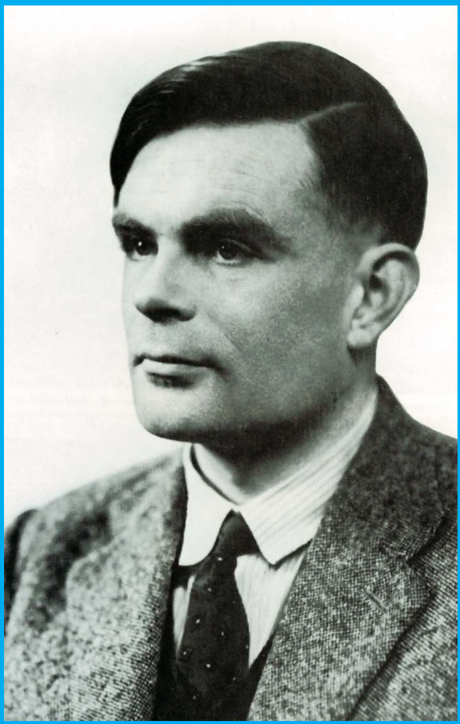
From Computation to Interaction: towards a science of information

Samson Abramsky

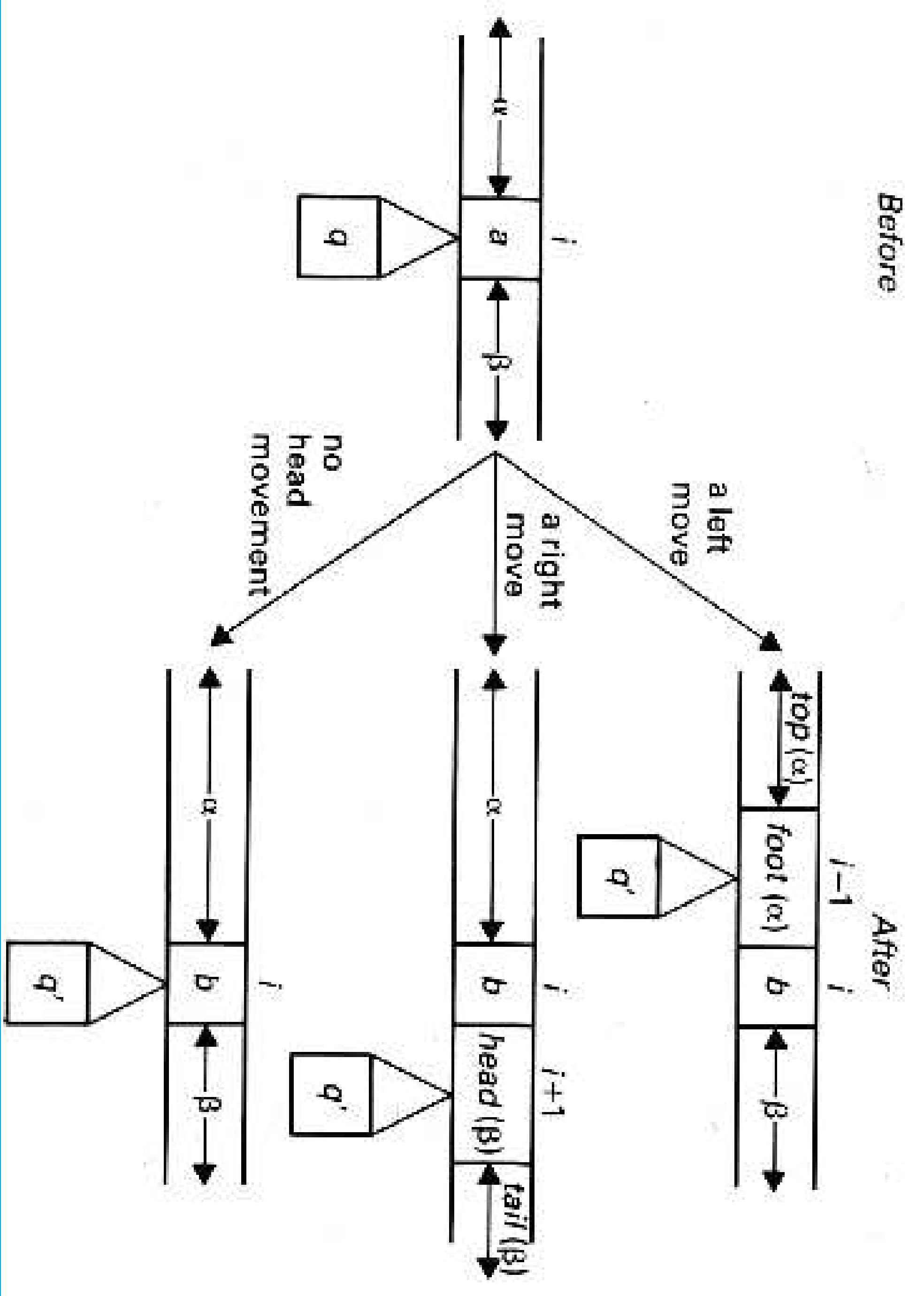
Alan Turing



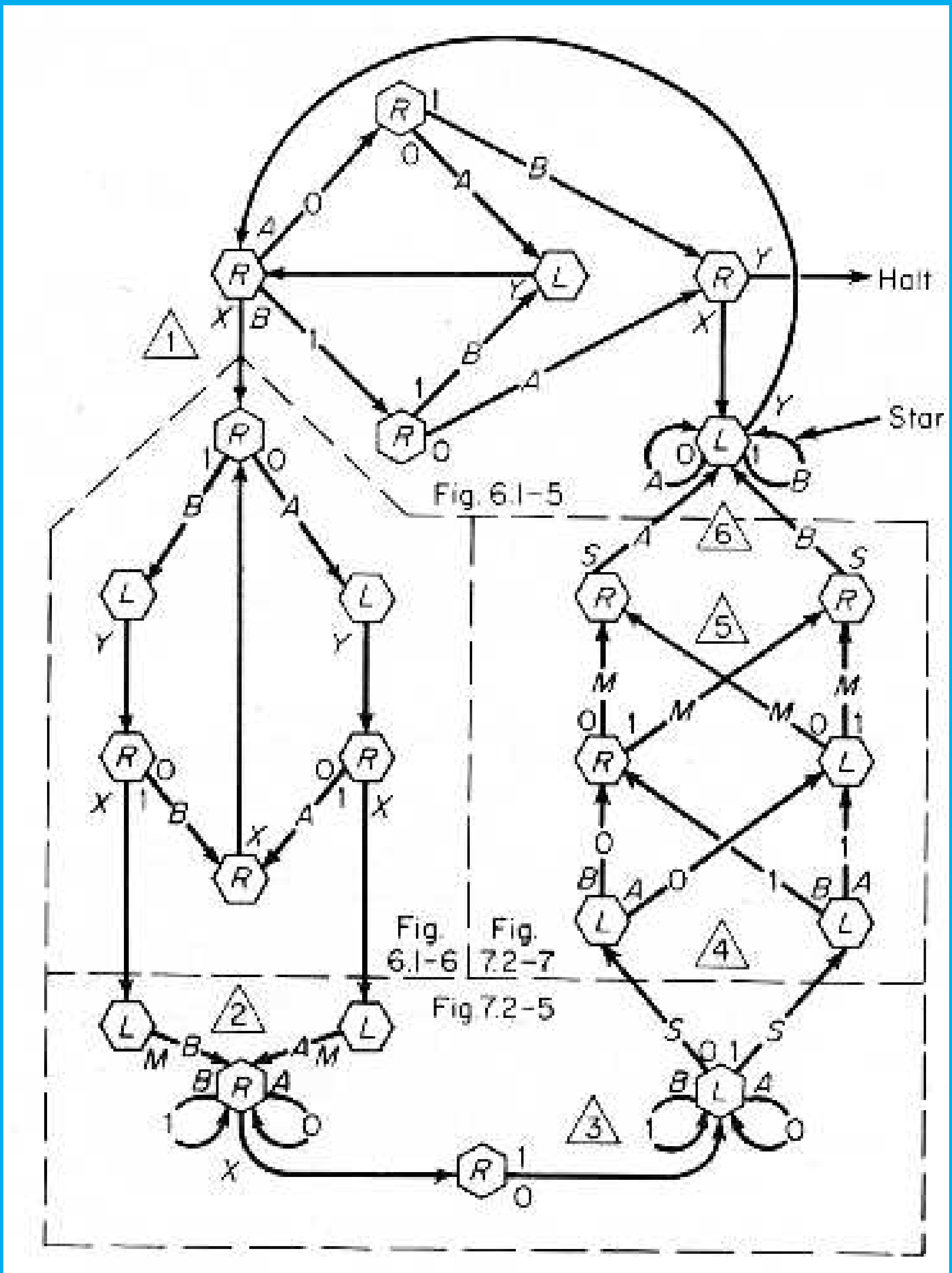
Pioneer of:
– mathematical theory of computation
– computer design and programming
– artificial intelligence
– mathematical theory of morphogenesis

Contributions to pure mathematics, logic.

Turing's model of computation



A universal Turing machine



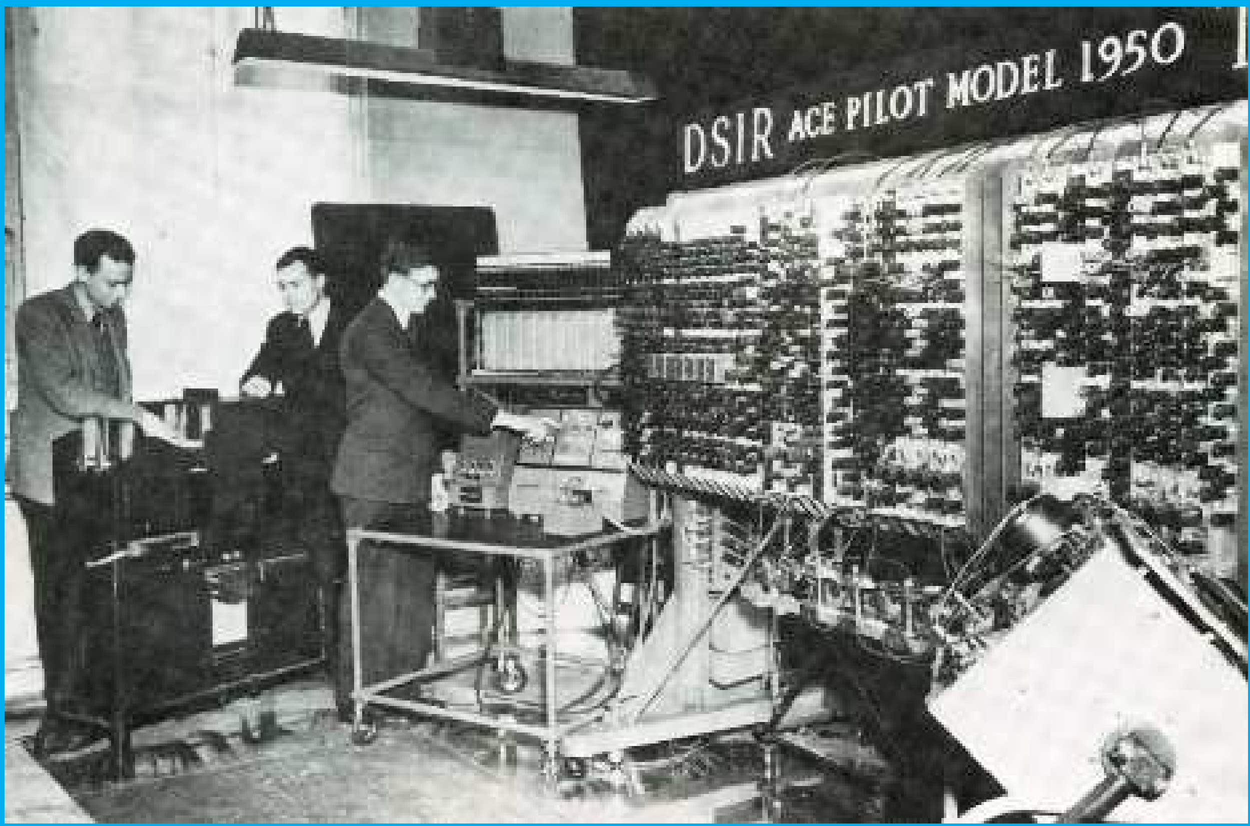
Key features

- a *convincing* conceptual analysis
- the computer performs *actions* in (discrete abstractions of) space and time – there is an explicit *process of computation*
- universal machines

To make all this *thinkable* at all back then:

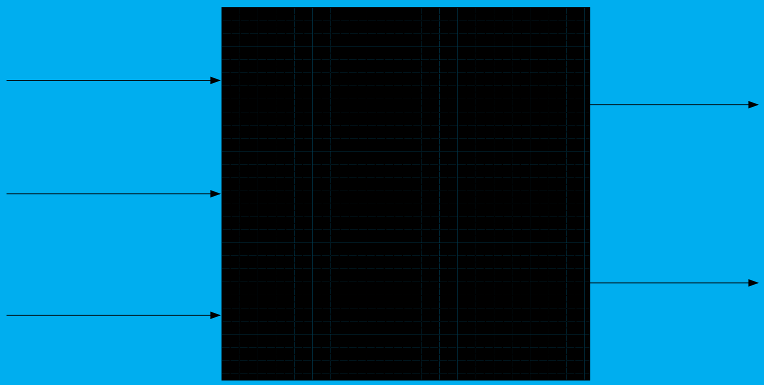
- Turing machines are sequential, monolithic, “unstructured”.

The early practice of computing



Stand-alone machines and programs:
computing "in the isolation ward".

First-generation models of computation

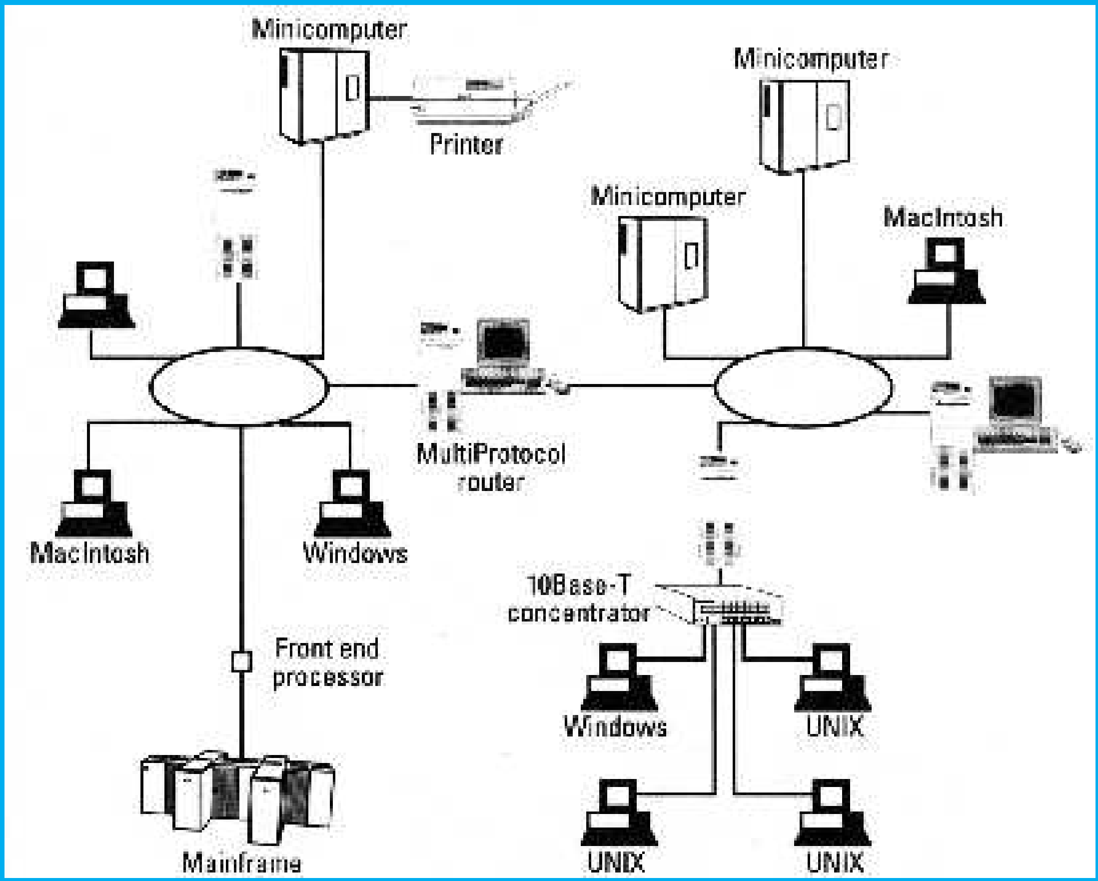


Programs are seen as computing *functions* or *relations* from inputs to outputs.

These models live on the intellectual inheritance from mathematics and logic (including, crucially, Turing's ideas).

Time, *processes* lurk in the background but are largely suppressed.

The technology changes



multitasking → distributed systems → Internet → “mobile” and “global” computing

Key features:
agents interacting with each other
information flowing around the system.

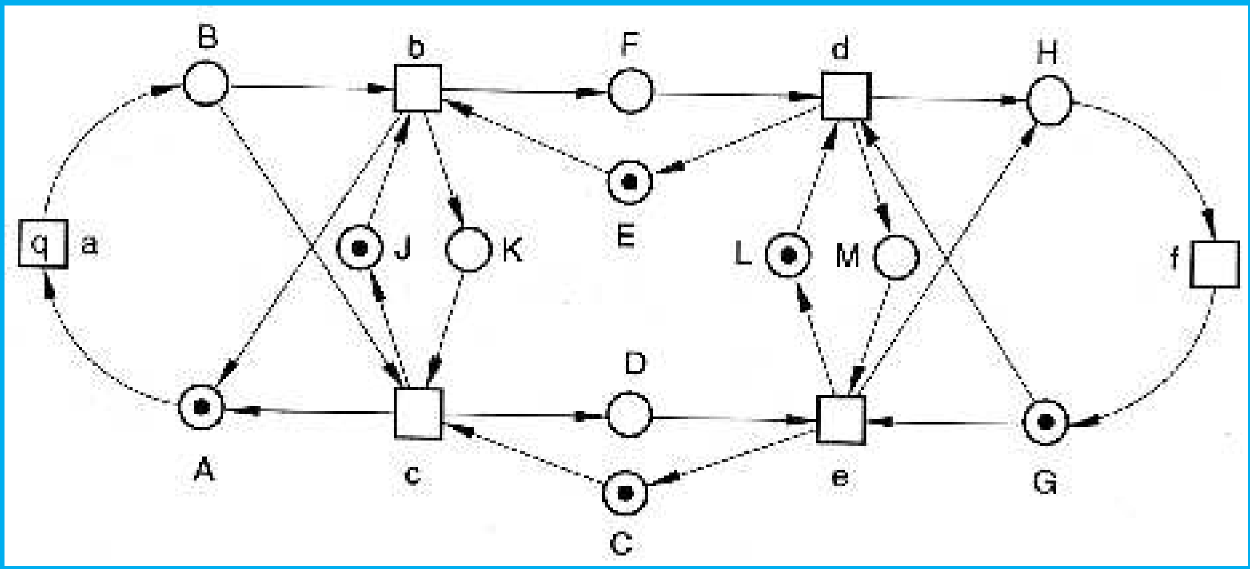
Insufficiency of the first-generation models

What does the Internet compute?
Surely not a mathematical function . . .

The old concepts fail to match the modern world of computing and its concerns: *robustness* in the presence of failures, *atomicity* of transactions, *security* of information flows, *quality* of user interface, *quantitative* aspects, . . .

Processes *vs.* **Products**:
How we compute *vs.* *What* we compute.
Processes were in the background, but now come to the fore: the "how" *becomes* the "what".

Second-generation models



Process models: Petri nets, CCS, CSP, . . .

Whereas 1st-generation models lived off the intellectual inheritance from mathematics and logic, there is no adequate pre-existing theory of *processes* or *agents*, *interaction*, *information flow*:

What is computed? What *is* a process? What are the analogues to Turing-completeness, universality?

A plethora of models, but no definitive conceptual analysis, comparable to Turing's analysis of computation in its "classical" sense.

Perhaps because it's a harder problem!

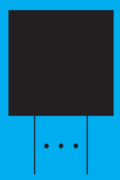
New Perspectives 1

Instead of isolated systems, with rudimentary interactions with their environment, the standard unit of description or design becomes a *process* or *agent*, the essence of whose behaviour is *how it interacts* with its environment.

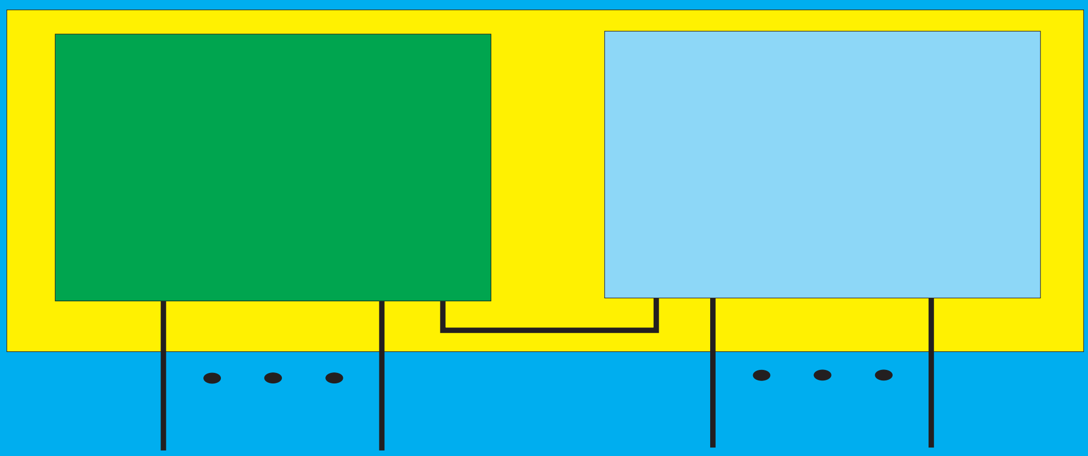
Who is the System? Who is the Environment? This *symmetry* will lead us to a key *duality*, and a deep connection to logic.

New Perspectives 2

Complex behaviour arises as the global effect of a *system* of interacting agents (or processes).
The key building block is the agent:



The key operation is *interaction* – plugging agents together so that they interact with each other



This model works at all "scales"

Macro-scale: processes in operating systems, software agents on the Internet, transactions. Micro-scale: how programs are implemented (subroutine call-return protocols, register transfer) all the way down into hardware.

Design (Synthesis) *vs.* **Description** (Analysis): artificial *vs.* natural information-processing systems.

Large issues: – *emergent behaviour* (intelligence?)
– understanding this complexity *compositionally*.

We need new conceptual tools, new theories, to help us analyze and synthesize these systems, to help us *understand* and *build*.

What is the "logic of interaction"

Specifying and reasoning about the behaviour of computer programs takes us into the realm of logic

For the 1st-generation models, logic could be taken "as it was"—static and timeless

For our 2nd-generation models, getting an adequate account—a genuine "logic of interaction"—may require a fundamental reconceptualization of logic itself.

This radical revision of our view of logic is happening anyway—prompted partly by the applications, and partly from ideas arising within pure logic.

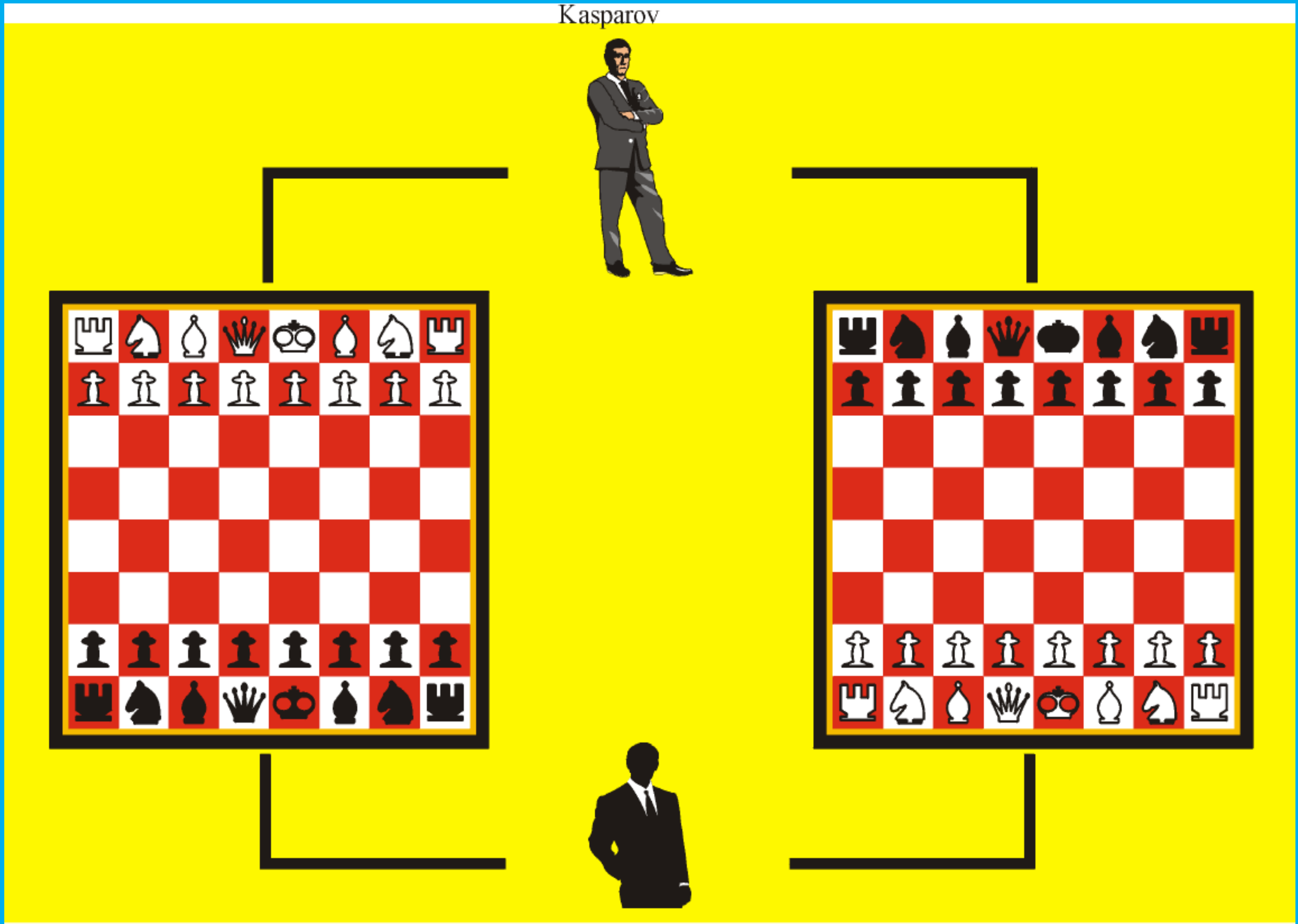
The static conception of logic

$$A \vee \neg A$$

"It is raining *or* it is not raining"—truth-functional semantics.
(*Tertium non datur?*)

Towards a dynamic conception of logic

How to beat an International Chess GrandMaster by the power of pure logic.

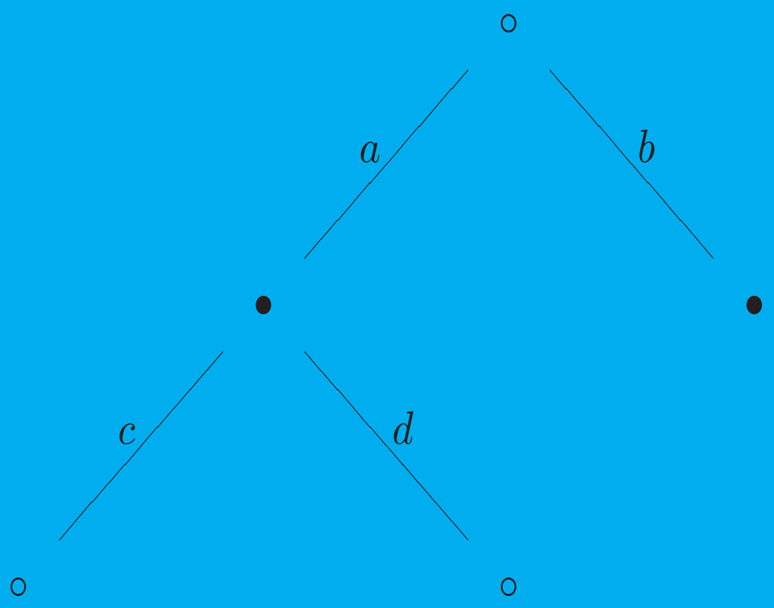


The copy-cat strategy

Game Semantics

Games as:
– interface types for computation modules
– propositions with dynamic content.

2-person games:
– Player *vs.* Opponent
– System *vs.* Environment.



Agents are strategies

In this setting, we model our agents or processes as *strategies* for playing the game.

These strategies *interact* by playing against each other.

We obtain a notion of correctness which is *logical* in character in terms of the idea of *winning* strategy—one which is guaranteed to reach a successful outcome however the environment behaves.

This in a sense replaces (or better, *refines*) the logical notion of "truth": winning strategies are our dynamic version of tautologies (more accurately, of *proofs*).

Building complex systems by combining games

We shall now see how games can be combined to produce more complex behaviours while retaining control over the interface.

This provides a basis for the *compositional* understanding of our systems of interacting agents—understanding the behaviour of a complex system in terms of the behaviour of its parts.

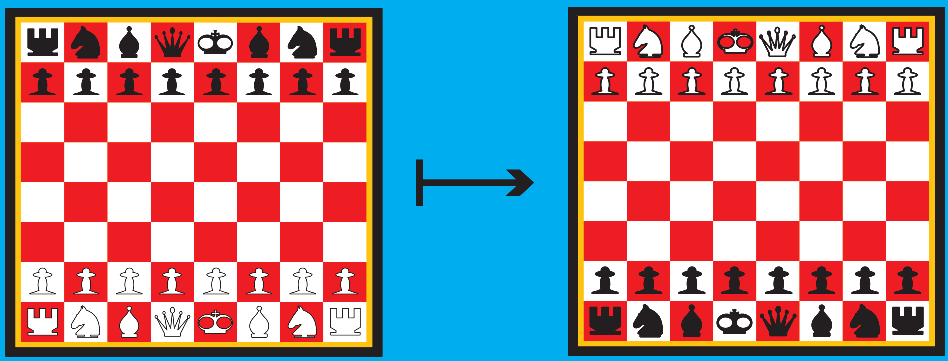
This is crucial for both analysis and synthesis, *i.e.* for both description and design.

These operations for building games can be seen as (dynamic forms of) "type constructors" or "logical connectives".

(The underlying logic will in fact be Girard's Linear Logic).

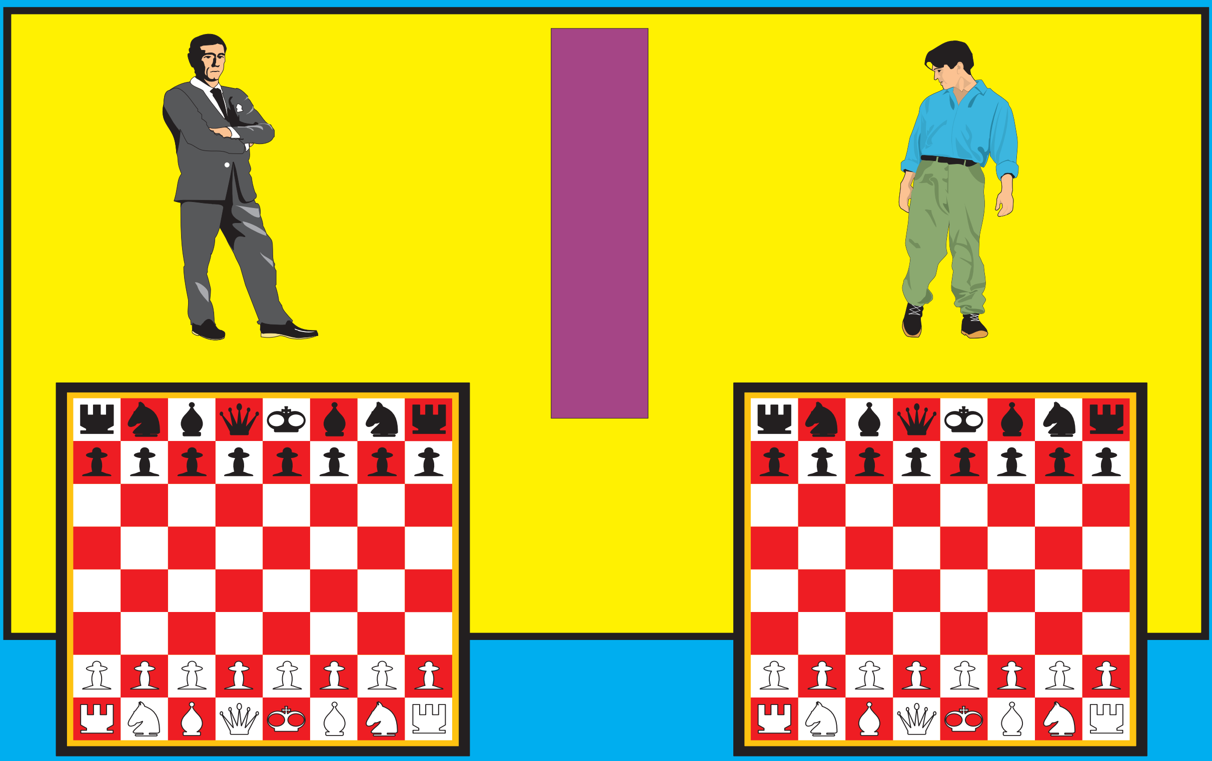
Duality—“Linear Negation”

$A^\perp$ — interchange rôles of Player and Opponent (reflecting the symmetry of interaction).



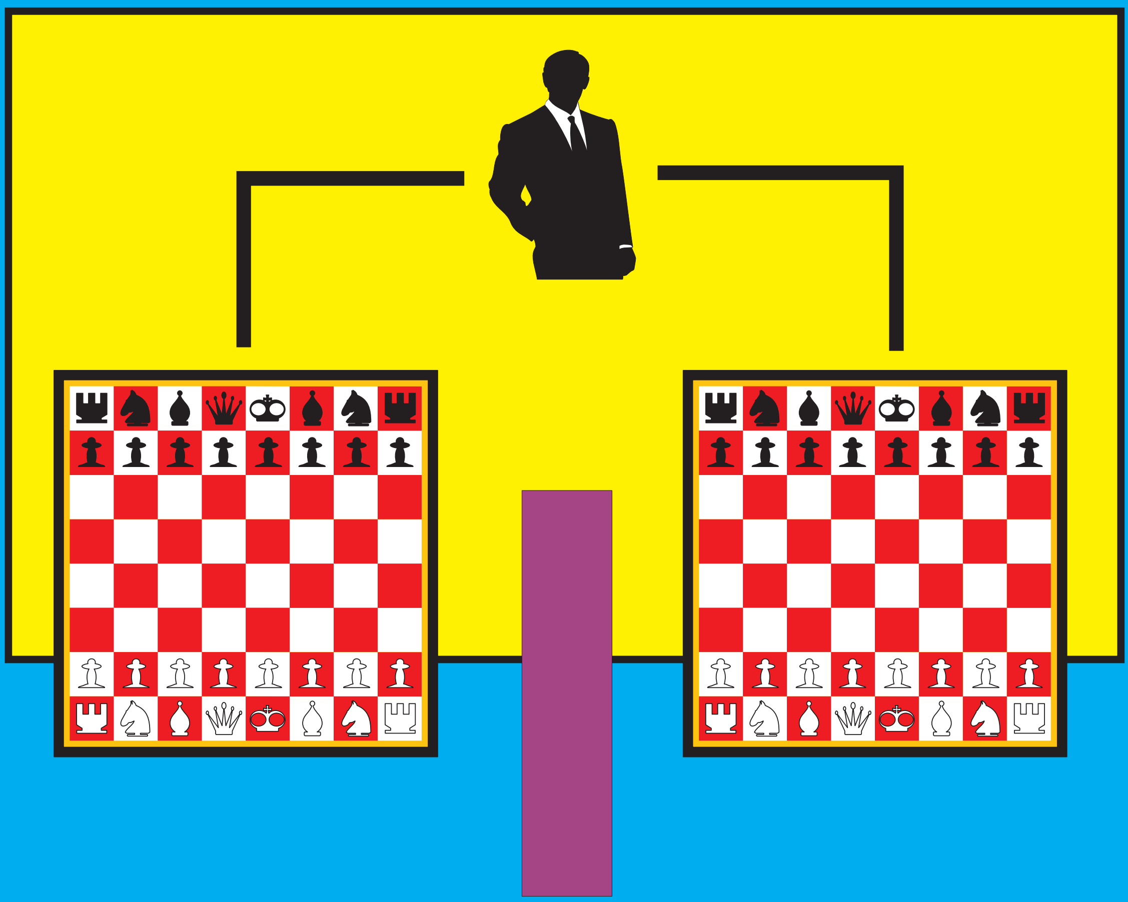
$$A^{\perp\perp} = A.$$

Tensor — “Linear conjunction”



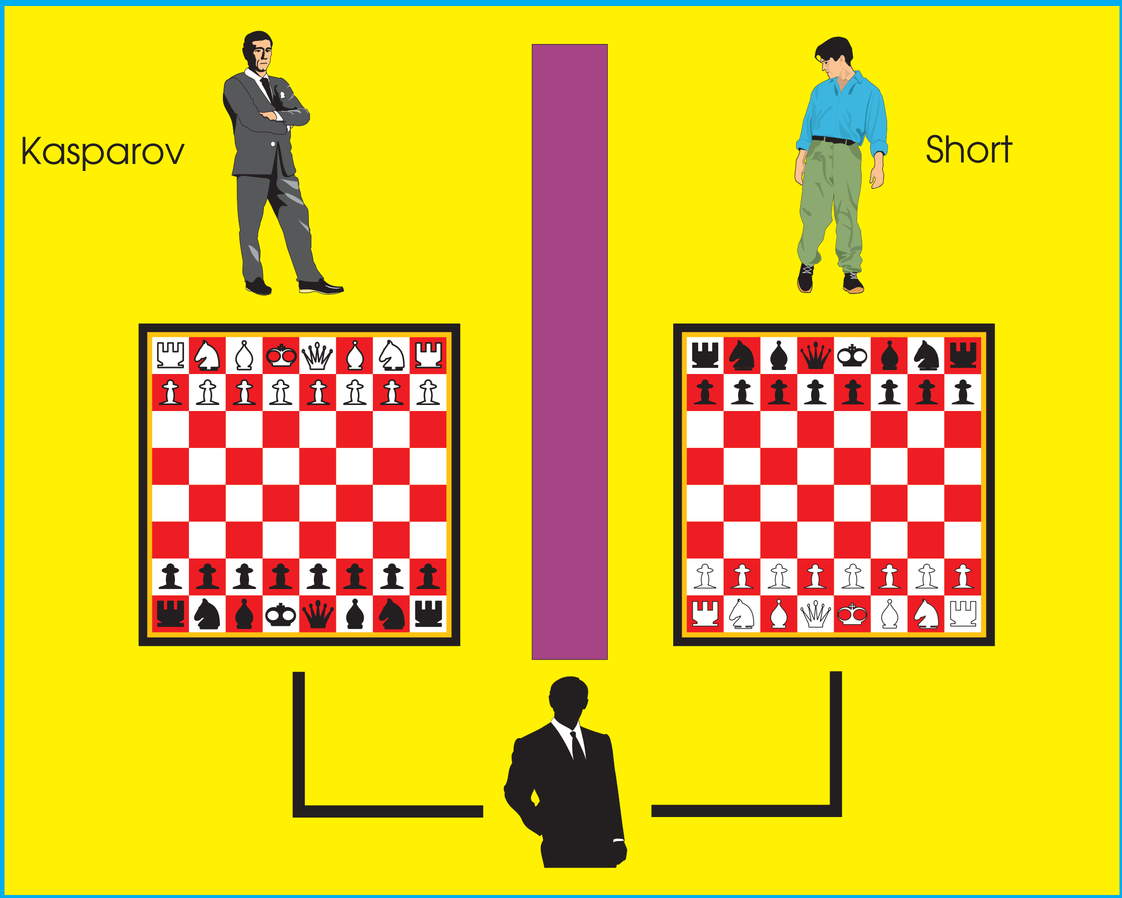
$$A \otimes B$$

Par — "Linear disjunction"



$$A \wp B$$

The Copy-cat Strategy



Logical structure

We can define $A \multimap B$ ("Linear implication") by

$$A \multimap B \;\equiv\; A^{\perp} \wp B$$

$$(cf.\;\; A \supset B \;\equiv\; \neg A \vee B.)$$

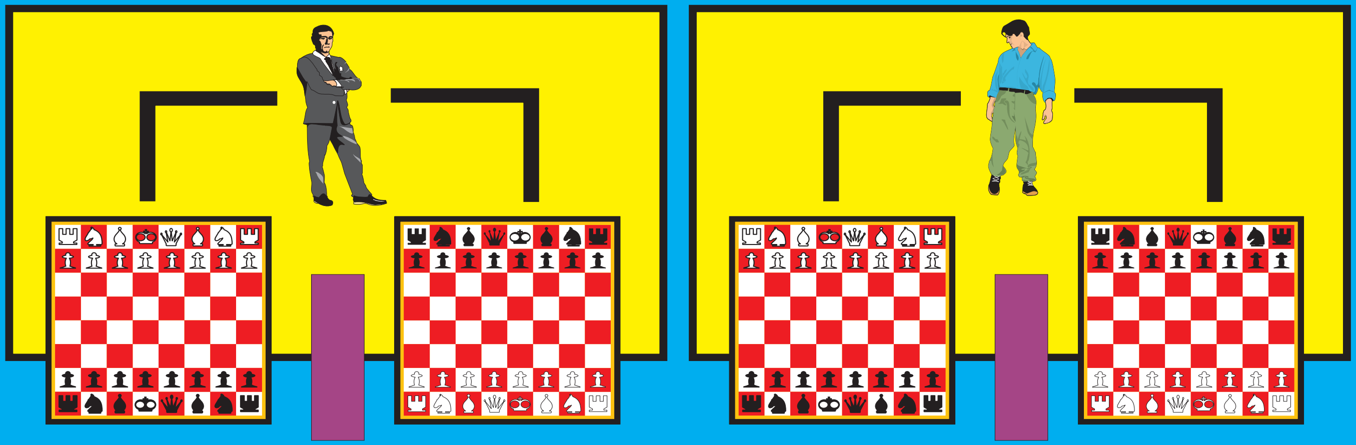
The Copy-cat strategy is a winning strategy for $A^{\perp} \wp A \;\equiv\; A \multimap A$.

De Morgan duality:

$$\begin{aligned} (A \otimes B)^{\perp} &= A^{\perp} \wp B^{\perp} \\ (A \wp B)^{\perp} &= A^{\perp} \otimes B^{\perp} \end{aligned}$$

Interaction

Constructors create "potentials" for interaction; the operation of plugging modules together so that they can communicate releases this potential into actual computation.



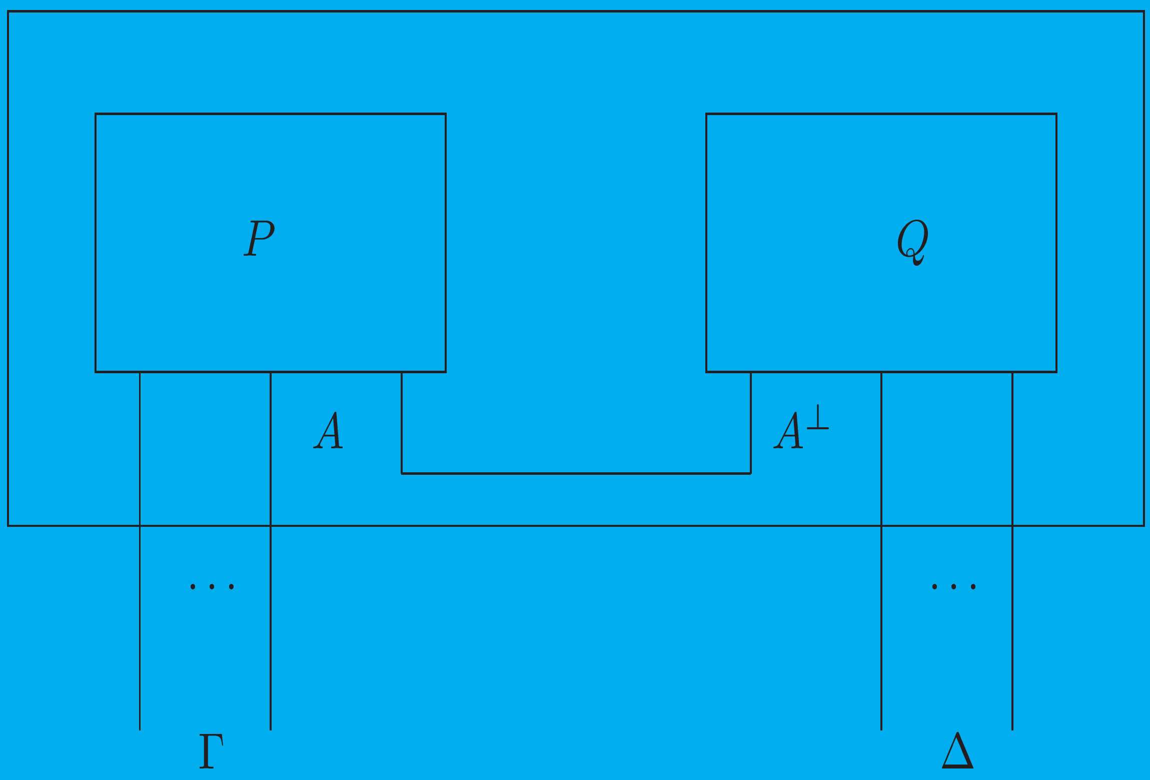
$$A^{\perp} \wp B \;\equiv\; A \multimap B \qquad B^{\perp} \wp C \;\equiv\; B \multimap C$$

Composition

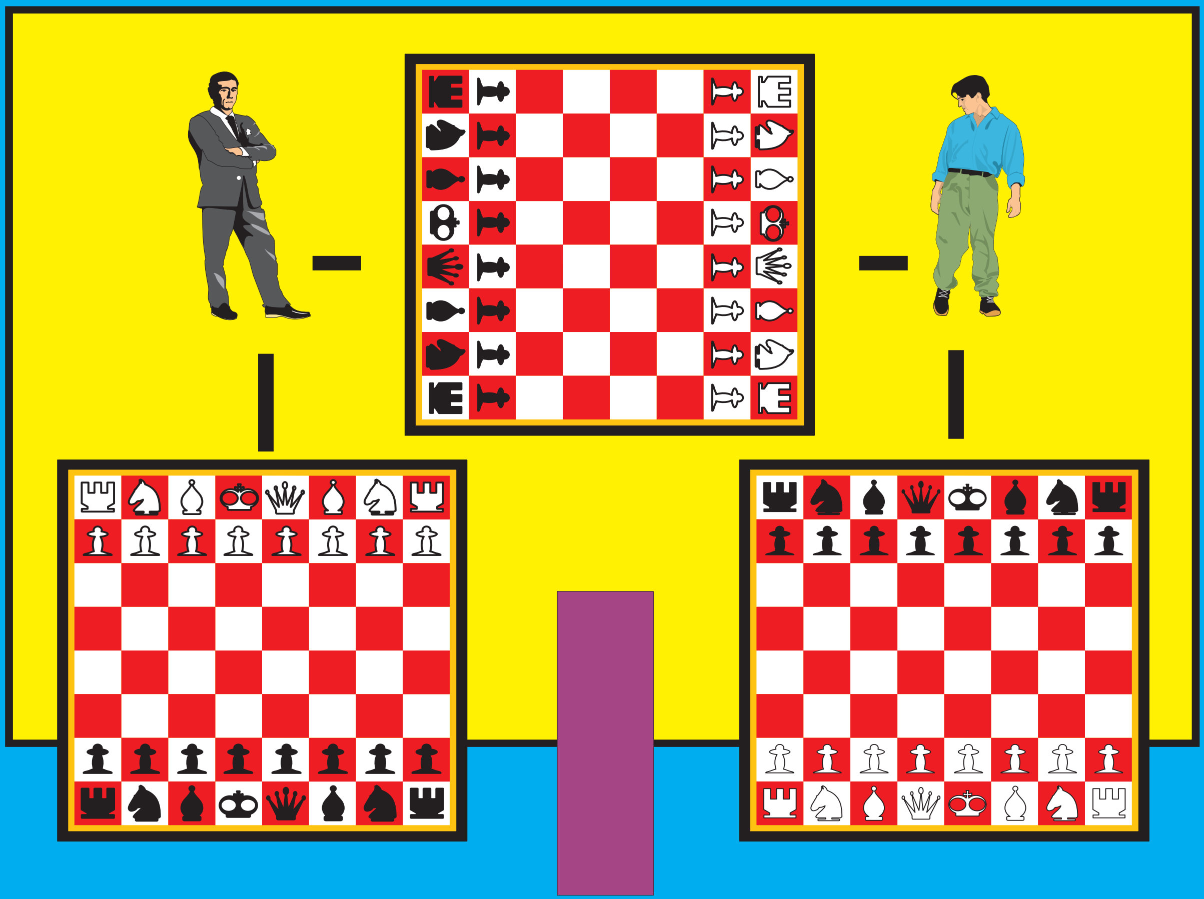
$$\frac{A \xrightarrow{\text{Gary}} B \xrightarrow{\text{Nigel}} C}{A \xrightarrow{\text{Gary; Nigel}} C}$$

Cut:

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, \perp A}{\Gamma, \Delta}$$

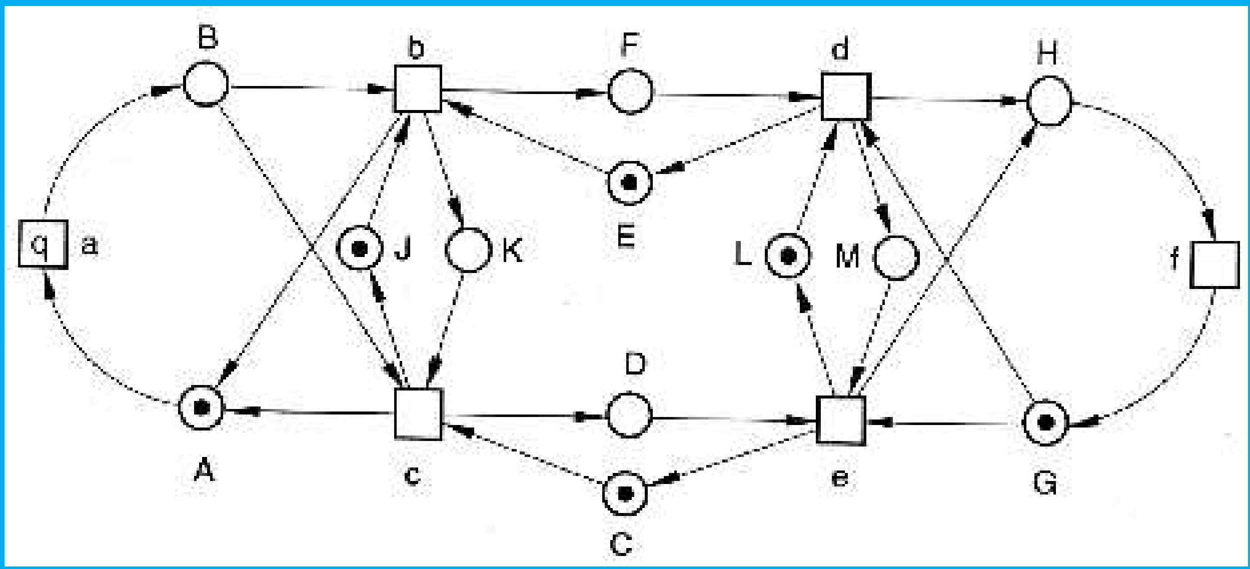


The Interaction Game



Caveat pre-emptor 1: concurrency

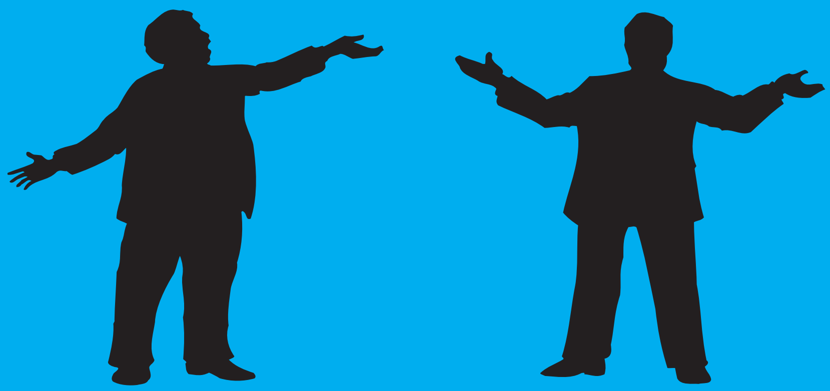
Game *trees* are sequential, too rigid. There is now a good theory of *concurrent games*; roughly, these are played on Petri Nets instead of trees.



Locally, each choice is made by one player or the other, but globally there are two distributed "teams" of players, acting concurrently. This sounds complicated, but there is a very elegant formalization.

Caveat pre-emptor 2: multi-party interactions

How many does it take to interact?—Two is the critical number!



0 Players	:	(truth-) values
1 Player	:	actions
2 Players	:	interaction

Multi-party interactions can be reduced to the 2-Player case with suitable *types* (just as many-place functions can be reduced to one-place using products).

Game Semantics: 1992–...

Builds on previous work, but new insights, new possibilities, new connections, leading to a (still emerging) new synthesis.

Achievements: Foundational analysis of programming languages and logics.
– "Completing Strachey's programme": fully abstract models for wide-spectrum modern programming languages, e.g. ML; *articulating* the "space of programming languages".
– Some striking new developments in logic ($e.g.$ "full completeness").

Applications: – Program analysis, certifying secure information flows (Hankin, Malacaria).
– Specifying reactive systems, Internet agents.
– Natural language: branching quantifiers, information dependencies, dialogues.

Further Vistas 1

Computability revisited:

We know (since Turing) what *computable functions* are, (but indirectly—probably necessarily so).

Can we capture—in a "machine-independent" way—what *computational processes* are, *without* referring back to Turing machines or any other explicit machine model?

Reasons for optimism: using game semantics, it has been possible to give such characterizations of the "shapes" of computations carried out within certain *programming disciplines*:

–purely functional programming

–stateful programming

–programming with non-local jumps (*e.g.* exceptions).

Further Vistas 2

Relating quantitative and qualitative views of computation and information

Information theory: players move in a game by *choosing* between alternatives; hence first moves and then positions can be *quantified* in terms of their information content.

Can we quantify *information flow between agents*?

Complexity theory: in the same spirit as for computability, can we characterize $e.g.$ polynomial-time computation in a *machine-independent* way?

(Towards) Informatics

The "strong Informatics thesis": there is a *unified science of information* embracing both "artificial" and "natural" computation.

The ideas we have presented (and related ideas due to many people) lend substance to this view by treating *description* and *design* in a unified way.

These ideas are "in the air". The University of Edinburgh has (last year) created a Division of Informatics, explicitly based on this "strong thesis", incorporating the former departments of CS, AI and Cog Sci. There are similar developments afoot in a number of other institutions.

This can be seen as further evidence for our claim that computation has overflowed its traditional boundaries.

On a personal note:

Robin Gandy 1919–1995

Gandy was a living link to Turing (as well as a very distinguished logician in his own right)

Turing's qualities

Endgame

```
Game Over
>
```