Parameterised Session Types

Communication Patterns Through the Looking Glass of Session Types

Andi Bejleri

Imperial College London

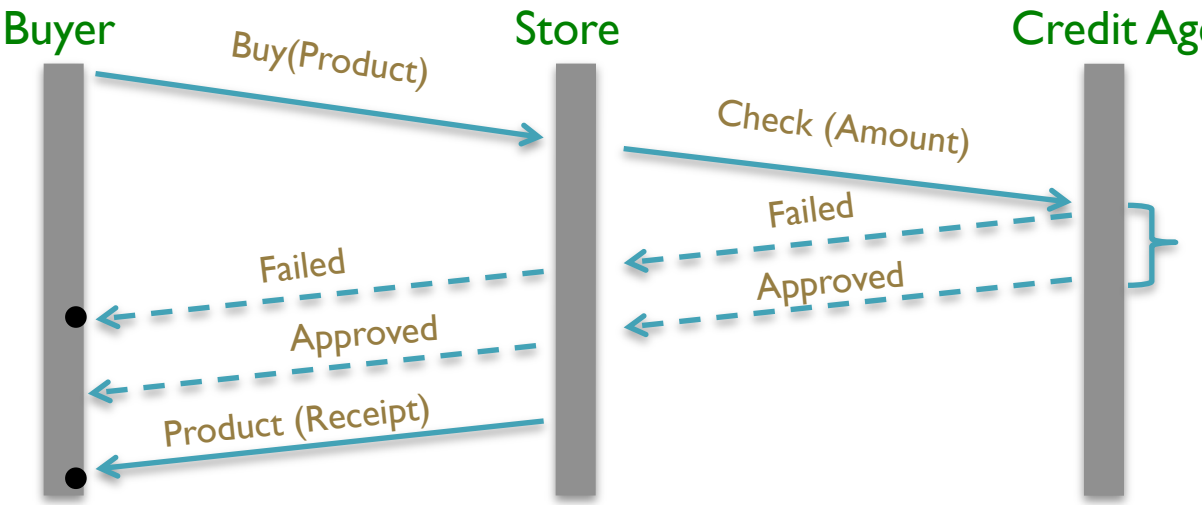
Behavioural Types- April 21, 2011

Session Types: What do they offer

- Intuitive, light-weight type annotation
  - Syntax of simply names and arrows, no question marks or bangs
  - One type for n processes
- A mode of organising *structured* communications from a *global point of view*
  - Consequently, describe the order of communications (part of programs logic)
- Efficient type-checking strategy of processes through projection of global types onto participants

Example: Buyer-Store-Credit Agency

Sequence diagram



Global type

$$Buyer \rightarrow Store : \langle String \rangle .$$

$$Store \rightarrow Credit\ Agency : \langle Int \rangle .$$

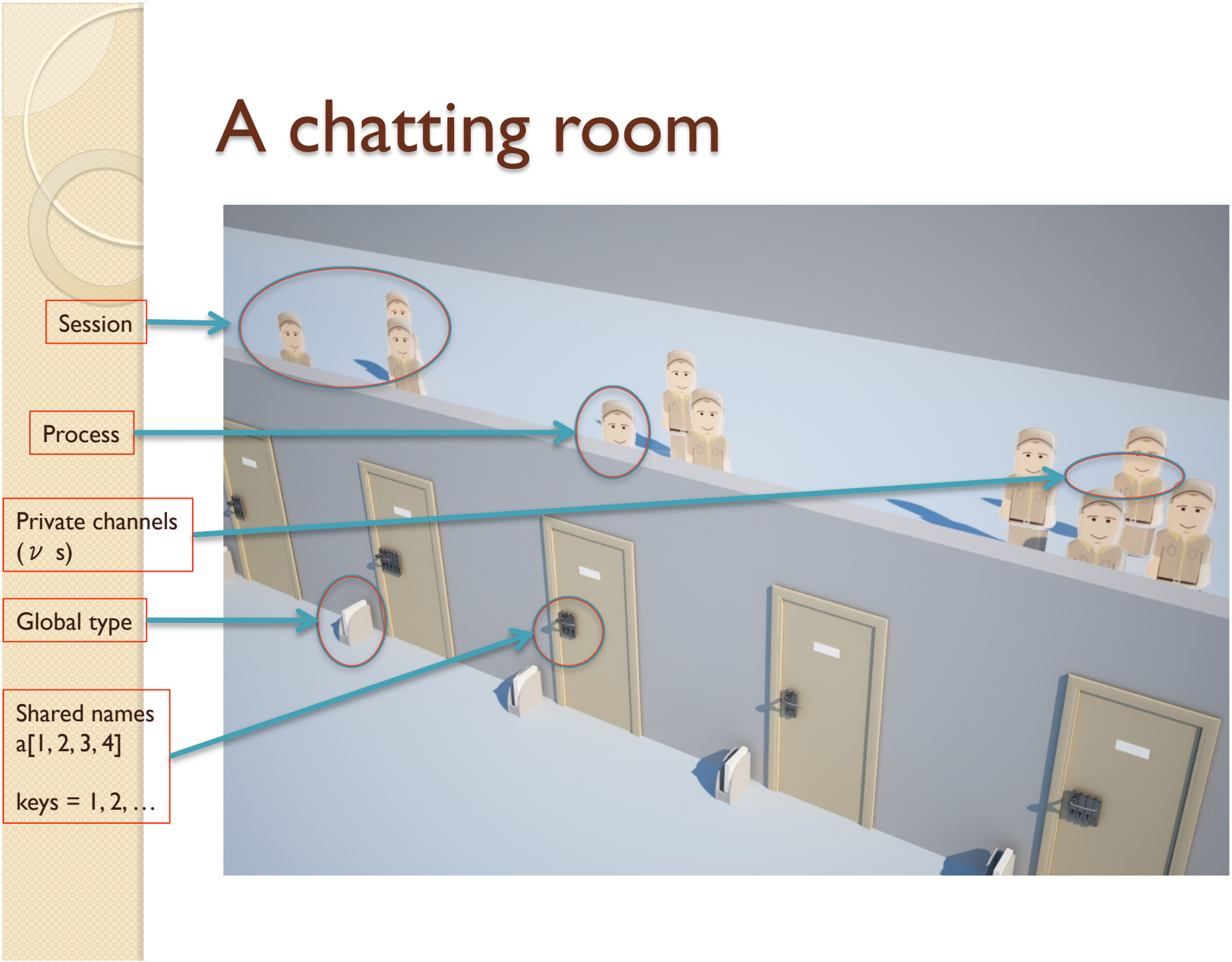
$$Credit\ Agency \rightarrow Store : \begin{cases} FAILED : Store \rightarrow Buyer : \{ FAILED : End \\ APPROVED : Store \rightarrow Buyer : \{ APPROVED : Store \rightarrow Buyer : \langle String \rangle .End \end{cases}$$

Message Causality

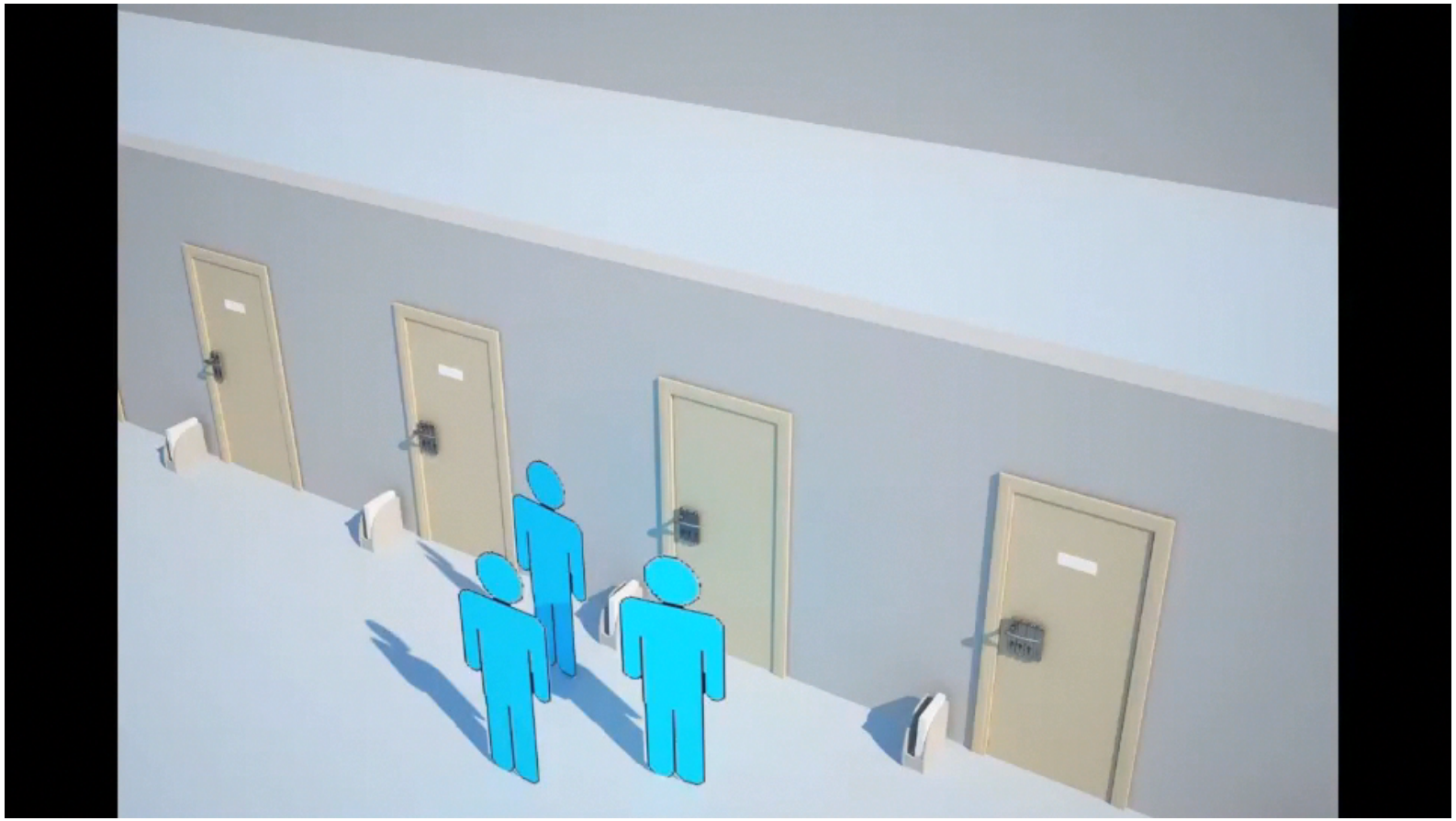
Branching Causality

Processes

A chatting room

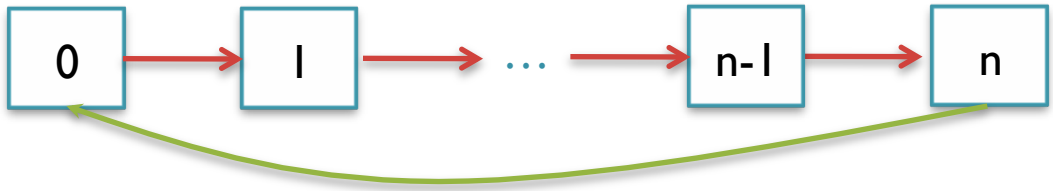


Semantics



Communication patterns

- Describe simple and elegant structured interactions
  - Ring, Star, Tree, 2D-Mesh, Hypercube
  - Used in: parallel algorithms, data exchange protocols, web services
- Simulation of the n-body algorithm over the Ring pattern



  - First iteration: send particles to the neighbor on its left and calculate forces exerted within their particles.
  - i-th iteration: forward particles received in i-1 iteration, calculate forces of them and receive the next particles set

Guaranteeing communication-safety for communication patterns I

- Communication patterns and global types describe structured interactions
  - Global types serve not only as a blue print of the system's architecture but also as a type system that guarantees communication-safety

Guaranteeing communication-safety for communication patterns II

- Communication patterns describe structured interactions of an arbitrary number of participants
  - In the n-body algorithm (for another implementation) ones needs 80 processes for 64 particles and 110 for 128 to obtain a good speed up.

- Unable to express sessions where the number of participants is known only at run-time
  - In parallel algorithms, the number of participants depends on the size of the problem instance

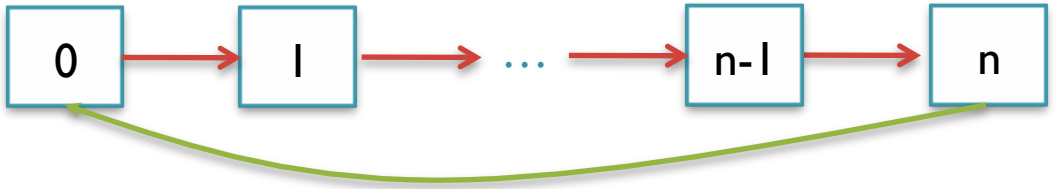
Parameterised Session Types

1. Parameterise participants
2. Iterate over parameterised causalities that abstract repetitive behavior of pattern
3. Compose sequentially global types

Gödel's R– primitive recursive function

Encoded as an iterator, for (i=n-1; i>0; i--)G; G'

The Ring communication pattern



R n → 0:<U>.end
λ i. λ X.n-i-1 → n-i:<U>.X
n

R 2 → 0:<U>.end
λ i. λ X.2-i-1 → 2-i:<U>.X
2

iterate

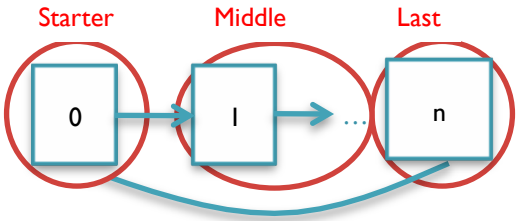
Ring of length 3

0 → 1:<U>.
1 → 2:<U>.
2 → 0:<U>.
end

Roles

- Blueprint that describes the nature of a communication pattern and the behavior that all run-time processes will share
- Concept similar to classes

Starter role of the Ring pattern



In our calculus (not complete):

$$Starter = y!\langle W[1],"1"\rangle; y?\langle W[n],x\rangle; R$$

Communication abstraction of processes/objects that will be generated at runtime

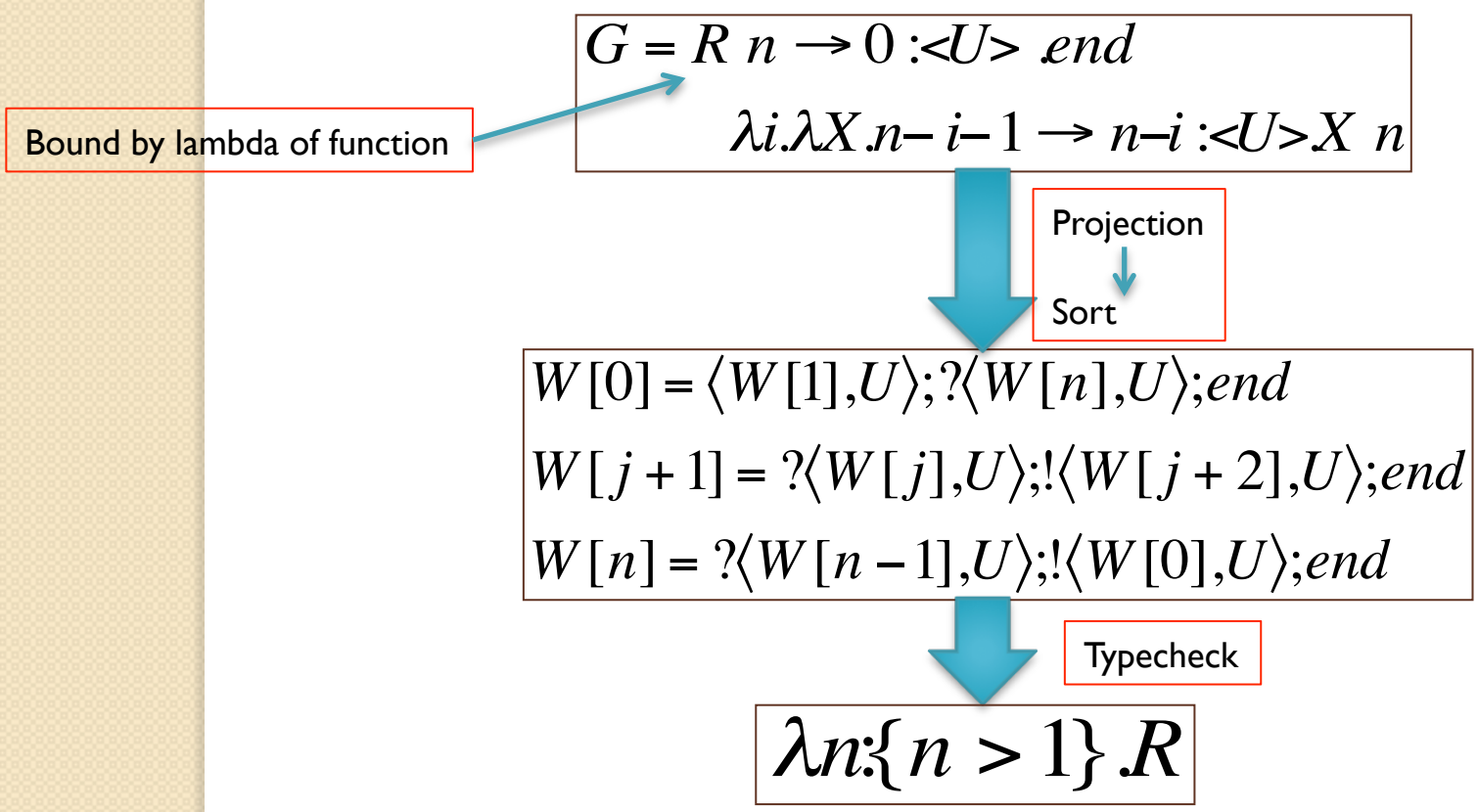
Abstraction of neighbors

In Java (not complete):

```
public class Starter{
    public Starter(int port_l, String host_r, int port_r){
        // Set up the sockets for the pattern
        ...
        PrintWriter out = null;
        BufferedReader = null;
        try{
            serverSocket = new ServerSocket(port_l);
            clientSocket = new Socket(host_r, port_r);
            out = ... // Init. the output stream on clientSocket
            in = ... // Init. the input stream on serverSocket
            // Exchange messages with neighbors
            out.println("1");
            String m = in.readLine();
            ... // Close streams and sockets
    }
}
```

Static type system

- Efficient type-checking through projection of global types onto the principals



Projection

- Projection

$$p \to p' : \langle U \rangle . G \upharpoonright q = \begin{cases} !\langle p'\{p = q\}, U \rangle (p); ?\langle p\{p' = q\}, U \rangle (p'); G \upharpoonright q & \text{if } C \triangleleft p = q \text{ and } C \triangleleft p' = q \\ !\langle p'\{p = q\}, U \rangle (p); G \upharpoonright q & \text{if } C \triangleleft p = q \\ ?\langle p\{p' = q\}, U \rangle (p'); G \upharpoonright q & \text{if } C \triangleleft p' = q \\ G \upharpoonright q & \text{otherwise} \end{cases}$$

- Ring pattern
  - Middle worker $W[j+1]$ $(1..n-1)$ appear in both sides $n-i-1 \to n-i$ …1⟶2:<U>.2⟶3:<U>…,

Sorting

- From projection of the Ring global type on $W[j+1]$:

$$!\langle W[j+2],U\rangle(W[n-i-1]);?\langle W[j],U\rangle(W[n-i])$$

Position where the action appears

Instance of the above type for W[2]

$$!\langle W[3],U\rangle;?\langle W[1],U\rangle$$

- Sort the sequence of actions on their appearance in G

$$?\langle W[1],U\rangle;!\langle W[3],U\rangle$$

- Sorting of the sequence of actions on the participants of G who represent them, $W[n-i-1], W[n-i]$

Static type system

- Values of parameters range over infinite sets of parameters

$$G = R\ n \rightarrow 0 :<U>\ .end$$
$$\lambda i.\lambda X.n - i - 1 \rightarrow n - i :<U>.X\ n$$

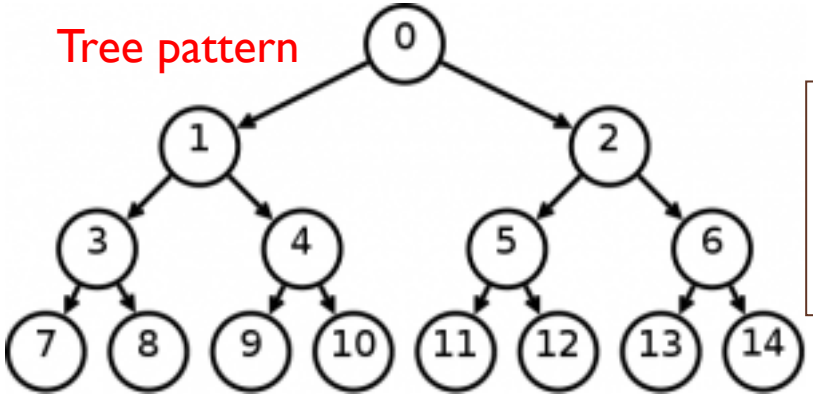
Full computation power of program (P)

$$\lambda n:\{n > 1\}.P$$

Control index calculation in MPI

- Index calculation in global types is simpler than the one in roles
  - A direct advantage of the global representation of interactions

Tree pattern



$$G = R\ end$$
$$\lambda i.\lambda X\, i \rightarrow 2 * i + 1 :<U>.$$
$$i \rightarrow 2 * i + 2 :<U>\ .X\ 2^n - 1$$

$$OddLeaf = a[2*(2^{n-1}+i-1) + 1](y).y\ ?(2^{n-1}+i-1, z); S$$

Final remarks

- A system that expresses sessions of *an arbitrary number of processes* through the *role* idiom, preserving:
  - An intuitive, light-weight type annotation
  - A global description of structured interactions
  - Efficient type-checking strategy: projection of global types onto participants and sorting of actions in the role types
  - A non-conservative type system that allows parameters to range over an infinite set of values

References

- *Parameterised Session Types: Communication patterns through the looking glass of session types*. Dissertation (To appear sometime in May-June)
  - *Practical Parameterised Session Types*. ICFEM 2010
  - *Session-based Programming for Parallel Algorithms*. PLACES 2009.
  - *Synchronous Multiparty Session Types*. PLACES 2008.

http://www.doc.ic.ac.uk/~ab406/