Balzer 1.1

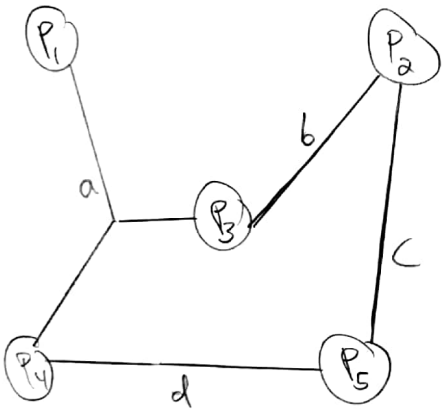
Session typed Concurrent Programming (STCP)

RoadMap:
1. message-passing concurrent programming
2. session types
3. Linear-logic-Session types
4. Manifest sharing

learning objectives:
1. how can we program using message-passing concurre
2. what session types are about
3. Benefits of Linear logic to programming
4. How to accomodate sharing in a logically motivated way

message-passing concurrency

processes that compute by exchanging messages along channels.



a network ↦ processes P_i connected by channels $\{a,b,c,d\}$.

channels are n-ary.

$ar(a) = 3$

$ar(b) = ar(c) = ar(d) = 2$

i.e., P_1 can send a message to either P_3 or P_4 on channel a.

thus, non-determinism.

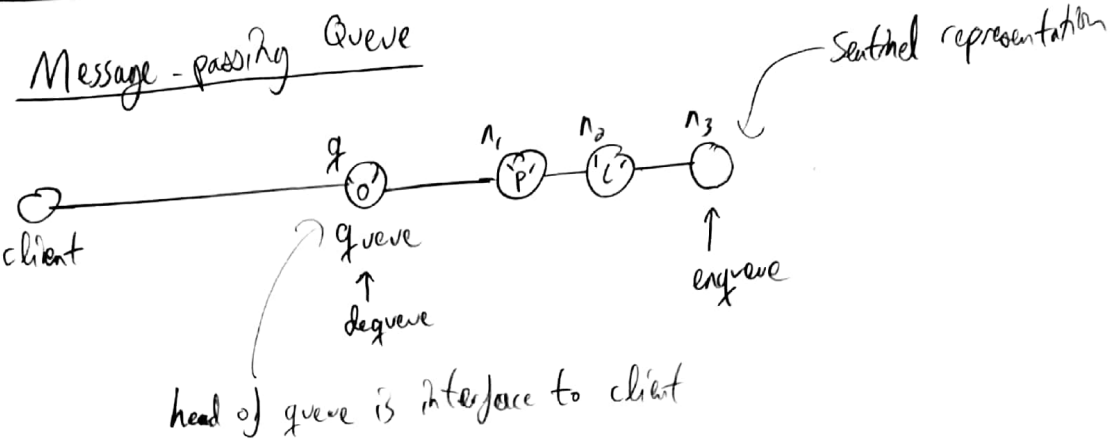
π-calculus (Milner, 1992)

↳ "universal"

Functional Programming is to λ-calculi as STCP are to process-calculi.

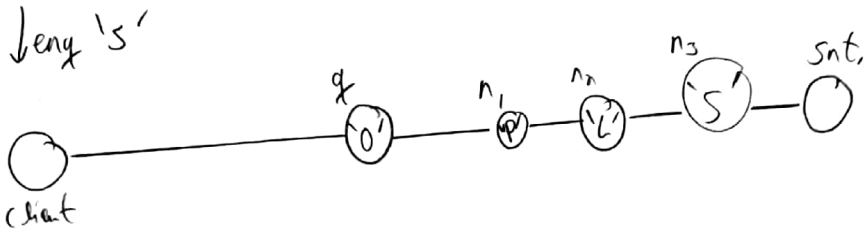
Balzer 1.2

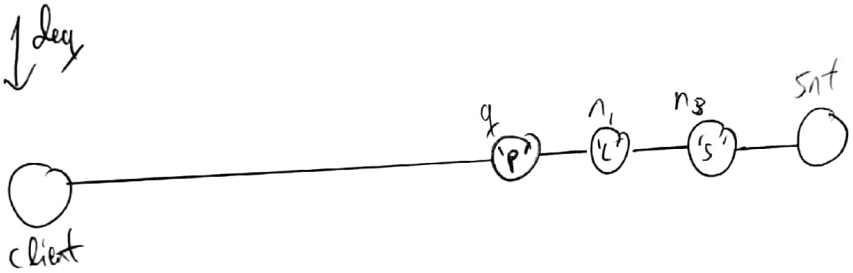
Message-passing Queue

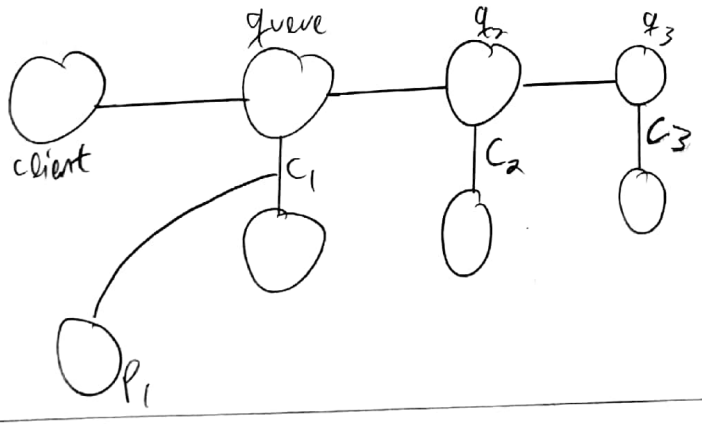


this is the process.

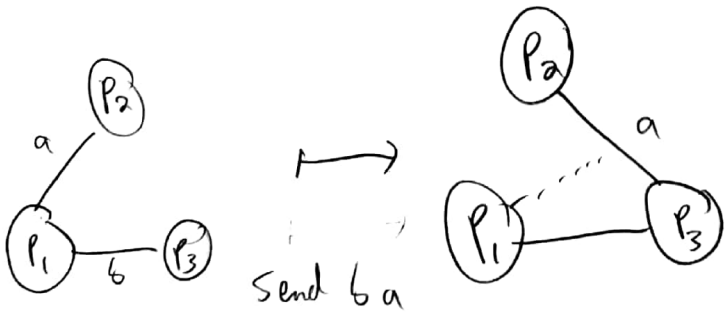
head of queue is interface to client







Channels passing other channels



"Send along b a"

"mobility"

"higher order channels"

Balzer 1,3

How do you type message-passing protocols

- types for protocols of message-exchange

session types (Kohei Honda 1993)

$$A ::= ?[T].A' \mid ![T].A' \mid \&\{l_1:A_1, \dots, l_n:A_n\} \mid \oplus\{l_1:A_1, \dots, l_n:A_n\} \mid \text{end} \mid X \mid \mu X.A'$$

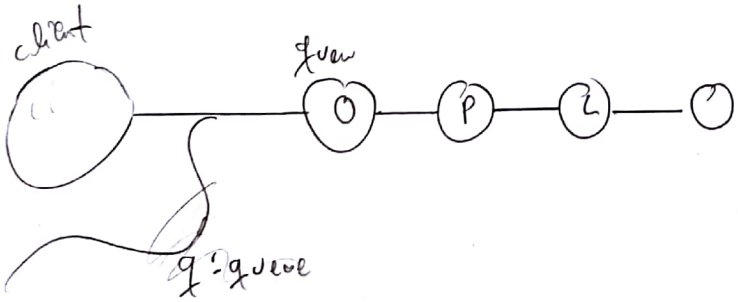
("with" over &; "plus" over ⊕; "type var" over X)

- $?[T].A'$: "I am willing to receive an input of type T, and after I will continue to be type A'"
- $![T].A'$: "I am willing to send ..."
- $\&\{l_1:A_1, \dots, l_n:A_n\}$: "choose between different types by label" / give choice to client / external choice
- $\oplus\{l_1:A_1, \dots, l_n:A_n\}$: give choice to process / internal choice.

$$T ::= A \mid \text{int} \mid \text{char} \mid \dots$$

a queue is an equi-recursive type

$$\text{queue} = \&\{\text{enq}: ?[\text{char}].\text{queue},\ \text{deq}: \oplus\{\text{none}: \text{end},\ \text{some}: ![\text{char}].\text{queue}\}\}$$



q : &{enq: ..., deq: ...} → "send 'enq' along q"

q : ?[char].queue → "send 's' along q"

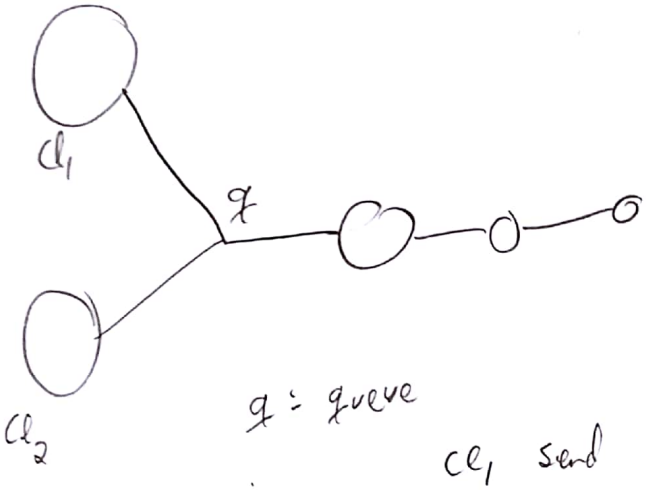
q : queue

IC stateful, types change over the

type of channel/process changes w message exchange.

Balzer 1.4

Consider two clients



q : queue

cl_1 send eng along q

q : ?[char], queue

cl_2 send deg along q

preservation in session types, called session fidelity guarantees that expectation of client matches with the one of provider if they match initially

R channels are resources (from linear logic)

linear logic rejects weakening and contraction