Interaction Nets

6CCS3COM Computational Models

Josh Murphy

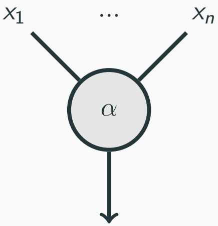
Contents

Interaction Nets

- A model of computation developed by Lafont in 1990s, based on the paradigm of **interaction**.
- They are an inherently **distributed** form of computation.
- Commonly represented **graphically**.

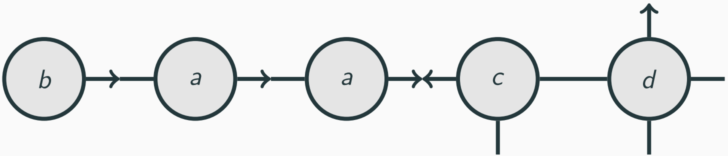
Agents

- **Agents** are nodes with:
  - One **principal port** (an outgoing edge depicted with an arrow).
  - A set of *n* **auxiliary ports** (other connected edges, labelled).
  - A **type** that determines its number of auxiliary ports (node symbol).
- Agents correspond with **functions**.



Nets

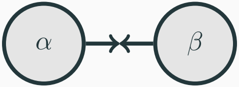
- An **interaction net** is a graph of agents, where agents are connected at their ports, and their is only one edge connected to each port.



- If a port is not connected to another agent then we say it is **free**.
- The **interface** of a net is its set of free ports.
- Two special nets: the **empty net** and **wirings** (nets with only edges)
- Nets correspond with **programs**.

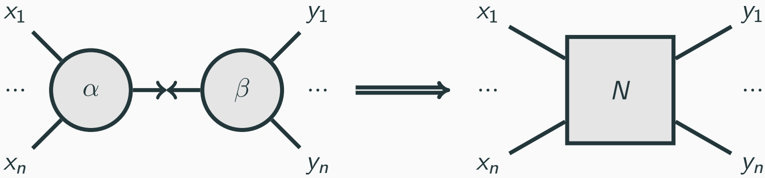
Active Pairs

- A pair of agents which are connected by their primary ports are called an **active pair**.
- Active pairs are the redexes of interaction nets — where all the computation happens!



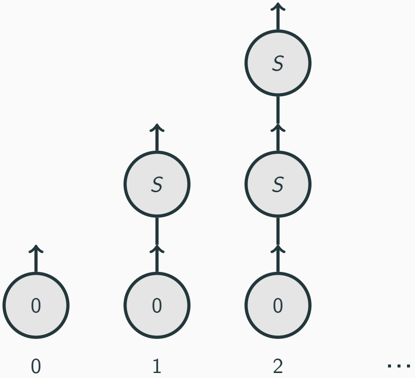
Interaction Rules

- An **interaction rule** replaces an active pair of agents (α, β) by a new net N with the same interface.
- There can be at most one interaction rule for each pair of agent types.



Example: natural numbers

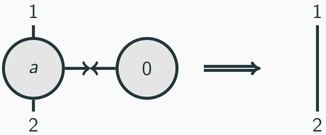
- Natural numbers can be defined over a 0 agent and the successor agent S.



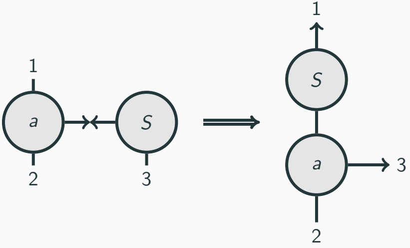
- These agents should never form an active pair, so no interaction rules are defined between them.

Example: addition

Rule 1

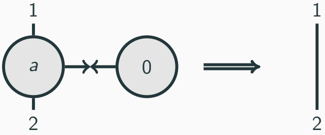


Rule 2

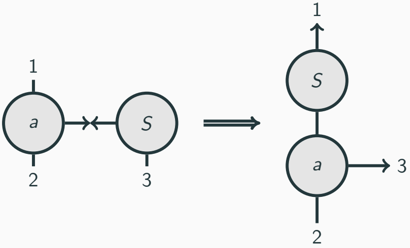


Example: addition

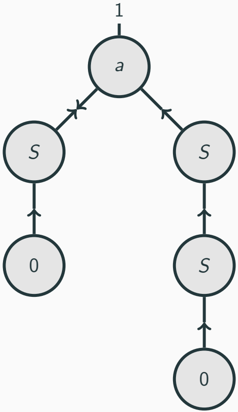
Rule 1



Rule 2



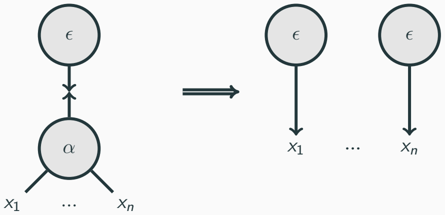
Exercise: Apply Rules 1 and 2 to the IN below.



Two common agents: Erasing and Duplication

- Two common agents in many Interaction Nets are the **Erasing** (ϵ) and **Duplicating** (δ) agents, with the following interaction rules.

Erasing:

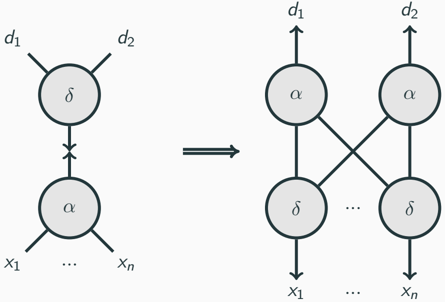


- The erasing agent is helpful to erase sub-nets (*e.g.* x_1 to x_n).

Two common agents: Erasing and Duplication

- Two common agents in many Interaction Nets are the **Erasing** (ϵ) and **Duplicating** (δ) agents, with the following interaction rules.

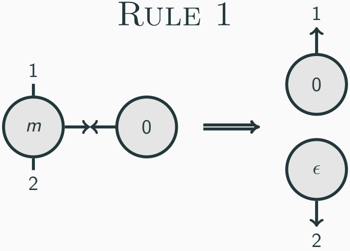
Duplicating:



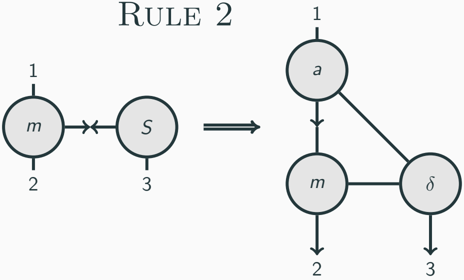
- The duplicating agent is helpful to copy the same operation (*e.g.* α) to two sub-nets (*e.g.* d_1 and d_2).

Example: multiplication

Rule 1

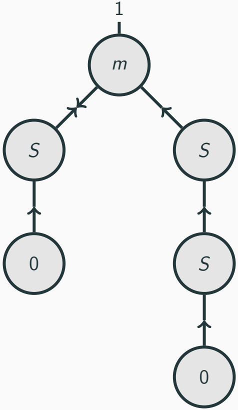


Rule 2



Rule 2: mult(x,y) = mult(x,y-1)+x

Exercise: Apply Rules 1 and 2 to the IN below.



Example: maximum

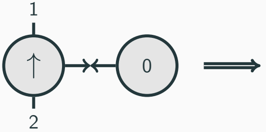
- The **maximum** function is typically defined as follows:
  - $\max(0,y) = y$
  - $\max(x,0) = x$
  - $\max(S(x), S(y)) = S(\max(x,y))$

- However, this specification can't be implemented directly into interaction nets: it is defined by cases on both of the arguments.
  - This would require two principal ports for the max agent.

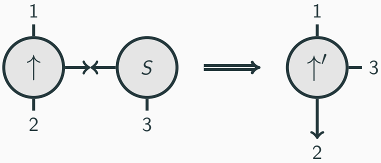
- So, we use this definition instead:
  - $\max(0,y) = y$
  - $\max(S(x), y)$ **= max'(x,y)**
  - $\max'(x,0) = x$
  - $\max'(x, S(y)) = S(\max(x, y))$

Example: maximum

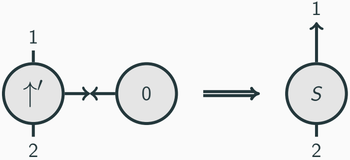
Rule 1



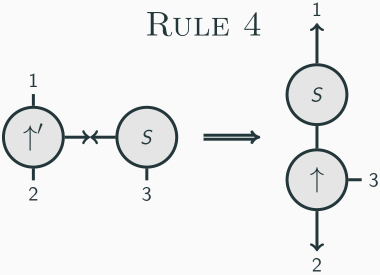
Rule 2



Rule 3



Rule 4



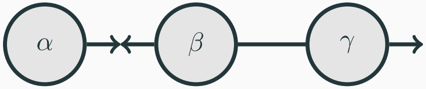
Exercises

- Define, using interaction nets, the following agents over natural numbers (represented with 0 and S as in previous examples).
    - Z, which produces a value True iff the number is 0, False otherwise.
    - $\downarrow$, which computes the minimum of two numbers.
    - **Challenge:** F, which computes the factorial of a number.

Normal forms

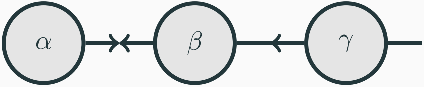
- When to stop applying interaction rules to a net?
  - **Normal form:** stop when they are no more active pairs.
  - **Interface normal form:** stop when the interface is only principal ports, or if there are auxilliary ports on the interface then they will never become principal by further interaction.
- Interactions are **confluent**.
  - It doesn't matter which order interactions occurs, you will always reach the same normal form or interface normal form.

Interface normal form



- This net is in **interface normal form**, but not normal form. Although this net has an active pair, its interface is composed only of principal ports.

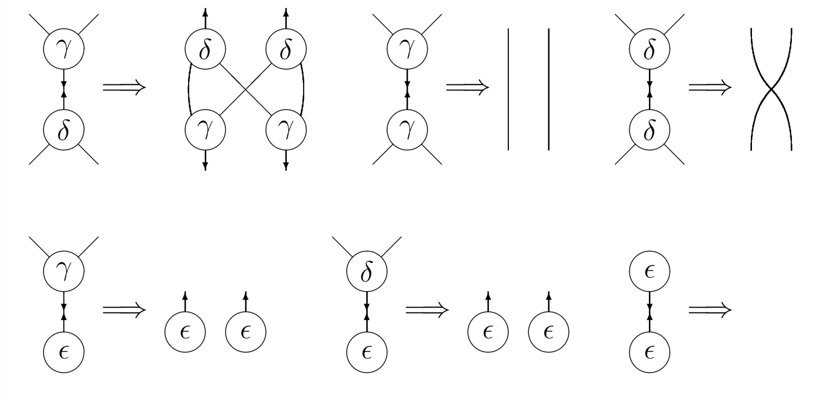
Interface normal form



- Is this net in **interface normal form**?
  - It depends on the outcomes of interactions.
  - If resolving the interaction between α and β (and any subsequent interactions) does not change the port of the interface to a principal port then the net is in interface normal form.

Lafont's interaction combinators

- These interaction rules are **universal**: all other interaction nets can be encoded using just these rules.



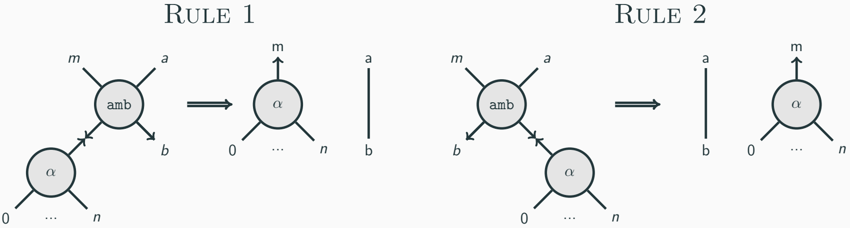
Determinism

- Interaction Nets are inherently deterministic.
  - **Exercise:** why? **agents have only one principal port**
    can only interact with one agent at a time
    for any active pair, there is one rule to apply.
- However, Interaction Nets allow for parallel computation (local interactions and confluence).
- If Interaction Nets could model non-determinism then they would be ideal for modelling parallel programming.

Non-deterministic extension

- Possible extensions include:
  1. Multiple interaction rules for each pair of agents, from which one will be chosen at random for each active pair.
  2. Edges that connect more than two ports, where the branch is chosen at random.
  3. Allowing agents to have multiple principal ports — known as ambiguous agents.
- In fact, it is sufficient to just add one ambiguous agent, `amb`, which has two principal ports.
  - If an `amb` agent forms two active pairs, then the interaction to be resolved is chosen at random.

`amb` agent



Summary

- Interaction Nets are a specialised model of computation with properties that make them appropriate for **distributed** and (local) **parallel** computation.
- Next, we will look at how to model **concurrent** computations.