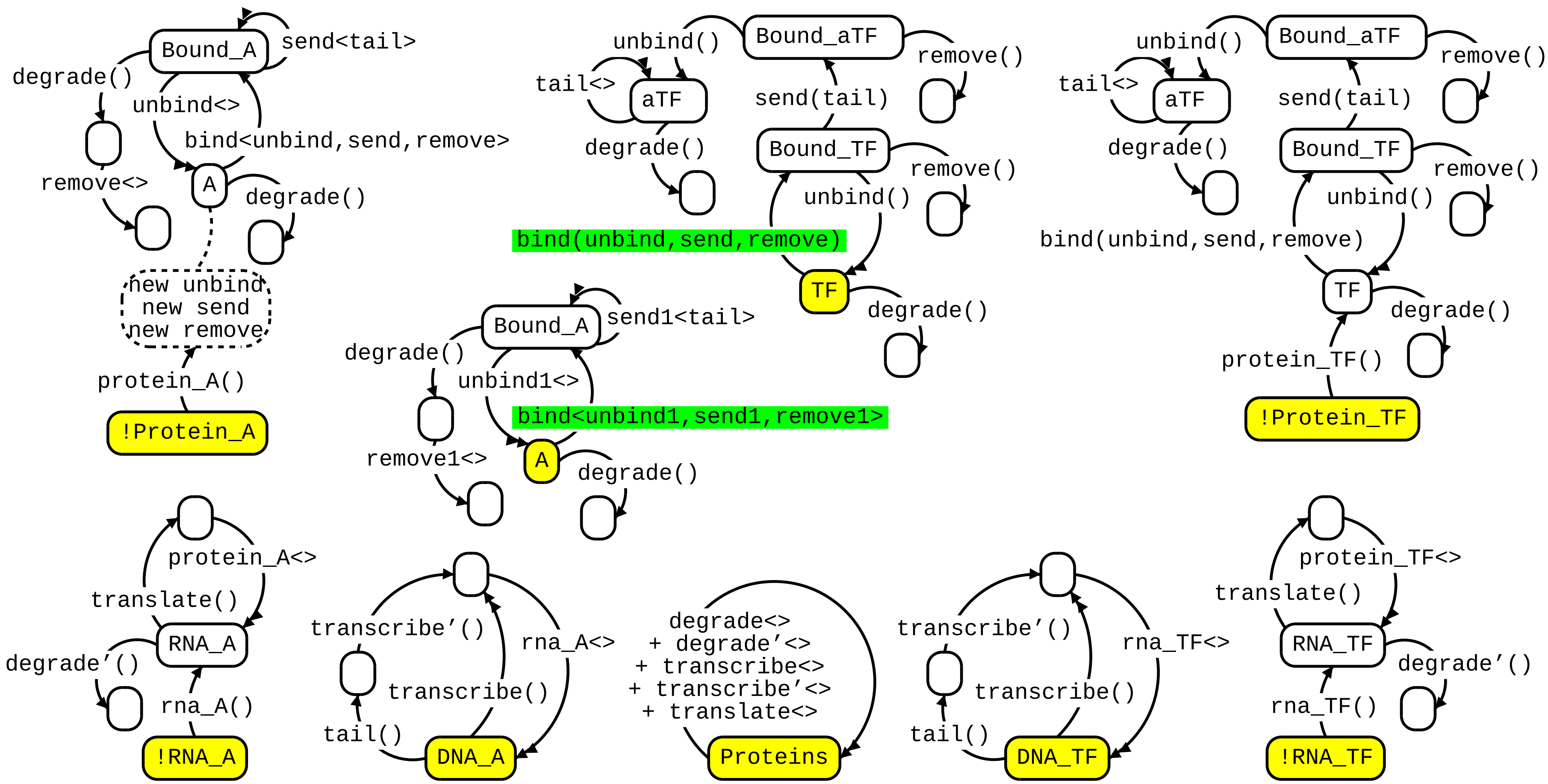
# Gene Regulation by Positive Feedback



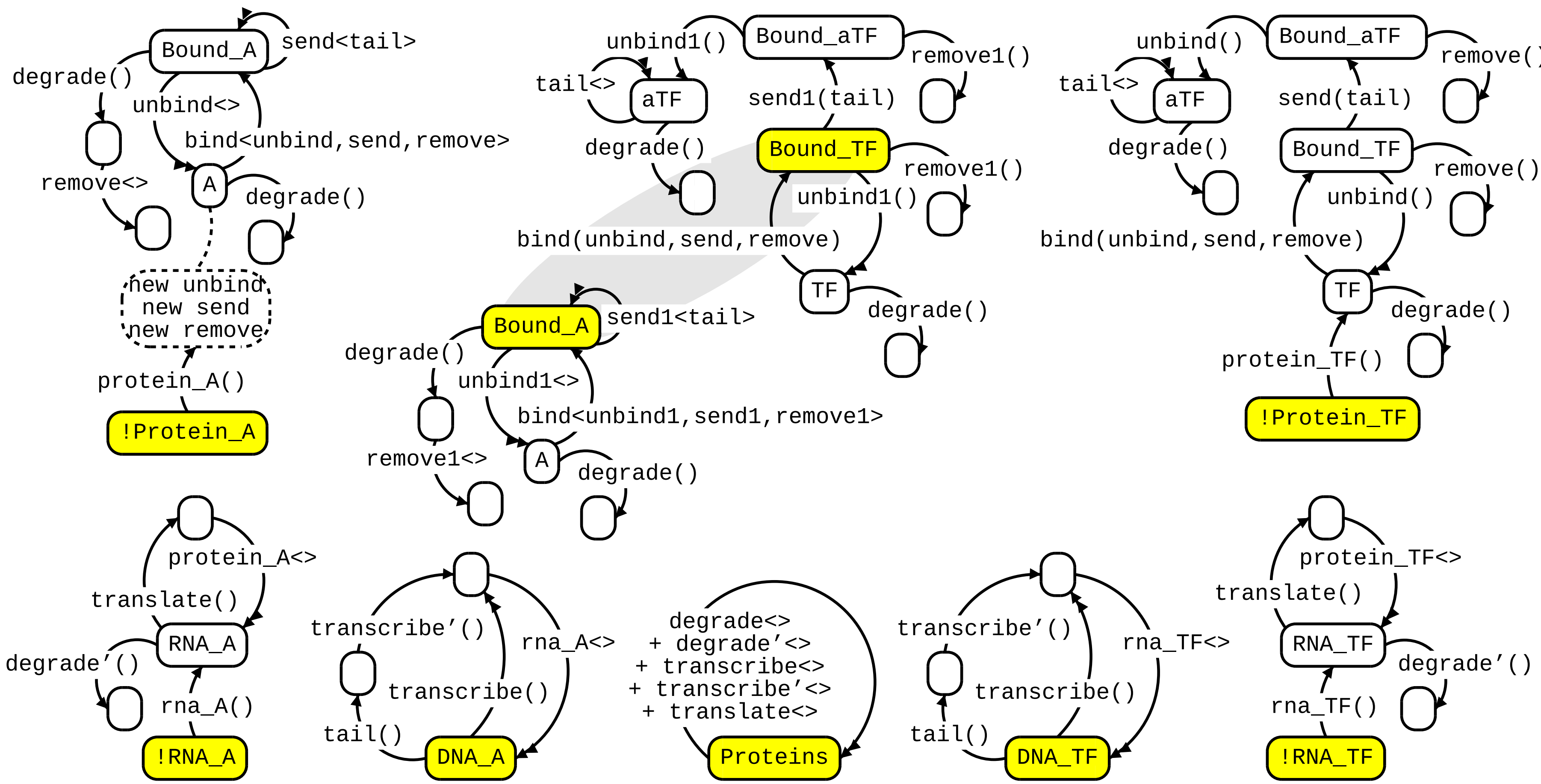
# Gene Regulation by Positive Feedback



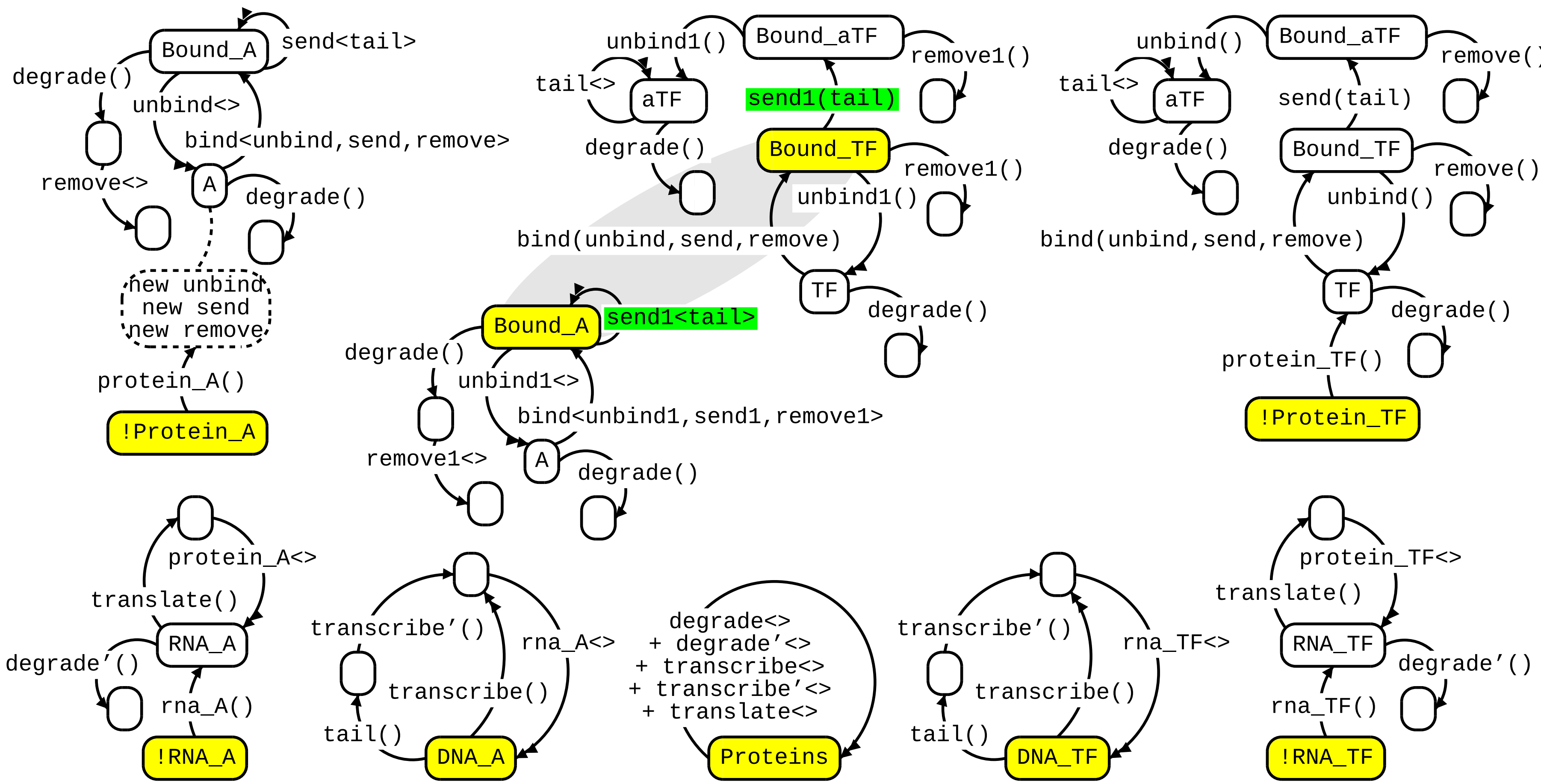
# Gene Regulation by Positive Feedback



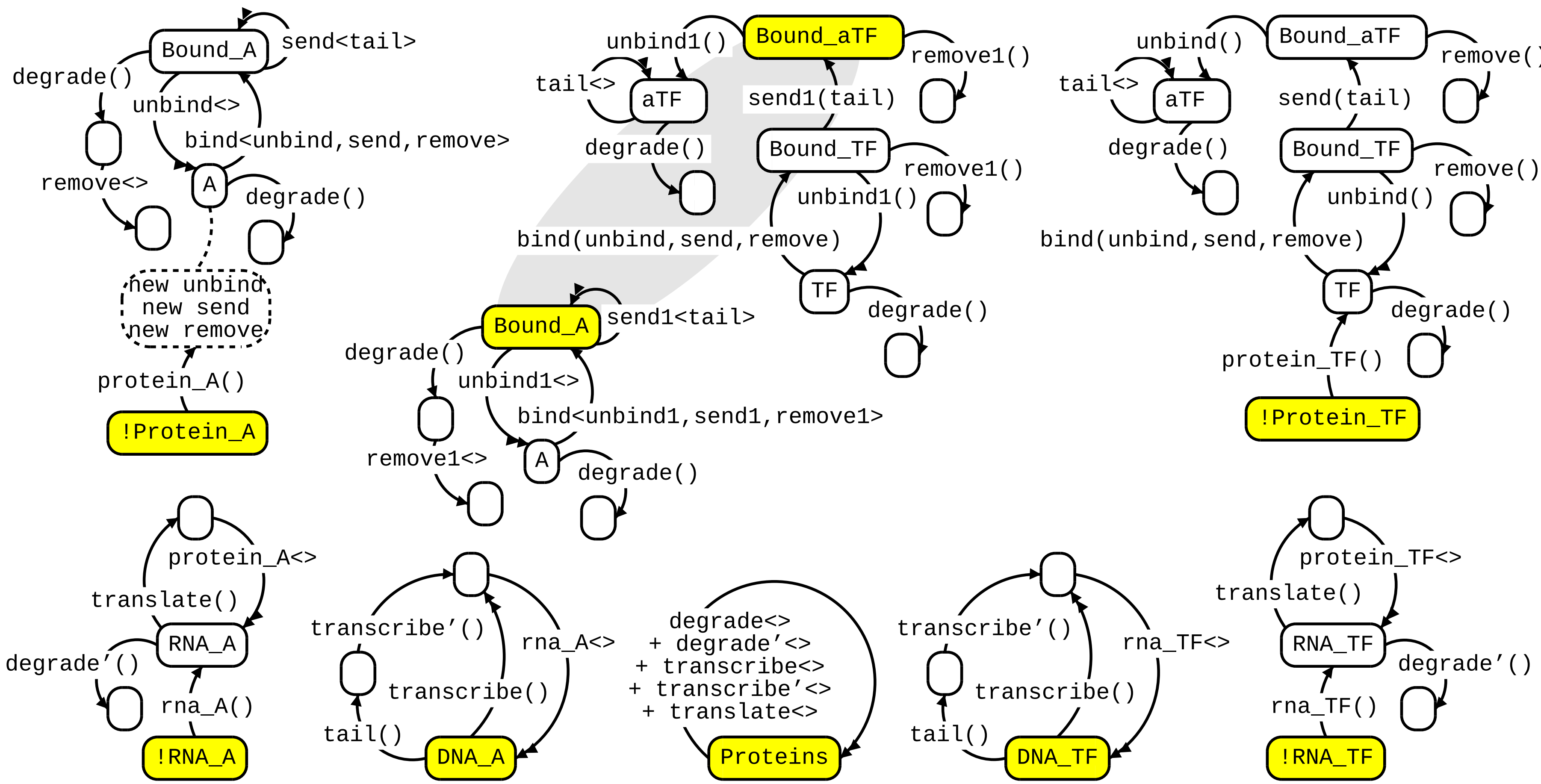
# Gene Regulation by Positive Feedback



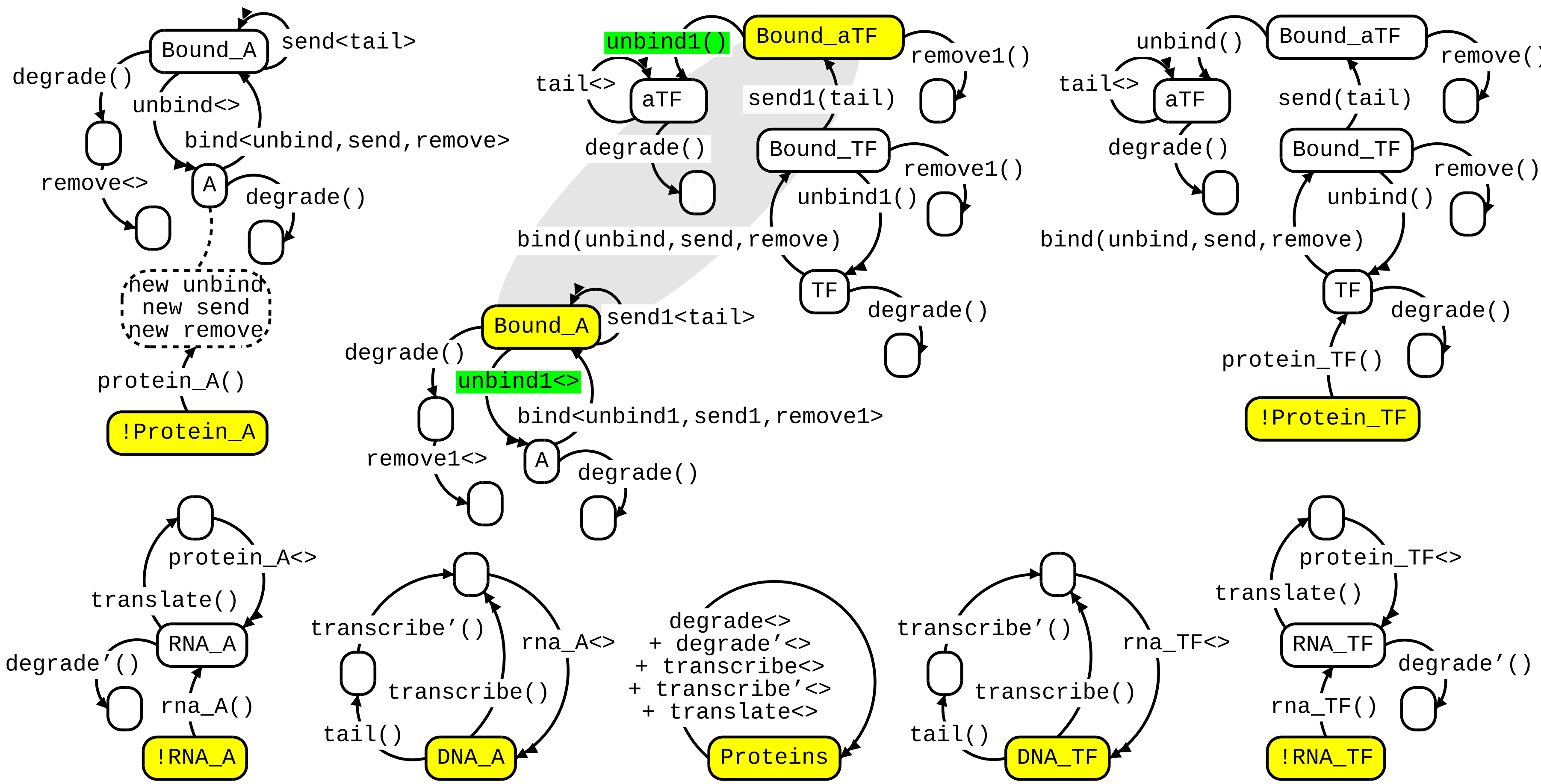
# Gene Regulation by Positive Feedback



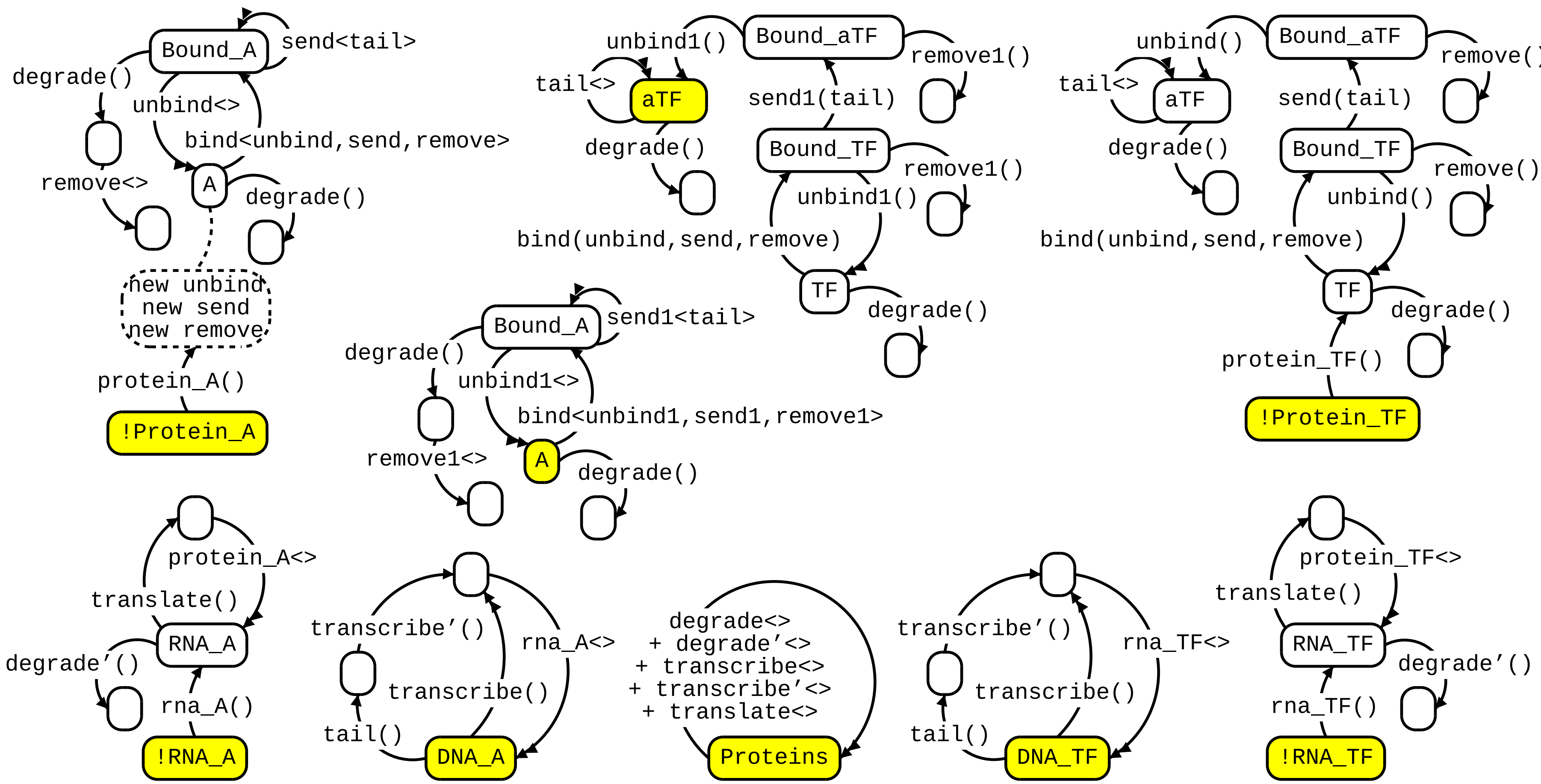
# Gene Regulation by Positive Feedback



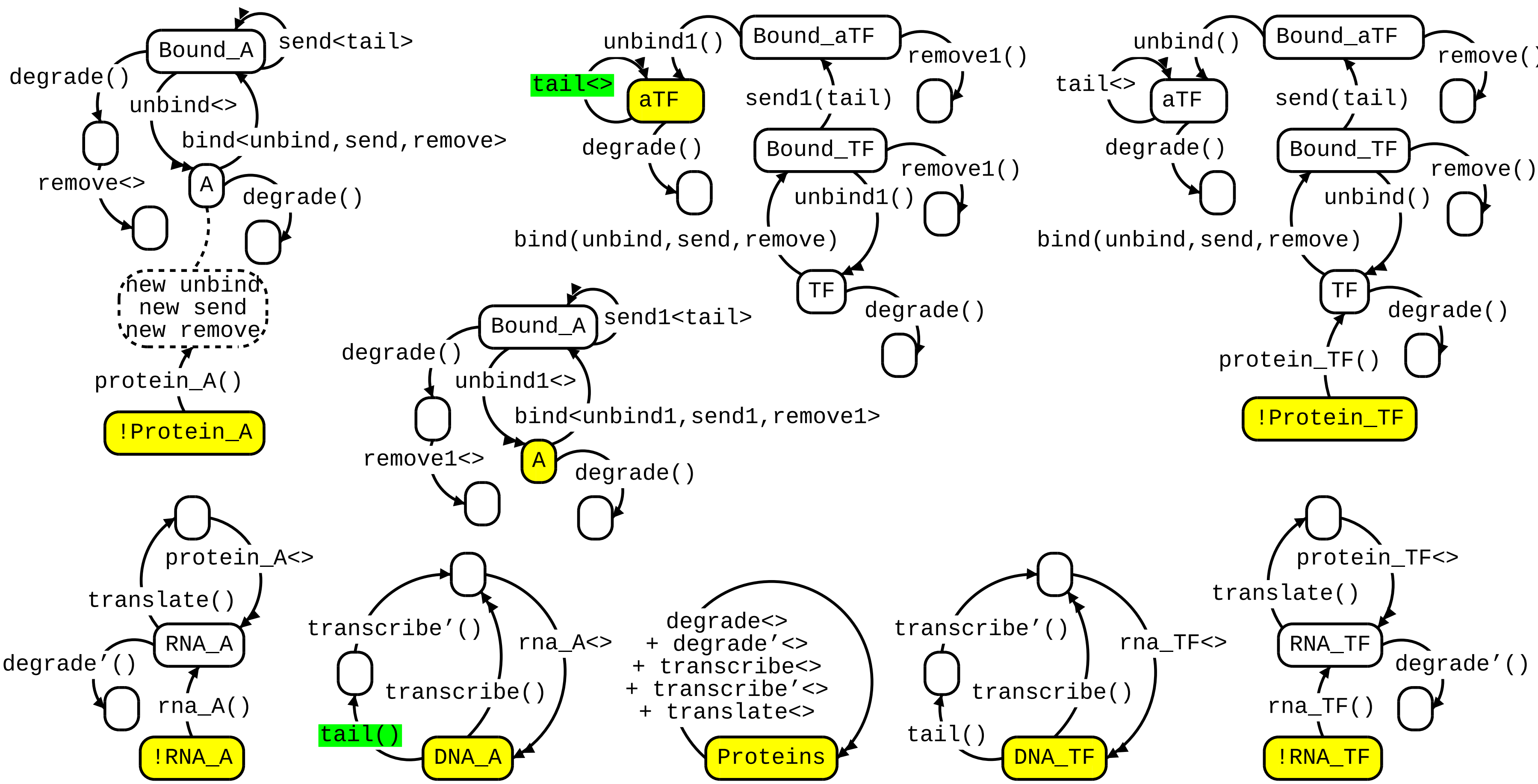
# Gene Regulation by Positive Feedback



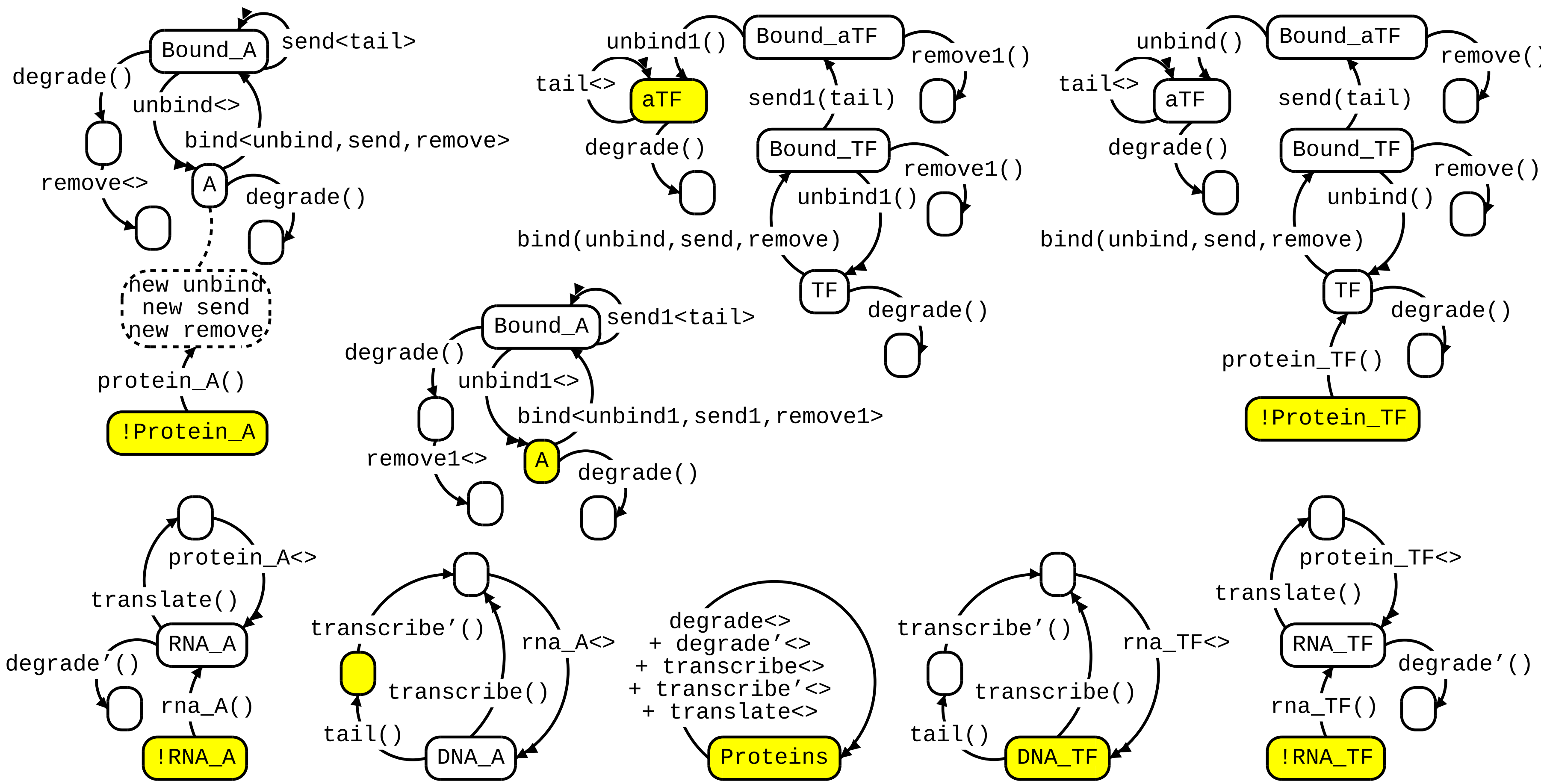
# Gene Regulation by Positive Feedback



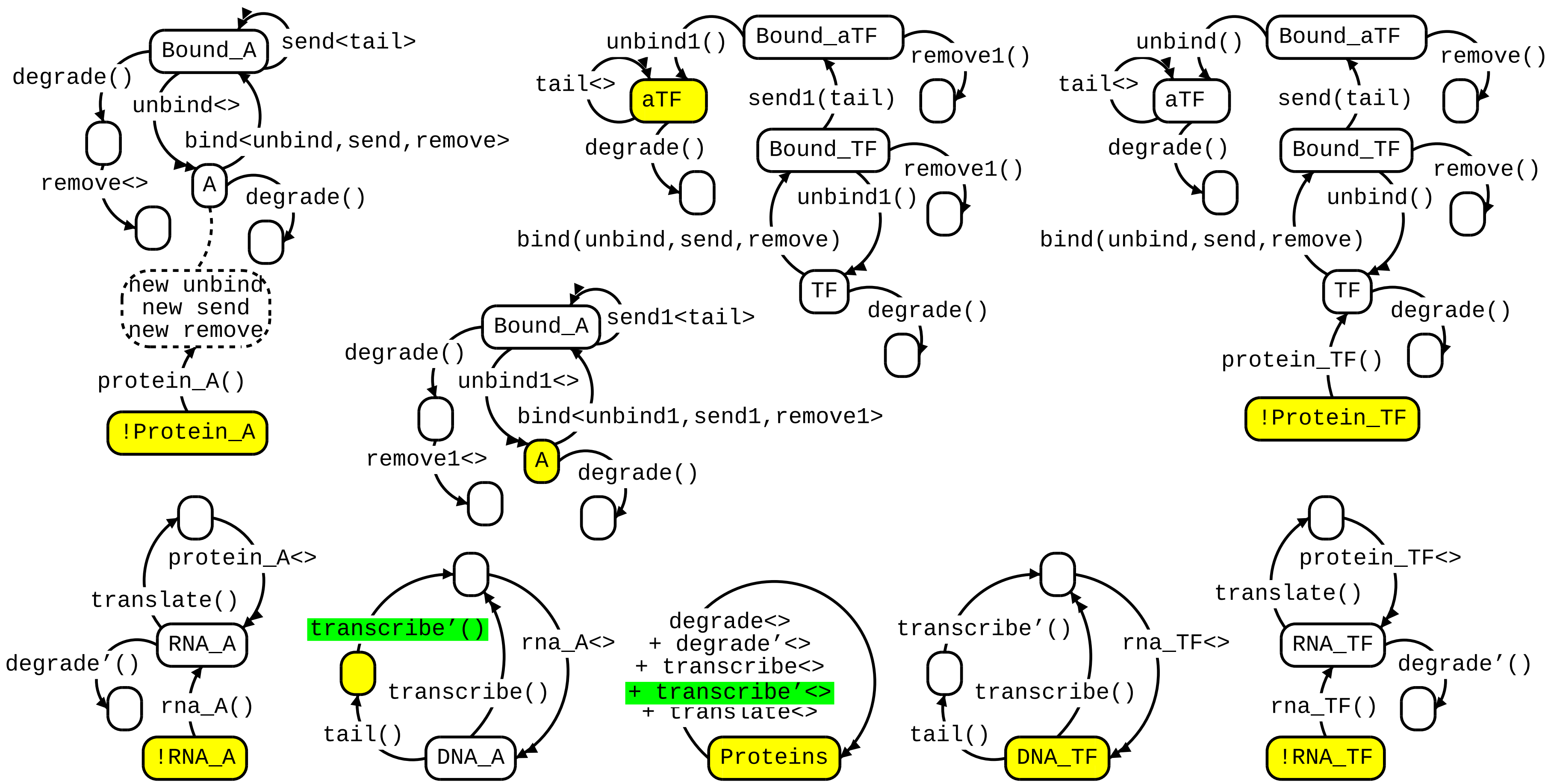
# Gene Regulation by Positive Feedback



# Gene Regulation by Positive Feedback



# Gene Regulation by Positive Feedback



# Abstract Machine for Stochastic Pi-Calculus (SPiM)

- Formalise how the simulator works (program specification).
- Prove properties about the simulator (program verification).
- Give greater confidence in the simulation results.
- Improve on existing simulators (BioSpi).

# Machine Data Structures

➢ General Machine Term:

$$\nu n_1 \, \nu n_2 \, ... \nu n_N \, (\Sigma_1 :: \Sigma_2 :: \ldots :: \Sigma_M :: [\,])$$

➢ Syntax Definition:

$$\begin{array}{rcll} V, U & ::= & \nu n \, V & \text{Restriction} \\ & \mid & A & \text{List} \\ A, B & ::= & [\,] & \text{Empty} \\ & \mid & \Sigma :: A & \text{Summation} \end{array}$$

# Machine Encoding

➢ Encoding $\llbracket P \rrbracket$:

$$\llbracket P \rrbracket \triangleq P \circ []$$

➢ Construction $(P \circ V)$:

$$\begin{array}{rrcl}
n \notin fn(P) \Rightarrow & P \circ (\nu n\, V) & \triangleq & \nu n\,(P \circ V) \\
& \mathbf{0} \circ A & \triangleq & A \\
& (P \mid Q) \circ A & \triangleq & P \circ Q \circ A \\
n \notin fn(P \circ A) \Rightarrow & (\nu m\, P) \circ A & \triangleq & \nu n\,(P_{\{n/m\}} \circ A) \\
& !\pi.P \circ A & \triangleq & (\pi.(P \mid !\pi.P) + \mathbf{0}) \circ A \\
& (\pi.P + \Sigma) \circ A & \triangleq & (\pi.P + \Sigma) :: A
\end{array}$$

# Machine Execution

- Reduction ($V \xrightarrow{r} V'$):

$$V \xrightarrow{r} V' \quad \Rightarrow \quad \nu x\, V \quad \xrightarrow{r} \quad \nu x\, V'$$

$$\left| \begin{array}{l} x = Next(A) \\ \wedge A \succ (x(m).P + \Sigma) :: A' \\ \wedge A' \succ (x\langle n \rangle.Q + \Sigma') :: A'' \end{array} \right. \quad \Rightarrow \quad A \quad \xrightarrow{rate(x)} \quad P_{\{n/m\}} \circ Q \circ A''$$

- Selection ($A \succ B$):

$$A \quad \succ \quad A$$

$$A \succ \Sigma' :: A' \Rightarrow \Sigma :: A \quad \succ \quad \Sigma' :: \Sigma :: A'$$

$$\Sigma :: A \succ (\pi'.P' + \Sigma') :: A \Rightarrow (\pi.P + \Sigma) :: A \quad \succ \quad (\pi'.P' + \pi.P + \Sigma') :: A$$

# Channel Activity

- Choose next channel $x = Next(A)$ by stochastic algorithm [Gillespie, 1977]
- Gillespie chooses next channel based on *activity*:

$$\text{activity} = \text{number of possible interactions on a channel}$$

- In SPiM, activity of channel $x$ in term $A$:

$$\text{Act}_x(A) = (\text{In}_x(A) * \text{Out}_x(A)) - \text{Mix}_x(A)$$

  - $\text{In}_x(A)$ = number of unguarded *inputs* on channel $x$ in $A$.
  - $\text{Out}_x(A)$ = number of unguarded *outputs* on channel $x$ in $A$.
  - $\text{Mix}_x(A)$ = sum of $\text{In}_x(\Sigma_i) \times \text{Out}_x(\Sigma_i)$ for each summation $\Sigma_i$ in $A$.

## Gillespie: Choosing the Next Reaction $Next(A)$

1. For all $x \in fn(A)$ calculate $a_x = \mathrm{Act}_x(A) * rate(x)$

2. Store non-zero values of $a_x$ in a list $(x_\mu, a_\mu)$, where $\mu \in 1...M$.

3. Calculate $a_0 = \sum_{\nu=0}^{M} a_\nu$

4. Generate two random numbers $n_{1,} n_2 \in [0, 1]$ and calculate $\tau, \mu$ such that:

$$\tau = (1/a_0) \ln(1/n_1)$$

$$\sum_{\nu=1}^{\mu-1} a_\nu < n_2 a_0 \leq \sum_{\nu=1}^{\mu} a_\nu$$

5. $Next(A) = x_\mu$ and $Delay(A) = \tau$.

# Correctness of the Machine

➢ *Safety*: no runtime errors (no crashes)

**Lemma 1.** $\forall V. V \in \mathrm{SPiM} \wedge V \xrightarrow{r} V' \Rightarrow V' \in \mathrm{SPiM}$

➢ *Soundness*: machine only performs valid executions steps (behaves well)

**Theorem 1.** $\forall V. V \in \mathrm{SPiM} \wedge V \xrightarrow{r} V' \Rightarrow [\![V]\!] \xrightarrow{r} [\![V']\!]$

➢ *Completeness*: machine can perform all execution steps of the calculus

**Theorem 2.** $\forall P. P \in \mathrm{SPi} \wedge P \xrightarrow{r} P' \Rightarrow (\![P]\!) \xrightarrow{r}\equiv (\![P']\!)$.

➢ *Termination*: machine does not loop forever unnecessarily

**Theorem 3.** $\forall P. P \in \mathrm{SPi} \wedge P \not\longrightarrow \;\Rightarrow\; (\![P]\!) \not\longrightarrow$

# Correctness of the Machine

➢ *Duration*: reactions in machine and calculus have same average duration

- Gillespie algorithm proved correct for selecting *next* reaction channel.
- Also need to ensure that reaction has correct *duration*
- E.g. reduction in $P_1$ is twice as fast as reduction in $P_2$:

$$P_1 \triangleq x^r\langle n\rangle.P + x^r\langle n\rangle.P \mid x^r(m).Q$$

$$P_2 \triangleq x^r\langle n\rangle.P \mid x^r(m).Q$$

- Sufficient to ensure that machine and calculus have same channel activity.

**Proposition 1.** $\forall V \in \text{SPiM}.\text{Act}_x(V) = \text{Act}_x([\![V]\!])$

**Proposition 2.** $\forall P \in \text{SPi}.\text{Act}_x(P) = \text{Act}_x((\![P]\!))$

# Implementation

- Abstract Machine maps almost directly to program code
- Implemented a polymorphic type system and type checker
- Correctness of the machine gives greater confidence in the simulation results
- Demo...

# Conclusion

- Presented a graphical representation for pi-calculus:
  - Precise, compositional, executable descriptions.
  - Used to model regulatory systems at the molecular level.
- Presented an abstract machine for the stochastic pi-calculus:
  - Correctness proof: safety, soundness, completeness, termination, duration.
  - Maps readily to program code.
  - Could be used as a basis for implementing new calculi.
- Built a simulator based on the machine.
  - Plan to incorporate a graphical editor as a front-end.

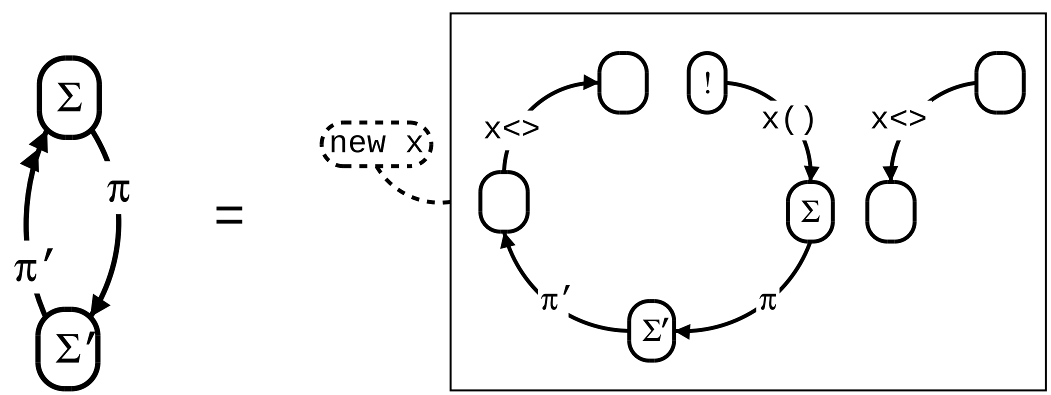
# References

[Gillespie, 1977] Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361.

[Priami et al., 2001] Priami, C., Regev, A., Shapiro, E., and Silverman, W. (2001). Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*. in press.

# Link Encoding

- Encoding uses restriction, replication, parallel composition and communication.
- A linked node $\longrightarrow$ a replicated input on a fresh channel $x$, in parallel with an output on $x$
- A link to the node $\longrightarrow$ an output on $x$.
- E.g.:



# Safety Proof

**Lemma 2.** $\forall V.V \in \mathrm{SPiM} \wedge V \xrightarrow{r} V' \Rightarrow V' \in \mathrm{SPiM}$

**Proof.** By Lemma 3, Lemma 4 and by induction on reduction in SPiM. □

**Lemma 3.** $\forall A \in \mathrm{SPiM}.A \succ B \Rightarrow B \in \mathrm{SPiM}$

**Proof.** By induction on selection in SPiM. □

**Lemma 4.** $\forall V.\forall P.V \in \mathrm{SPiM} \wedge P \in \mathrm{SPi} \Rightarrow P \circ V \in \mathrm{SPiM}$

**Proof.** By induction on construction in SPiM. □

# Soundness Proof

**Theorem 4.** $\forall V. V \in \mathrm{SPiM} \wedge V \xrightarrow{r} V' \Rightarrow [\![V]\!] \xrightarrow{r} [\![V']\!]$

**Proof.** By Lemma 5, Lemma 6 and by induction on reduction in $\mathrm{SPiM}$. □

**Lemma 5.** $\forall A. A \in \mathrm{SPiM} \wedge A \succ B \Rightarrow [\![A]\!] \equiv [\![B]\!]$

**Proof.** By induction on selection in $\mathrm{SPiM}$. □

**Lemma 6.** $\forall V. \forall P. V \in \mathrm{SPiM} \wedge P \in \mathrm{SPi} \Rightarrow [\![P \circ V]\!] \equiv P \mid [\![V]\!]$

**Proof.** By induction on construction in $\mathrm{SPiM}$. □

$$
\begin{array}{rcl}
[\![\nu n\, V]\!] & \triangleq & \nu n\, [\![V]\!] \\
[\![ [] ]\!] & \triangleq & \mathbf{0} \\
[\![\Sigma :: A]\!] & \triangleq & \Sigma \mid [\![A]\!]
\end{array}
$$

# Completeness Proof

**Theorem 5.** $\forall P.P \in \mathrm{SPi} \wedge P \xrightarrow{r} P' \Rightarrow [\![P]\!] \xrightarrow{r}\equiv [\![P']\!]$.

**Proof.** By Lemma 7 and by induction on reduction in $\mathrm{SPi}$, where the rule for parallel composition is expanded over the remaining rules. $\Box$

**Lemma 7.** $P \equiv Q \Rightarrow [\![P]\!] \equiv [\![Q]\!]$

**Proof.** By induction on structural congruence in $\mathrm{SPi}$. $\Box$

**Lemma 8.** $\forall V.V \in \mathrm{SPiM} \wedge U \equiv V \wedge V \xrightarrow{r} V' \Rightarrow \exists U'.U \xrightarrow{r} U' \wedge U' \equiv V'$

**Proof.** By induction on structural congruence in $\mathrm{SPiM}$. $\Box$

# Structural Congruence

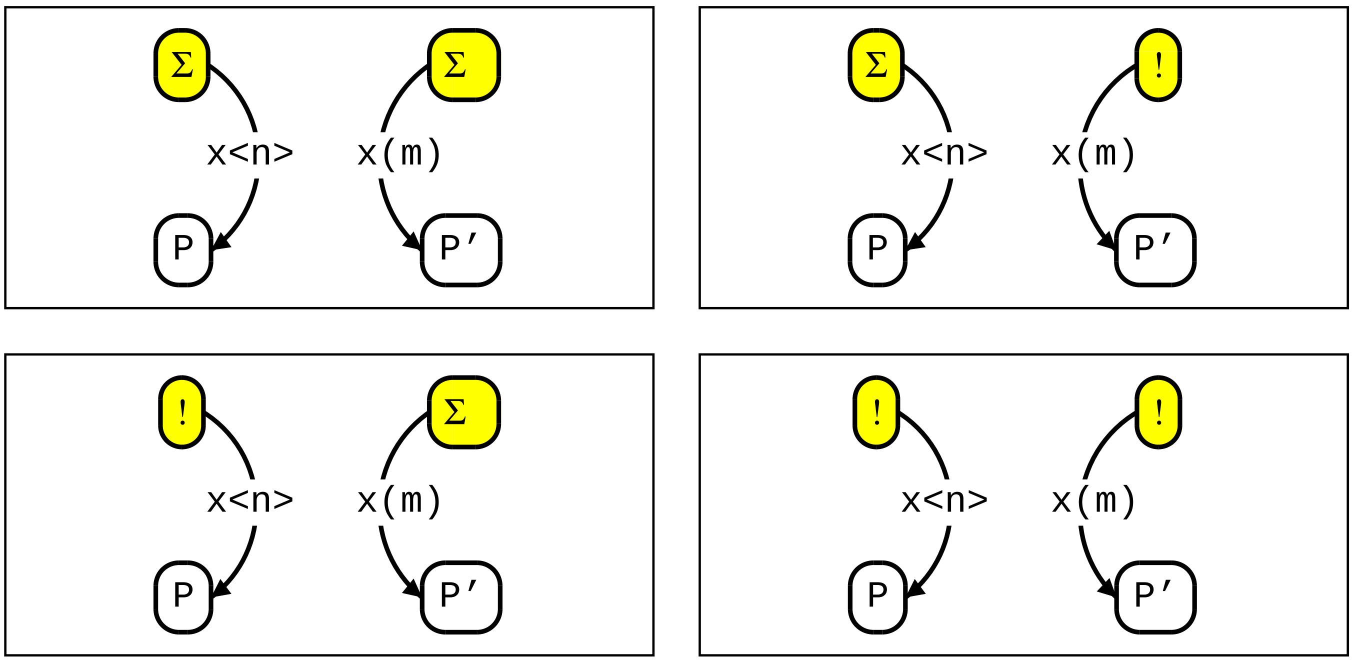
$$
\begin{array}{rcl}
V \equiv_\alpha U \Rightarrow V & \equiv & U \\
x \notin fn(V) \quad \Rightarrow \quad \nu x\, V & \equiv & V \\
\nu x\, \nu y\, V & \equiv & \nu y\, \nu x\, V \\
\Sigma :: \Sigma' :: A & \equiv & \Sigma' :: \Sigma :: A \\
A \equiv A' \quad \Rightarrow \quad \Sigma :: A & \equiv & \Sigma :: A' \\
(\pi.P + \pi'.P' + \Sigma) :: A & \equiv & (\pi'.P' + \pi.P + \Sigma) :: A \\
\Sigma :: A \equiv \Sigma' :: A \Rightarrow (\pi.P + \Sigma) :: A & \equiv & (\pi.P + \Sigma') :: A
\end{array}
$$

# Termination Proof

**Theorem 6.** $\forall P.P \in \mathrm{SPi} \wedge P \not\longrightarrow \;\Rightarrow\; [\![P]\!] \not\longrightarrow$

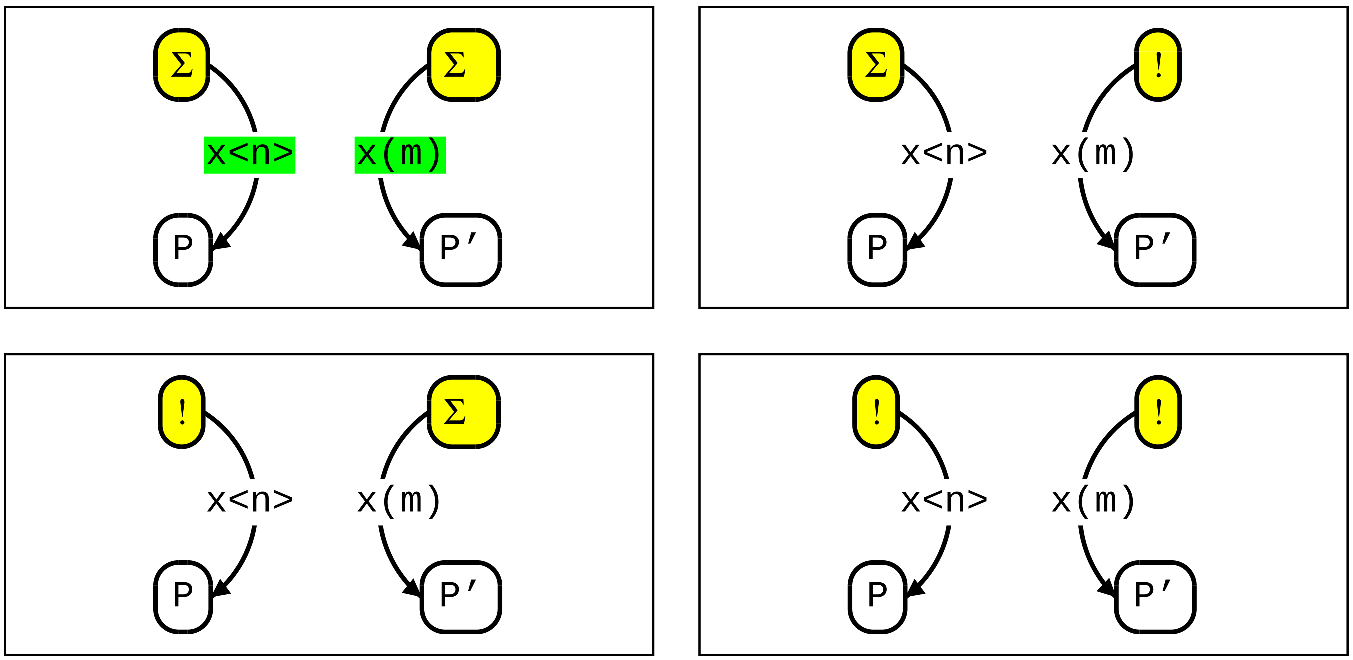
**Proof.** By Theorem 4 and by basic relationships between encoding and decoding. $\square$

# Graphical Semantics



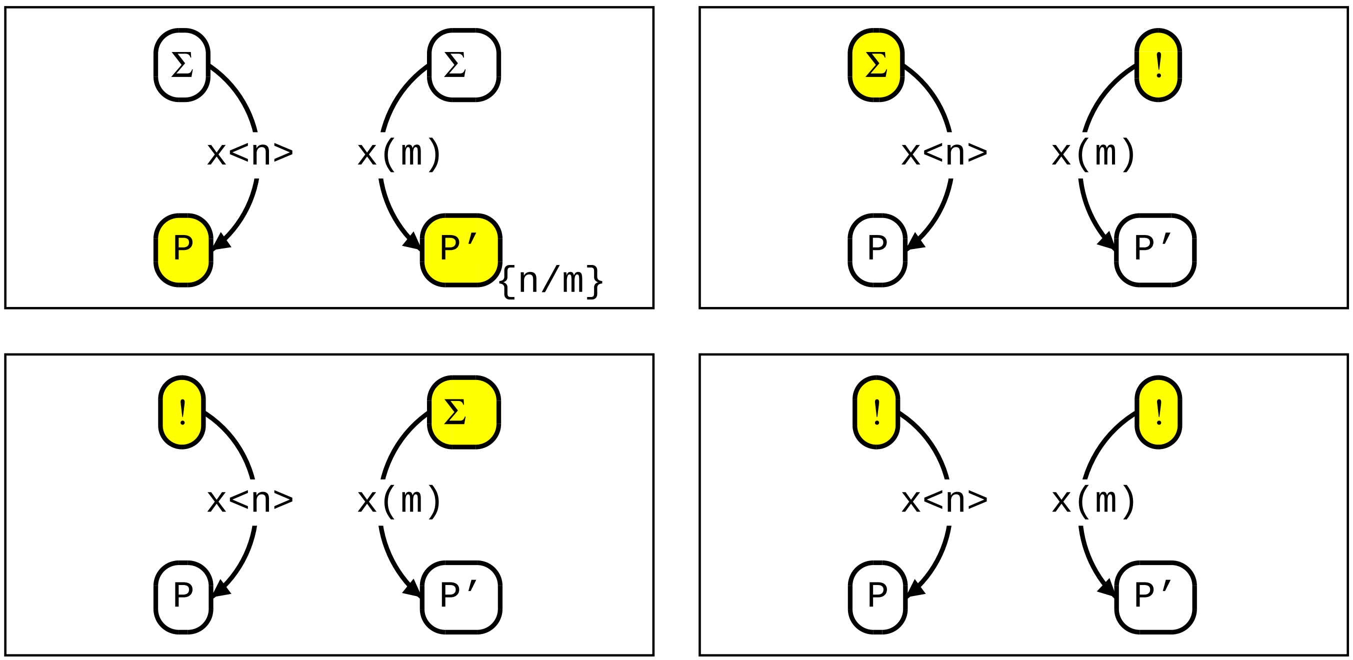

➢ Requires some imagination: for substituting names and for cloning graphs.

# Graphical Semantics



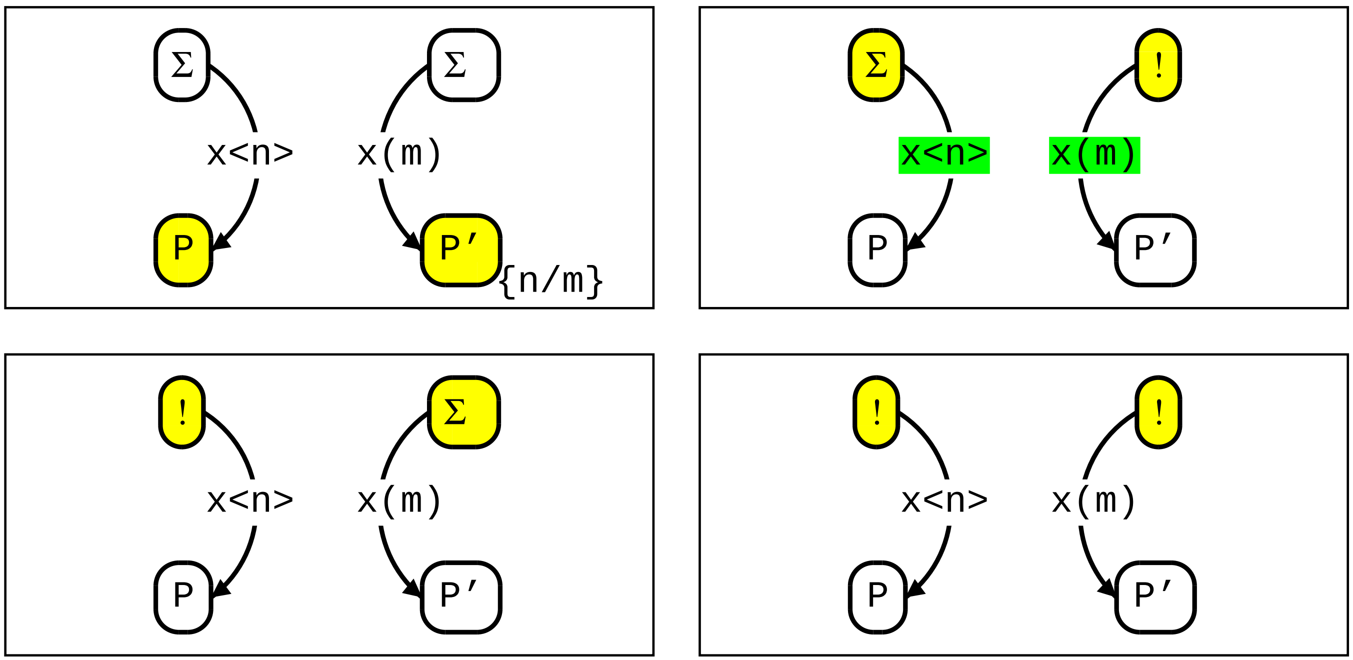

➢ Output $x\langle n \rangle$ can send a message to input $x(m)$ on channel $x$.

# Graphical Semantics



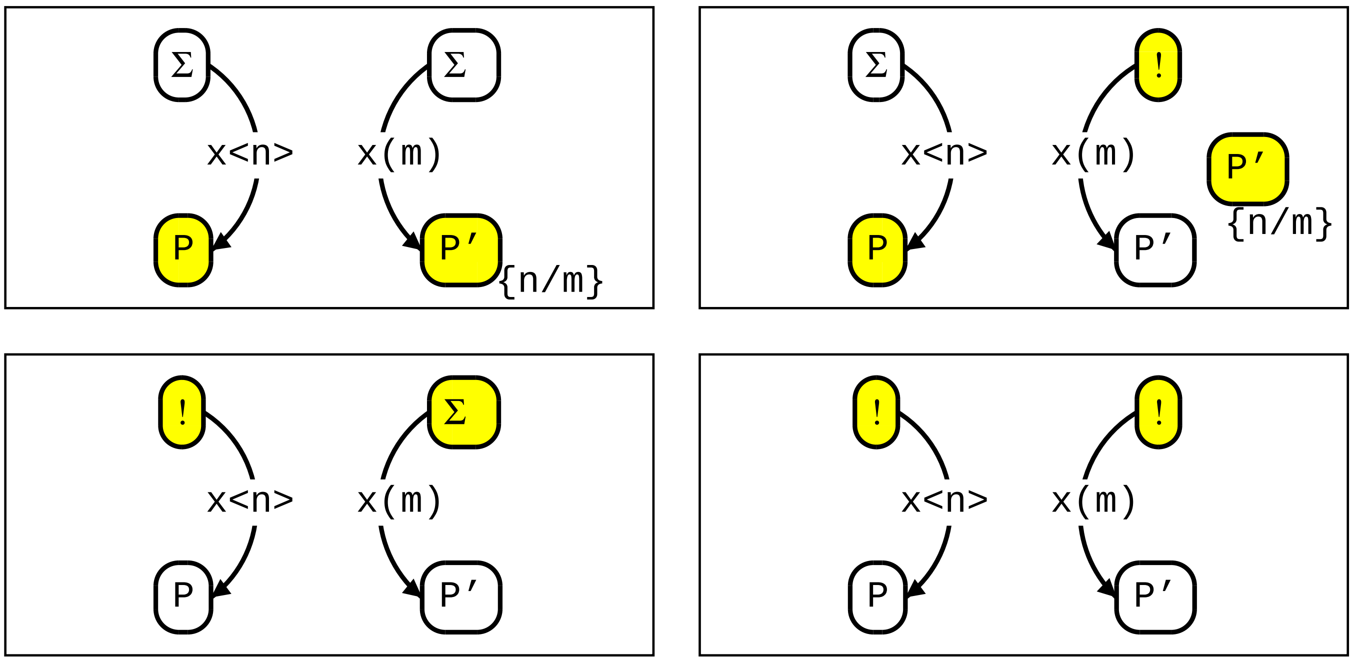

➢ $n$ is assigned to $m$ in process $P'$.

# Graphical Semantics



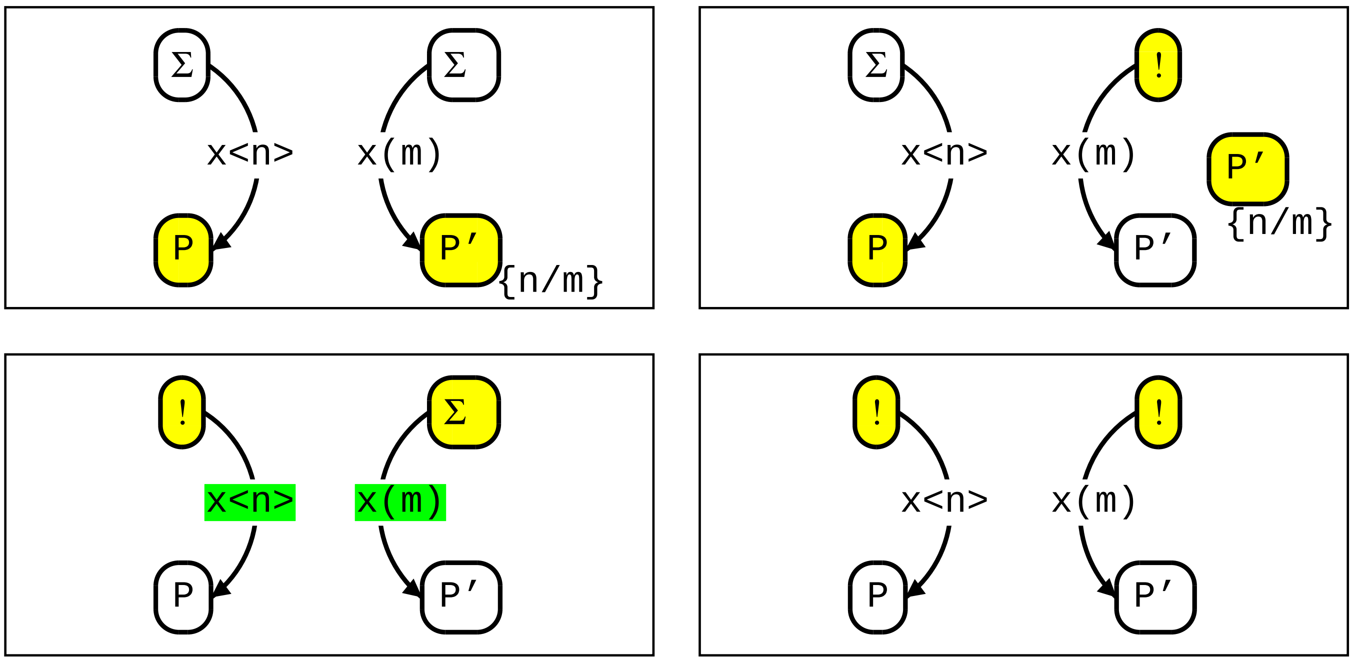

➢ Output $x\langle n\rangle$ can send a message to replicated input $!x(m)$.

# Graphical Semantics



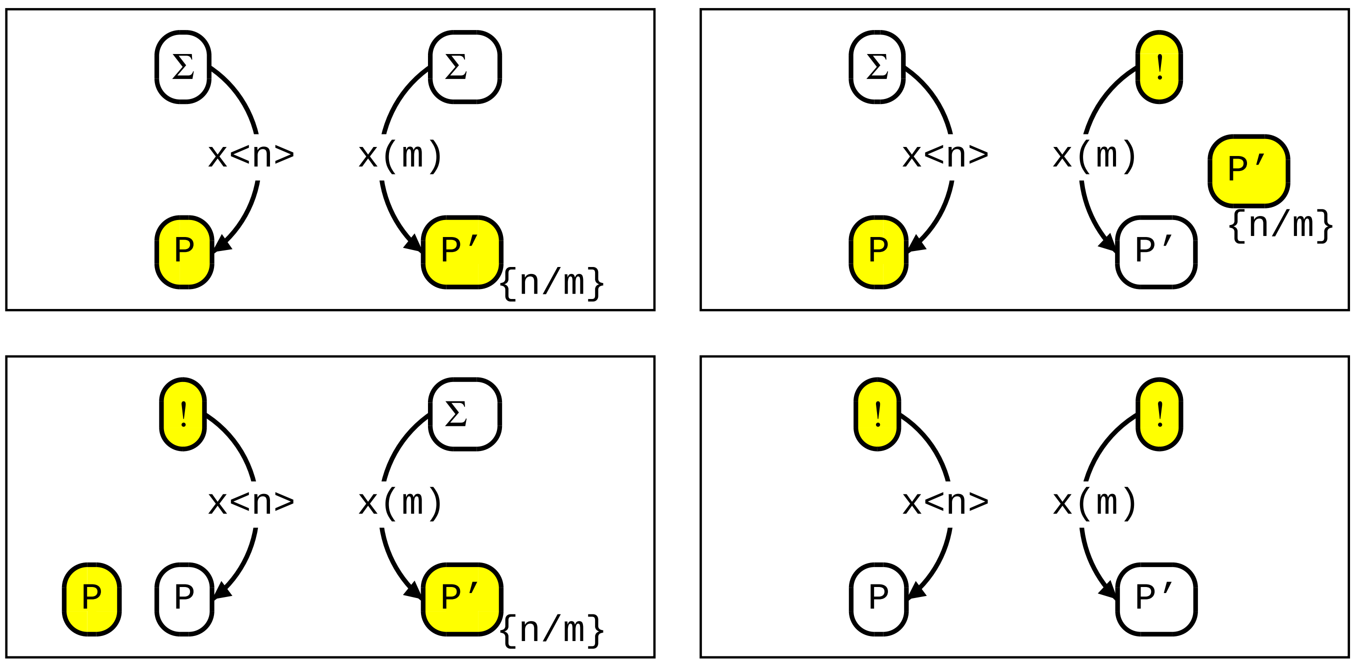

➢ A clone of $P'$ is spawned and $n$ is assigned to $m$ in the clone of $P'$.

# Graphical Semantics



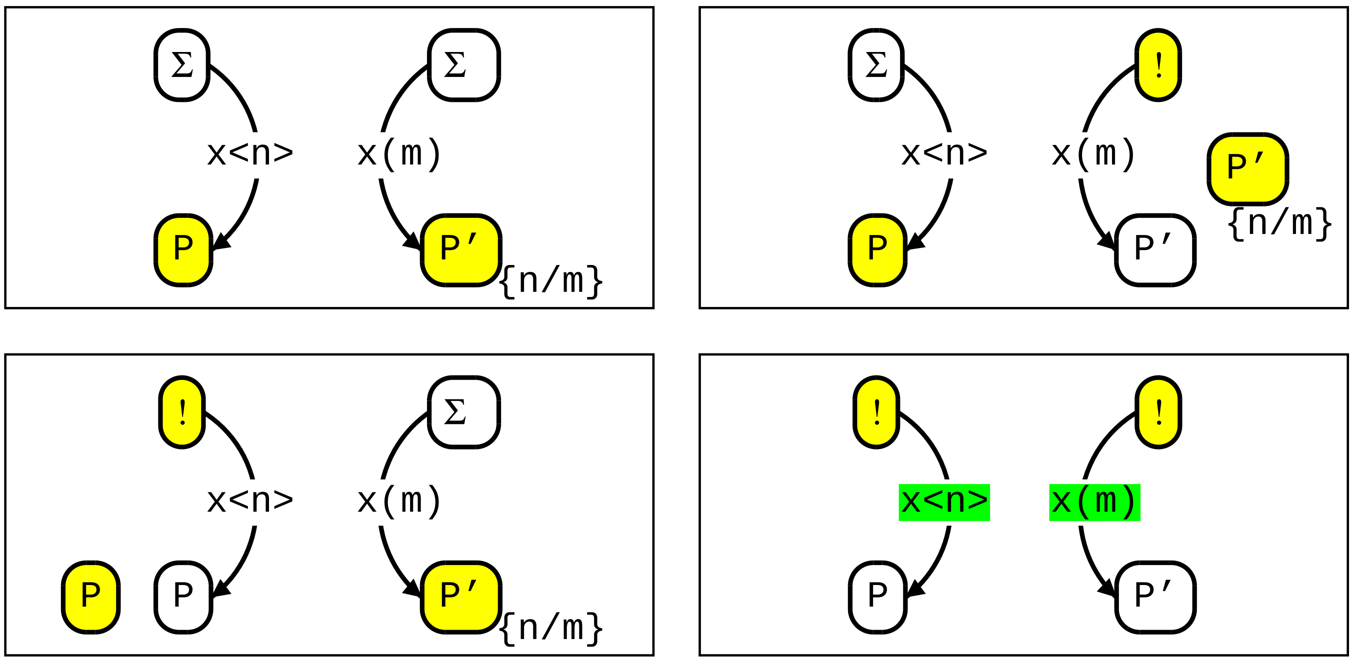

➢ Replicated output $!x\langle n\rangle$ can send a message to input $x(m)$.

# Graphical Semantics



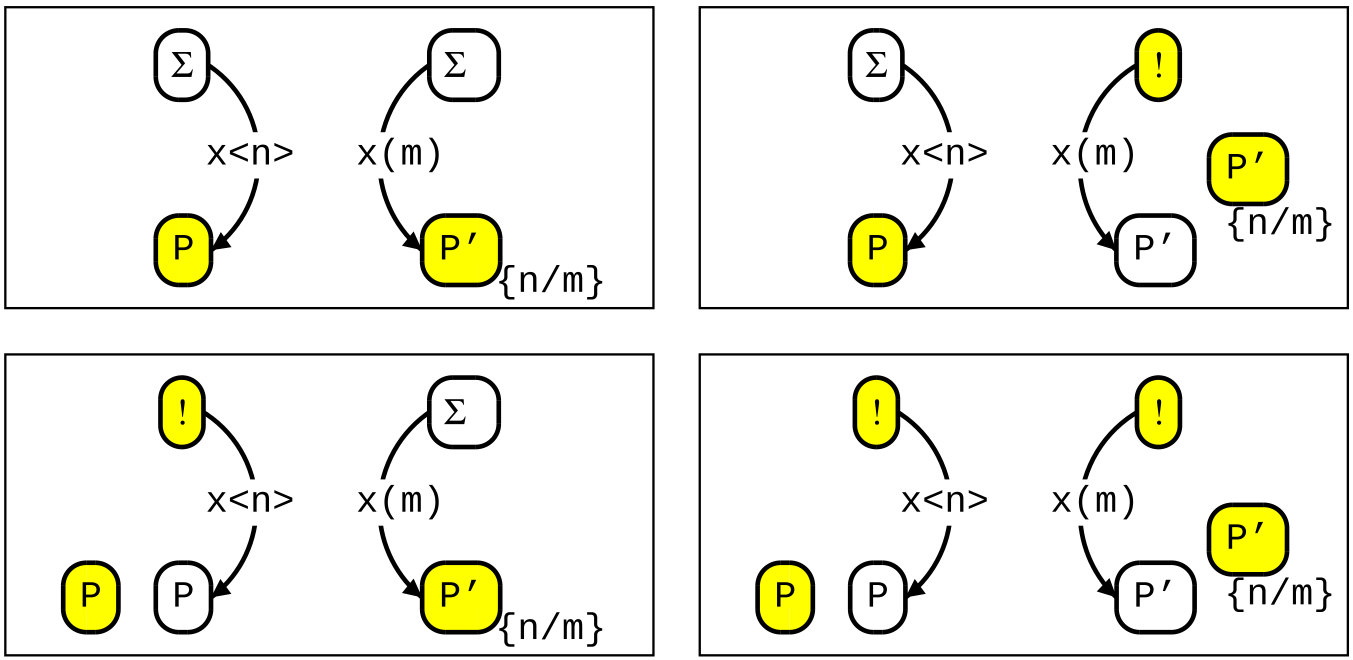

➢ A clone of $P$ is spawned and $n$ is assigned to $m$ in $P'$.

# Graphical Semantics




➢ Replicated output $!x\langle n\rangle$ can send a message to replicated input $!x(m)$.

# Graphical Semantics




➢ Clones of $P$ and $P'$ are spawned, and $n$ is assigned to $m$ in the clone of $P'$.

# Ionization: $Mg + 2Cl \rightleftharpoons Mg^{+2} + 2Cl^{-}$