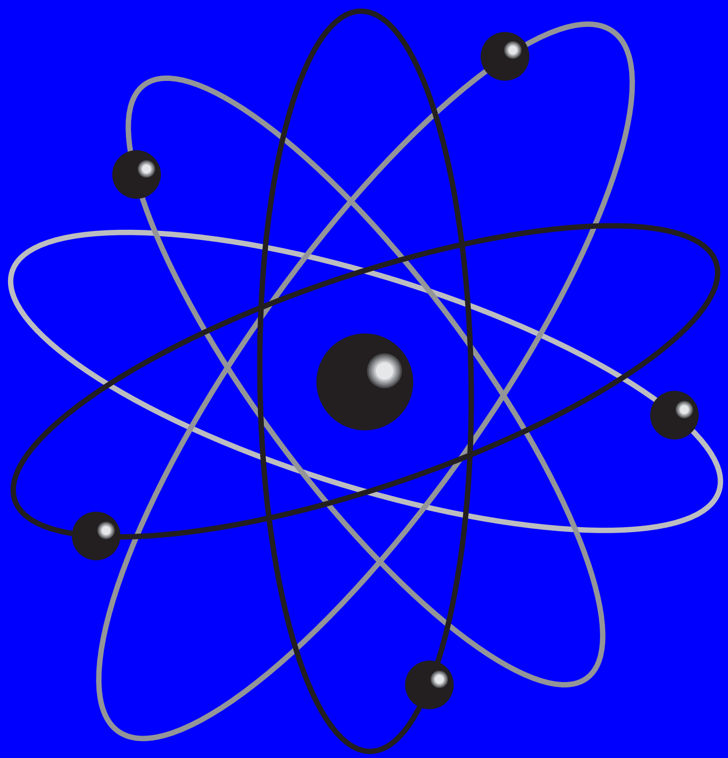
Interaction: splitting the atom of computation



What is Computer Science?

- Science? - where are the experiments?
- Engineering? - what is the stuff we engineer?
- Mathematics? - how does it relate to the applications?
- Information Technology? - whither the intellectual capital?

The virtue of this uncertainty: Computer Science may be something new in the intellectual firmament, and may even come to change our views of the pre-existing disciplines.

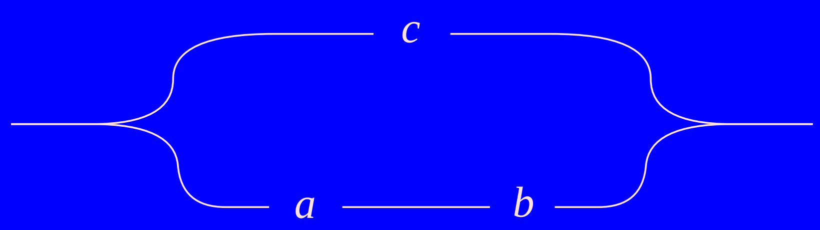
One answer to our question: the science of computation is logic.

Strong thesis ("computational logicism"): *floruit c.* 1985.

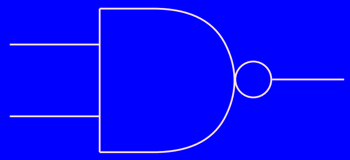
- Japanese Fifth Generation project
- The Alvey Programme.

Some milestones:

- Shannon's identification of boolean algebra and switching circuits (1938).



- Combinational logic circuits.



Functional Programming

Computation = simplification of an expression to make its value explicit:

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 1 \\ f(n+2) &= f(n) + f(n+1) \end{aligned}$$

$$\begin{aligned} f(4) &= f(2) + f(3) = f(0) + f(1) + f(1) + f(2) \\ &= 1 + 1 + 1 + f(0) + f(1) \\ &= 1 + 1 + 1 + 1 + 1 \\ &= 5 \end{aligned}$$

Logic Programming

Computation = search for proof of query from the program:

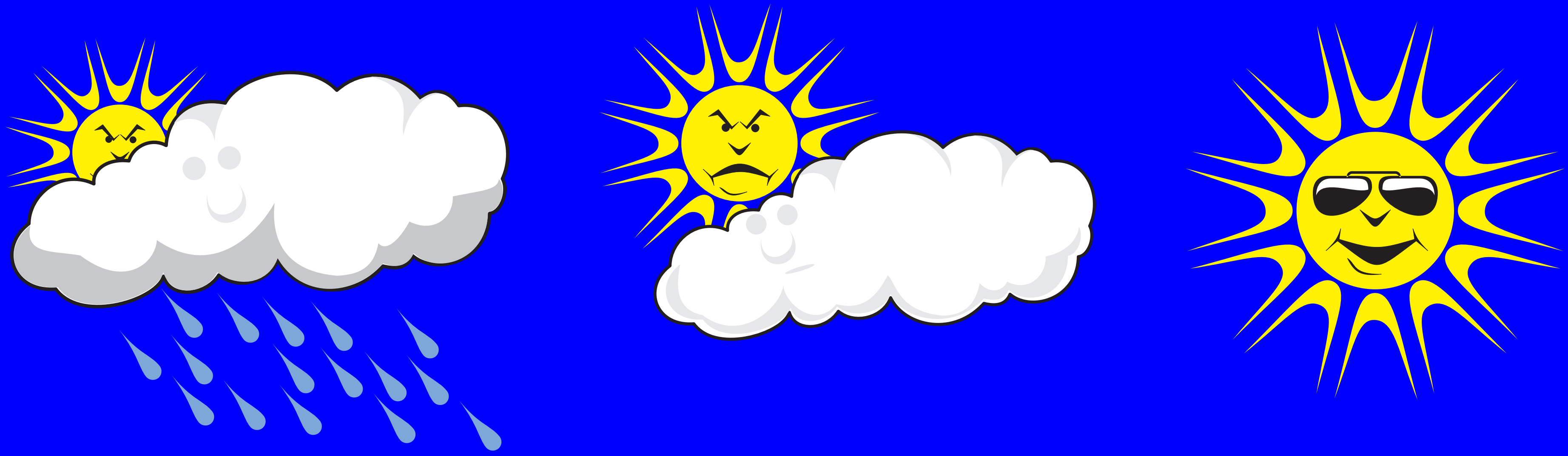
$$\begin{array}{ll} \texttt{Man(Socrates)} & :- \; . \\ \texttt{Mortal(x)} & :- \; \texttt{Man(x)}. \end{array}$$

$? \, \texttt{Mortal(x)}$

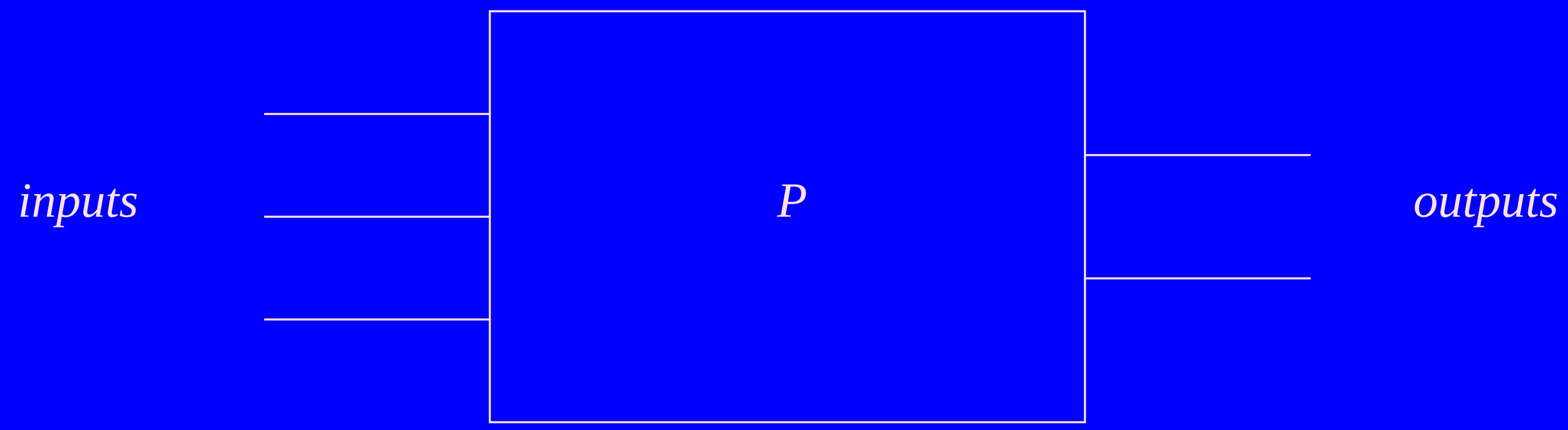
$> \; \texttt{x} = \texttt{Socrates}$

But, what is logic?

Static conception of logic:
"It is raining *or* it is not raining".
Truth-functional semantics
(*Tertium non datur?*)



Static conception of the semantics of computation
Programs as functions/relations.



Insufficiency of the static conception

Computation consists in large part of *practical syllogisms* (*cf.* Aristotle)

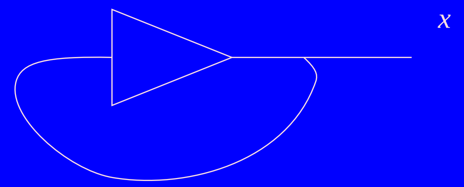
$$\frac{A_1, \ldots, A_n}{\text{perform action}}$$

This dynamic aspect, the interweaving of reasoning and action, is not adequately catered for by the static conception of logic.

(i) Distributed Systems

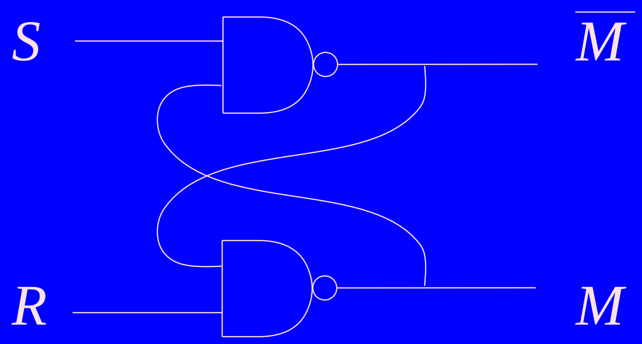
- The application *is* the behaviour and the quality of the interface.
- What is input and what is output - who is the System and who is the environment - become relative to point of view.
- Computation *is* interaction.

(ii) Feedback, recursion, strange loops (*cf.* the death of logicism): Paradox, what paradox?



$$x = \sim x$$

Feedback is the basis of memory, state, history:



M is 1 if the last input set to 1 was S (Set); 0 if the last input set to 1 was R (Reset).

(iii) The *oddness* of Classical (truth-functional) Logic

John weighs 100 pounds
Mary weighs 100 pounds
The maximum capacity of the lift is 150 pounds

$$
\begin{aligned}
A &\equiv \text{John is in the lift} \\
B &\equiv \text{Mary is in the lift} \\
C &\equiv \text{The maximum capacity of the lift is exceeded}
\end{aligned}
$$

Then $A \wedge B \supset C$ is true, but *neither* $A \supset C$ *nor* $B \supset C$ are true.

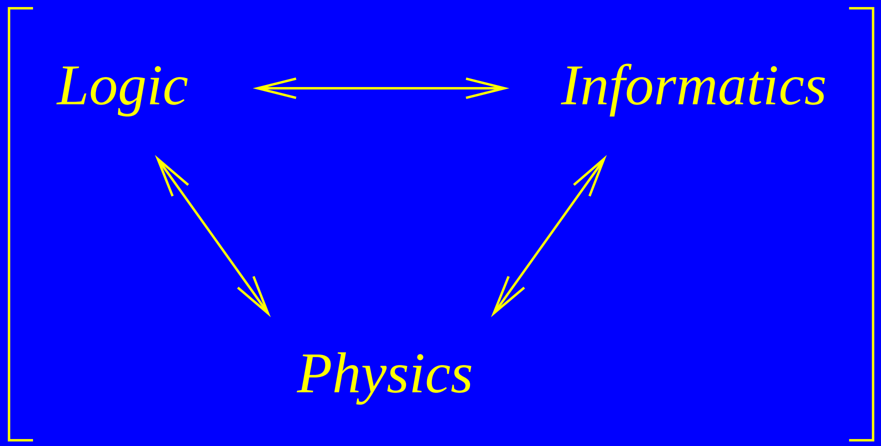
But we have the classical tautology:

$$(A \wedge B \supset C) \supset [(A \supset C) \vee (B \supset C)]!$$

Of course it is possible to revise the static conception of logic by introducing time and change explicitly into the "universe of discourse", as in modal and temporal logic.

But to retrieve a tighter link between logic and computation, and to obtain a deeper analysis, we need a more radical revision of our conception of logic.

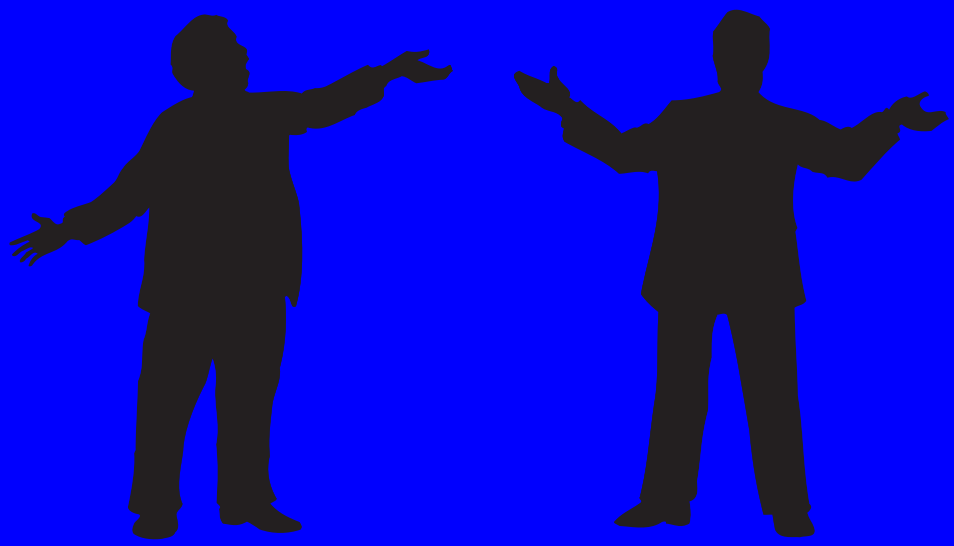
From statics to dynamics



The atom of logic - *truth*
The atom of computation - *values* (or *results* or *input-output relations*).

Splitting the atom to open up the microstructure within.
What articulates that structure? - *interaction*.

How many does it take to interact?



0	Players	:	(truth-) values
1	Player	:	actions
2	Players	:	interaction = dialogue = debate
>2	Players	:	can be reduced to the 2-Player case with suitable types (just as many-place functions can be reduced to one-place).

The dialogical view of logic (Lorenzen *et al.*)

Proponent *asserts* a thesis

Opponent *attacks* it

Proof = winning strategy for Proponent.

P	|	$A \vee B$
O	|	?
P	|	chooses A or B to defend
O	|	attacks the chosen formula

P	|	$A \wedge B$
O	|	chooses A or B to attack
P	|	must defend
	|	the chosen formula

P	|	$\exists x.\, A(x)$
O	|	?
P	|	chooses witness t
O	|	must attack $A(t)$

P	|	$\forall x.\, A(x)$
O	|	chooses witness t
P	|	must defend $A(t)$

P	$A \supset B$	P	$\neg A$
O	?	O	?
P	can defend by attacking A	P	can defend by attacking A
O	must defend A		
	once O has proved A,		
	P must defend B		

Once O has asserted a basic sentence, P can also assert it.

"Logical truths" are those for which Proponent has a winning strategy, regardless of the contents of the basic sentences.

Logic now plays the rôle of giving structural invariants of the information flows of certain *interactive processes* (strategies).

(i) Identity Axiom (or identity *function*)

$$\begin{array}{c|ccc} P & A & \supset & A \\ O & & ? & \\ P & ? & & \\ O & A & & \\ P & & & A \end{array}$$

P "copies" O's defense of A on the "input" (antecedent) side to make his own defense of A on the "output" (consequent) side.

Application to chess: how to beat an International Grand-Master …

(ii) Modus Ponens (or function application)

$$
\begin{array}{c|cccccccc}
P & [(A & \supset & B) & \wedge & A] & \supset & B \\
O & & & & & & ? & \\
P & & ? & & & & & \\
O & ? & & & & & & \\
P & & & & & ? & & \\
O & & & & & A & & \\
P & A & & & & & & \\
O & & & B & & & & \\
P & & & & & & & B
\end{array}
$$

This is a combination of two "copy-cat" strategies.
Logic is in the world and acting, rather than merely *talking about* the world of action.

(iii) Classical Logic: the Excluded Middle

$$
\begin{array}{c|ccccc}
P & A & \vee & (\neg & A) \\
O & & ? & & \\
P & & & \neg & A \\
O & & & ? & \\
P & & & & ? \\
O & & & & A \\
P & A & & &
\end{array}
$$

Note that P is being allowed to *change his mind* about which disjunct to assert. If we required P to make his choice in advance and stick to it, we would validate *constructive* rather than *classical* reasoning.

A static tautology has been converted into a *process* with a logical property (being a winning strategy for a certain dialogue game).

(iv) Classical Logic: non-local jumps or context-switching

Note "choices" for P (non-determinism of classical cut-elimination procedure).
Classical logic is computationally or dynamically "wilder" than constructive logic.

P	$(A$	$\wedge$	B	$\supset$	$C)$	$\supset$	$[(A$	$\supset$	$C)$	$\vee$	$(B$	$\supset$	$C)]$
O						$?$							
P				$?$									
O	$?$												
P							A	$\supset$	C				
O								$?$					
P							$?$						
O							A						
P	A												
O			$?$										
P											B	$\supset$	C
O												$?$	
P											$?$		
O											B		
P			B										
O					C								
P									C				C

(v) More context-switching: the well-ordering principle for natural numbers.

$$
\begin{array}{l|llll}
P & \forall X \subset \mathcal{N}. & & [X = \emptyset \vee (\exists x \in X.\ \forall y \in X.\ x \leq y)] & \\
O & X & = \{...\} & & \\
P & & & X = \emptyset & \\
O & X & = \{5, ...\} & & \\
P & & & & \forall y \in X.\ \ 5 \leq y \\
O & X & = \{5, 2, ...\} & & \\
P & & & & \forall y \in X.\ \ 2 \leq y \\
O & X & = \{5, 2, 1, ...\} & & \\
P & & & & \forall y \in X.\ \ 1 \leq y
\end{array}
$$

The fact that P's strategy is winning is *equivalent* to the well-ordering principle.

What we have gained is a *strategy*, a *process*, to give this fact a dynamic embodiment.

Some further developments

The Characterization Problem:

What class of abstract objects is definable in a given formal system? (Delineate the class by "structural" (syntax-independent) means.)

Classical examples The completeness theorem for first-order logic.
Turing universality (assuming Church's thesis).

Classical problems Characterizing classes of recursive functions intrinsically (Gödel's problem).
Characterizing (and hence separating!) complexity classes, e.g. $\mathbf{P} \neq \mathbf{NP}$.

In semantics of computation "Full abstraction" problems.

The dynamic view of logic and computation, in the form of *game semantics*, has allowed significant progress on such characterization problems, both for logics and for programming languages.

What we characterize are now the *strategies* for dialogues rather than the abstract objects (sets, functions etc.) which these strategies "realize".

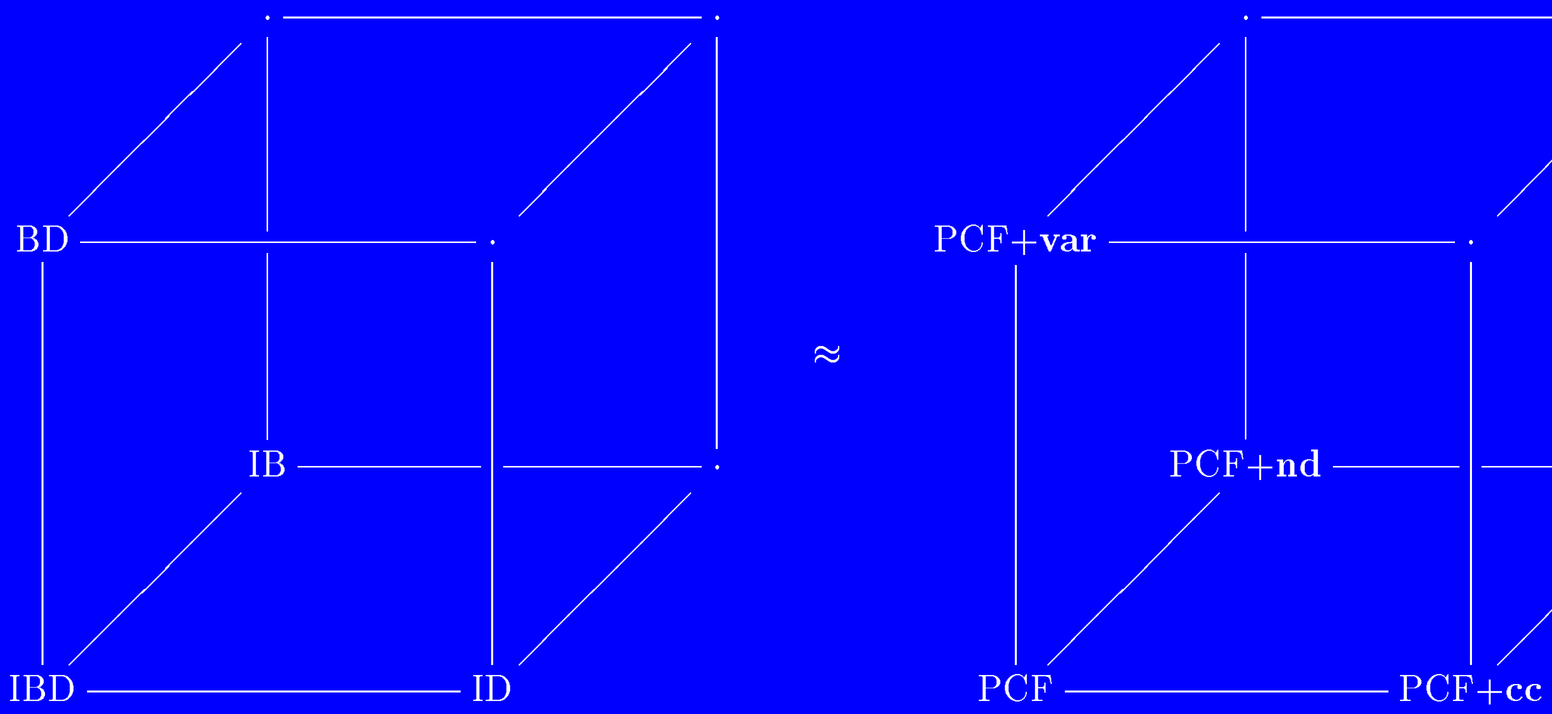
We can give structural properties on strategies - how "well-behaved" they are required to be in terms of how they conduct their dialogues, *e.g.* whether context-switching is allowed, which pick out *precisely* which strategies correspond to:

- intuitionistic proofs
- classical proofs
- pure functional programs
- functional programs with local state
- functional programs with non-local control operators (*e.g.* *context-switching*)

Moreover, the various classes of strategies are well related to each other by means of "factorization" properties and other mathematical structure.

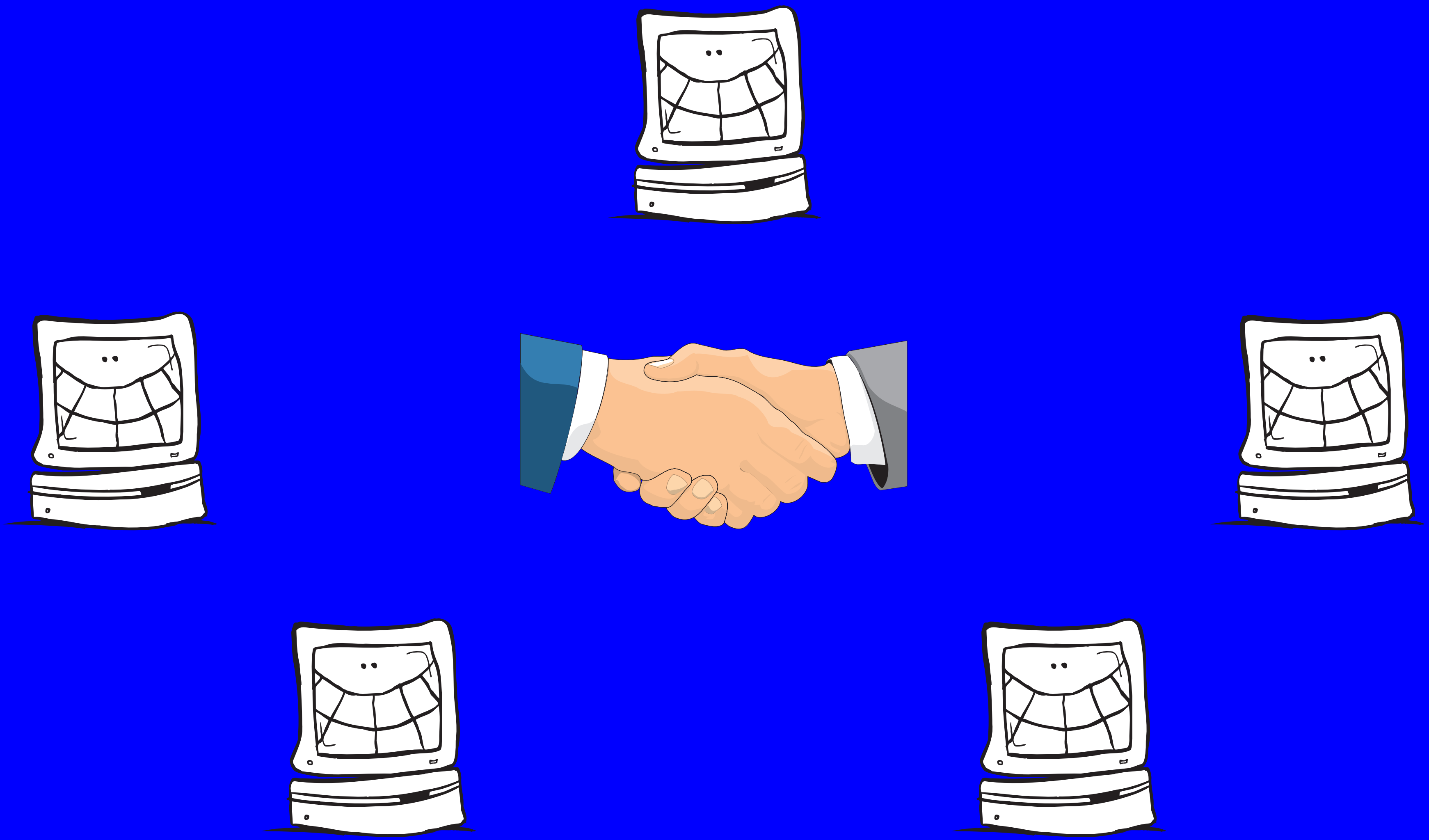
In this way, a coherent map of the semantic space of computation is *beginning* to emerge.

The “semantic cube”



Towards Applications

- Logical principles for structuring dialogues/protocols/interactions.
- Semantics for concurrent object-oriented languages (such as JAVA).
- Type systems to aid in achieving security, robustness and efficient compilation.



In this way, a coherent map of the semantic space of computation is *beginning* to emerge.

The "semantic cube"