Game Semantics

* the various "domain models" of PCF all live in categories of <u>functions</u>:
  - CPOs and continuous functions
  - dI-domains and stable functions
  - hypercoherences and strongly stable functions
  - ... (and more!)
* none of these models fully captures the <u>sequential</u> nature of PCF computation...

* before the invention of game semantics (c. 1993) only <u>one</u> model of PCF existed that wasn't based on a category of functions: the <u>sequential</u> <u>algorithms</u> model

this model is still a category however

* as the name suggests, this model captures the idea of sequentiality...

no "parallel or," etc.

* but <u>several</u> algorithms can implement the <u>same</u> function

$\Longrightarrow$ an "intensional" model

the semantics of a term contains information about <u>how</u> the term is computed, not just the function computed...

* Game semantics, like sequential algorithms, captures sequentiality whilst being "intensional"

* based on 2-player <u>games</u>:

  everything except the program

  - O (for "opponent") plays the role of the <u>context</u>
  - P (for "player") plays the role of the <u>program</u> being modelled

* a program is interpreted as a <u>strategy</u> that tells P how to respond to O's "moves"

  compositional (and indeed higher-order)

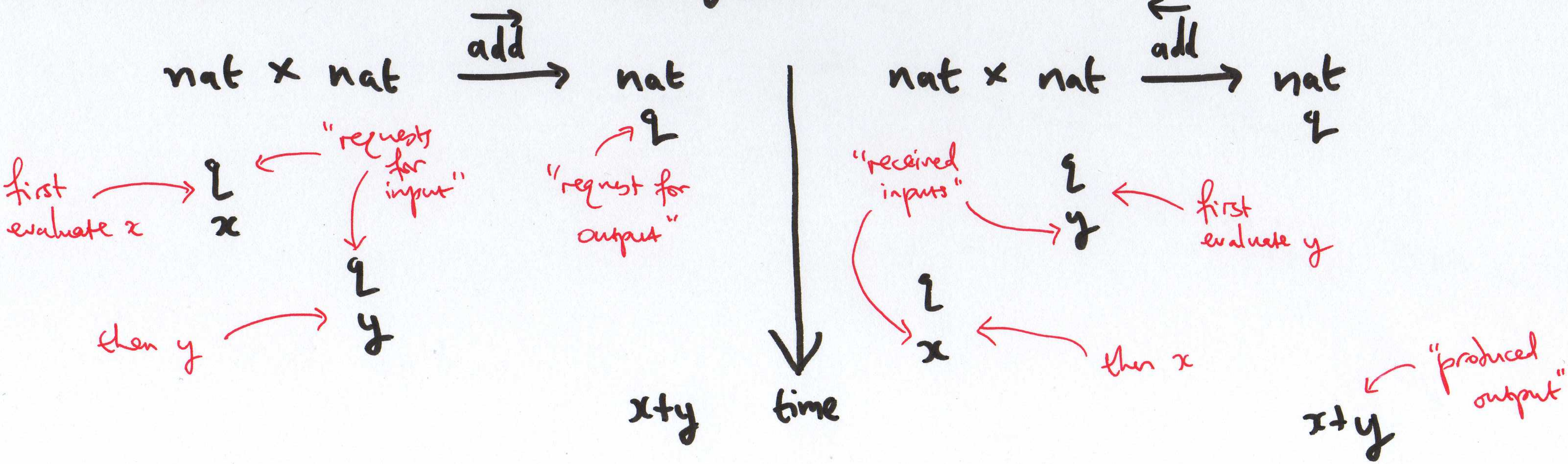
* strategies compose and form a category...

Intensional vs. Extensional models

* a program computing $x+y$ <u>must</u> evaluate both x and y …
  but can do this in either order

* in an extensional model, these two programs are interpreted
  by the <u>same</u> function: $x, y \mapsto x+y$

* an intensional model <u>distinguishes</u> them:



* So, in an extensional model, $\overrightarrow{add}$ and $\overleftarrow{add}$ are <u>necessarily</u> interpreted by the same arrow

* whereas, in an intensional model, they are distinguished

$\Longrightarrow$ $\boxed{\text{Is this good or bad?}}$

* if we're only interested in functional programming... maybe it's bad

* but, in more "powerful" languages, $\overrightarrow{add}$ and $\overleftarrow{add}$ can easily be distinguished... in ML:

$$\text{fn } f \Rightarrow f(\underbrace{x}_{\text{a ref}} := 0;\ 2,\ x := 1;\ 3);\ !x$$

* This is a <u>major</u> advantage of game semantics over domain models: to model "imperative" features such as references, exceptions, nondeterminism ...

domain models must be entirely <u>reworked</u>:

static handling

- "Continuation passing" — for "exceptions", jumps...
- "State passing" style — for references
- Combinations — continuation & state passing for ML-style exceptions

dynamic handling

* Game semantics, on the other hand, needs no such <u>global</u> transformation of the model: instead of requiring <u>every</u> function in the model to take additional parameters (one for the continuation, one for the current store, ...) we simply <u>relax</u> constraints on strategies

* different $\boxed{\text{combinations of constraints}}$ correspond to different "styles" of programming language, e.g.

  - <u>innocent</u> <u>and</u> <u>well-bracketed</u> = functional
  - <u>innocent</u> only = functional + continuation passing
  - <u>well-bracketed</u> only = functional + state passing

  etc...

* moreover, these correspondances are very tight: many <u>definability</u> results hold for game models

  - <u>every</u> innocent and well-bracketed strategy is the interpretation of some functional program
  - <u>every</u> innocent strategy interprets some functional program <u>using control operators</u> (call/cc...)
  - <u>every</u> well-bracketed strategy interprets some functional program <u>using references</u>

* a kind of semantic taxonomy of programming languages according to the <u>features</u> offered to the programmer

In this course...

* informal introduction to game semantics
* formal development of innocent strategies
    - models of PCF
    - models of PCF with control operators
* categories of innocent / innocent and well-bracketed strategies
* maybe more... depending on time...