**Stochastic Simulation**

- In explicit state simulation,
  - one transition per reaction/rule
  - each transition has a numerical weight
  - number of instances (multiplied by rate constant)
  - updating weights is just a matter of doing some arithmetic

- In implicit state simulation,
  - one transition per instance
  - weight = rate constant
  - must update transitions, not weights:
    - all new transitions → pick a transition
    - remove invalid transitions

[Conceptually, two phases: negative & positive update]

---

**Multi-sums (1)**

- recall the notion of pushout:
  - Given $A \rightarrow_{B} C$

- construct $B \rightarrow_{C} [st. the square commutes]

- there is a unique $D \rightarrow_{E}$ such that

- the triangles $B \rightarrow_{D} C$ commute

---

**Multi-sums (2)**

- if $A$ is an initial object, i.e. there is
  - exactly one arrow from $A$ to $B$
  - for all objects $B$

- a span $A \rightarrow_{B} \rightarrow_{C}$ is essentially just

- a choice of two objects, $B$ and $C$, and
  - the pushout is known as the sum, or
  - coproduct, written $B + C$

---

**Multi-sums (3)**

- Given $B$ and $C$, it is sometimes the case that

- there is not a unique object $B + C$ but a
  - family of objects $D_i$:

- given $B, C$, we construct a family

- of co-spans such that any

- factors uniquely through exactly one member of the family:

- The family of co-spans $B \rightarrow_{D_i} C$ is

- called the multi-sum (or multi-coproduct) of
  - $B$ and $C$
**Examples**

- In the category of sets and total functions, the co-product of \( A \) and \( C \) is the disjoint union \( B \cup C \).
- Similarly for the categories of simple graphs or rick graphs and homomorphisms.
- In the category of sets and injective functions, the co-product breaks down:

**5. Gluings and Overlaps**

- In the category of rick graphs and names, any pair of objects, \( A \) and \( B \), has a multi-sum.
- We describe the different ways that \( A \) and \( B \) can be "glued together" to form a bigger graph.

\[ A \rightarrow B \rightarrow \text{etc.} \]

- Any such gluing automatically induces an overlap (e.g., intersection of sets) by taking the pull-back:

\[ O_i : \]

- We want to say that \( R_1 \) activates \( R_2 \) if, for some overlap \( O_i \), the further pull-back satisfies:

\[ O_i \not\cong \text{not an isomorphism} \]

- This means that \( O_i \) was modified in some way by the action of \( R_1 \).

**9. Activation (5)**

- Activation is a relation between roles.
- Given \( R_i = \frac{p}{R_i} \), and \( q \): \( \frac{p}{R_2} \).
- We first construct the multi-sum at \( R_1, R_2 \):

\[ R_1 \rightarrow \ldots \rightarrow R_2 \]

- Then we construct the overlap of each \( G_i \):

\[ G_i \]

- We want to say that \( R_1 \) activates \( R_2 \) iff.

\[ R_1 \not\cong \text{not an isomorphism} \]

- So a new instance of \( R_2 \) has been created...
**Activation**

- Another way to say this:
  - If the error $Q_i^+$ is an isomorphism,

  This means that $O_i^+$ is contained in $P_i$, the
  produced region of $vi$

  $\implies O_i^+$ was not modified by $vi$ (by def.)

**Inhibition**

- Compute, as above, but for $L_i$ and $L_j$:

  \[
  O_i^+ \quad \overset{L_i}{\to} \quad P_i \quad \overset{L_j}{\to} \quad O_j^-
  \]

  and then:

  \[
  P_i \quad \overset{L_j}{\to} \quad O_j^-
  \]

  $\implies$ iso: $O_i^+ \cong O_j^-$ is contained in $P_i$
  and will not be modified by $vi$

  $\not\implies$ not iso: $O_i^+$ will be modified

  $vi$ inhibits $v_j$ iff, for some $i$, $O_i^+ \neq 0_i^-$

**NB:**

- Activation and inhibition are just relations between rules; but, if we know that particular $G_i^+$ (or $G_i^-$) for which $O_i^+ \neq 0_i^-$:
  - Any matching that factors thru $G_i^+$ (keep $G_i^-$) will be created/destroyed by the action of $vi$
  - This is the basis of how we 'update weights' for graph-based/implicit state simulation

**Implicit State Simulation**

- Given rules $v_i \ldots v_n$ where

  \[
  v_i := \overset{L_i}{\to} P_i \overset{L_j}{\to} O_j^-
  \]

  we compute the multi-sums

  \[
  G_{ij}^+: = R_i \overset{L_j}{\to} L_j
  \]

  \[
  G_{ij}^-: = L_i \overset{L_j}{\to} L_j
  \]

  ($j$ ranges over the family)

  and remove all induced $m_{ij}: L_j \to G$

  1. Pick a rule $v_i$ and a matching $m_i: L_i \to G$
  2. For all $j,k$, consider all factorizations $m_{ijk}: G_{ijk} \to G$
  3. Rewrite $G$ to $G'$ using $m_i$
  4. For all $j,k$, consider all factorizations $m_{ijk}: G_{ijk} \to G'$
  5. For all $j,k$, consider all factorizations $m_{ijk}: G_{ijk} \to G'$

**ISS**

- As a first optimization, keep only those members of the $G_{ijk}^+$ / $G_{ijk}^-$ families that have a productive / counter-productive overlap: i.e.

  $0_{ij}^+ \neq 0_{jk}^-$, $0_{ij}^- \neq 0_{jk}^+$

  - 'Productive' / 'counter-productive'
  - If $v_i$ does not activate $v_j$, there will be no productive overlap
  - If $v_i$ does not inhibit $v_j$, there will be no counter-productive overlap

**ISS (3)**

- Negative update

**ISS (4)**

- Positive update