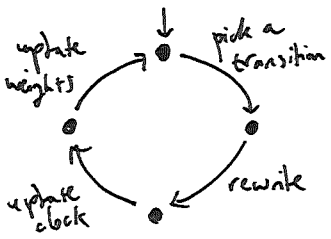
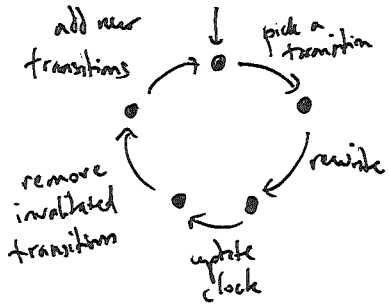
Stochastic Simulation

1



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

- in <u>implicit</u> state simulation
  - * one transition per <u>instance</u>
    - ↳ weight = rate constant
  - * must update <u>transitions</u>, not weights:



[conceptually, two phases: negative & positive update]

Multi-sums (1)

2

- recall the notion of push-out:

given $B \leftarrow A \rightarrow C$

construct $B \rightarrow D \leftarrow C$ [s.t. the square commutes]

such that, for any $B \rightarrow E \leftarrow C$,

there is a <u>unique</u> $D \rightarrow E$ such that

the triangles $B \rightarrow D \rightarrow E$, $C \rightarrow D \rightarrow E$ commute

Multi-sums (2)

3

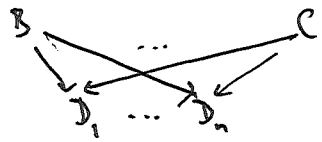
- if A is an <u>initial</u> object, i.e. there is exactly one arrow from A to B for <u>all</u> objects B,

a span $B \leftarrow A \rightarrow C$ is essentially just a choice of two objects, B and C, and the push-out is known as the sum, or co-product, written $B+C$

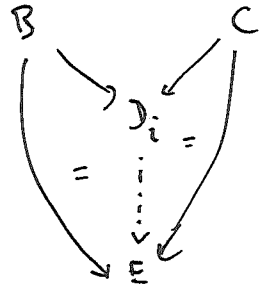
Multi-sums (3)

4

- given B and C, it is sometimes the case that there is not a unique object $B+C$ but a <u>family</u> of objects D_i:
  - * given B, C, we construct a family



of co-spans such that any $B \rightarrow E \leftarrow C$ factors <u>uniquely</u> through <u>exactly one</u> member of the family:



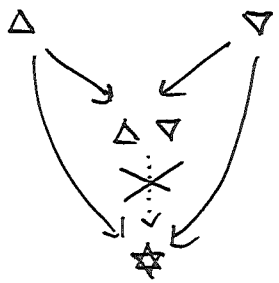
* the family of co-spans $B \rightarrow D_i \leftarrow C$ is called the <u>multi-sum</u> (or multi-co-product) of B and C

EXAMPLES

- in the category of sets and total functions, the co-product of B and C is the <u>disjoint</u> union $B \cup C$
- similarly for the categories of simple graphs or site graphs and homomorphisms

$\sim$

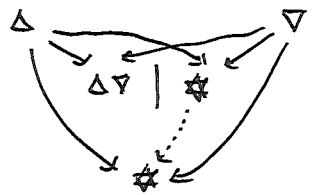
- in the category of sets and injective functions, the co-product breaks down:



↳ due to the co-span 'identifying' the two triangles...

↳ such 'unwarranted identifications' are only OK if we can use non-injective functions

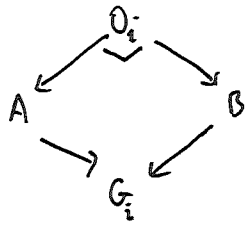
↳ otherwise, we get a multi-sum:



GLUINGS AND OVERLAPS

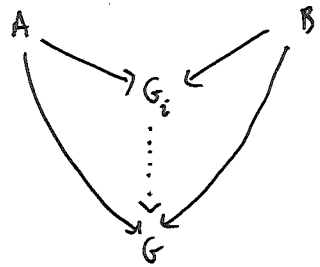
- in the category of site graphs and <u>monos</u>, any pair of objects, A and B, has a multi-sum
  - * these describe the different ways that A and B can be 'glued together' to form a bigger graph

  - * any such gluing automatically induces an 'overlap' (cf. intersection of sets) by taking the pull-back:

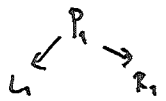


'TYPES OF MATCHING PAIRS'

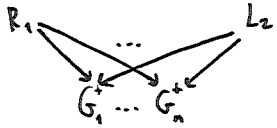
- each member G_i of the multi-sum of A and B defines one way that a pair of matchings — one from A, the other from B, can jointly map into some graph G:



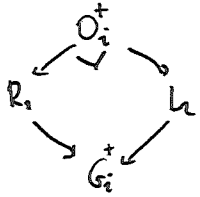
ACTIVATION (1)

- activation is a relation between rules
- given $r_1 :=$  and $r_2 :=$ 

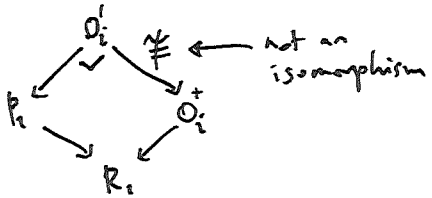
  * we first construct the multi-sum of R_1, L_2:



  * then we construct the overlap of each G_i:



- we want to say that <u>r_1 activates r_2</u> iff, for <u>some</u> overlap O_i^+, the further pull-back satisfies:



- this means that O_i^+ was <u>modified</u> in some way by the action of r_1

↳ and so a <u>new</u> instance of r_2 has been created...

ACTIVATION (2)

(9)

- another way to say this:

  if the arrow $O_i' \to O_i^+$ is an isomorphism,

  this means that O_i^+ is contained in P_1, the preserved region of r_1

  ↳ O_i^+ was not modified by r_1 (by def^{n})

INHIBITION:

(10)

- compute, as above, but for L_1 and L_2:

$$\begin{tikzcd} & O_i^- \arrow[dl] \arrow[dr] & \\ L_1 \arrow[dr] & & L_2 \arrow[dl] \\ & G_i^- & \end{tikzcd}$$

and then:

$$\begin{tikzcd} & O_i' \arrow[dl] \arrow[dr] & \\ P_1 \arrow[dr] & & O_i^- \arrow[dl] \\ & L_1 & \end{tikzcd}$$

iso or not?

↳ iso: $O_i' \cong O_i^-$ is contained in P_1 and will not be modified by r_1

↳ not iso: O_i^- will be modified

r_1 inhibits r_2 iff, for some i, $O_i' \not\cong O_i^-$

NB: activation and inhibition are just relations between rules; but, if we know the particular G_i^+ (or G_i^-) for which $O_i^+ \not\cong O_i'$:

- any matching that factors thru' G_i^+ (resp. G_i^-) will be created/destroyed by the action of r_1
- this is the basis of how we 'update weights' for graph-based/implicit state simulation

(11)

IMPLICIT STATE SIMULATION (1)

(12)

- given rules $r_1 \dots r_n$ where

$$r_i := \begin{tikzcd} & P_i \arrow[dl] \arrow[dr] & \\ L_i & & R_i \end{tikzcd}$$

we compute the multi-sums

$$G_{ij}^+ := \begin{tikzcd} R_i \arrow[dr] & & L_j \arrow[dl] \\ & G_{ijk}^+ & \end{tikzcd}$$

$$G_{ij}^- := \begin{tikzcd} L_i \arrow[dr] & & L_j \arrow[dl] \\ & G_{ijk}^- & \end{tikzcd}$$

(k ranges over the family)

ISS (2)

(13)

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

$$O_{ijk}' \not\cong O_{ijk}^+ \qquad\qquad O_{ijk}' \not\cong O_{ijk}^-$$

'productive' 'counter-productive'

- if r_i does not activate r_j, there will be no productive overlap
- if r_i does not inhibit r_j, there will be no counter-productive overlap

ISS (3)

(14)

1. pick a rule r_i and a matching $m_i : L_i \to G$
2. for all j,k, consider all factorizations $m_{ijk}^- : G_{ijk}^- \to G$

$$\begin{tikzcd} L_i \arrow[r] \arrow[dr, bend right, "m_i"'] & G_{ijk}^- \arrow[d, dotted, "m_{ijk}^-"] & L_j \arrow[l] \arrow[dl, dotted, "m_{jk}^-"] \\ & G & \end{tikzcd}$$

negative update

and remove all induced $m_{jk}^- : L_j \to G$

3. rewrite G to G' using m_i
4. for all j,k, consider all factorizations $m_{ijk}^+ : G_{ijk}^+ \to G$

$$\begin{tikzcd} R_i \arrow[r] \arrow[dr, bend right, "m_i'"'] & G_{ijk}^+ \arrow[d, dotted, "m_{ijk}^+"] & L_j \arrow[l] \arrow[dl, dotted, "m_{jk}^+"] \\ & G' & \end{tikzcd}$$

positive update

and add all induced $m_{jk}^+ : L_j \to G'$