
HUNT FOR BIN LADEN
CNN
EXPERTS AGREE: AL QAEDA
LEADER IS DEAD OR ALIVE
NAS
42.40
PHILIPS

What is an algorithm?

Don Knuth (1966)



Algorithms are concepts which have existence apart from any programming language... Algorithms were present long before Turing et al. formulated them, just as the concept of the number "two" was in existence long before the writers of first grade textbooks and other mathematical logicians gave it a certain precise definition.

A Neolithic Algorithm





Eve's Algorithm

- If something's left and it's my turn
  - Put one in my pile
  - Now it's his turn
- If something's left and it's his turn
  - Put one in his pile
  - Now it's my turn

An algorithm is a discrete process whose evolution has a finite description

Euclid (c. -300)



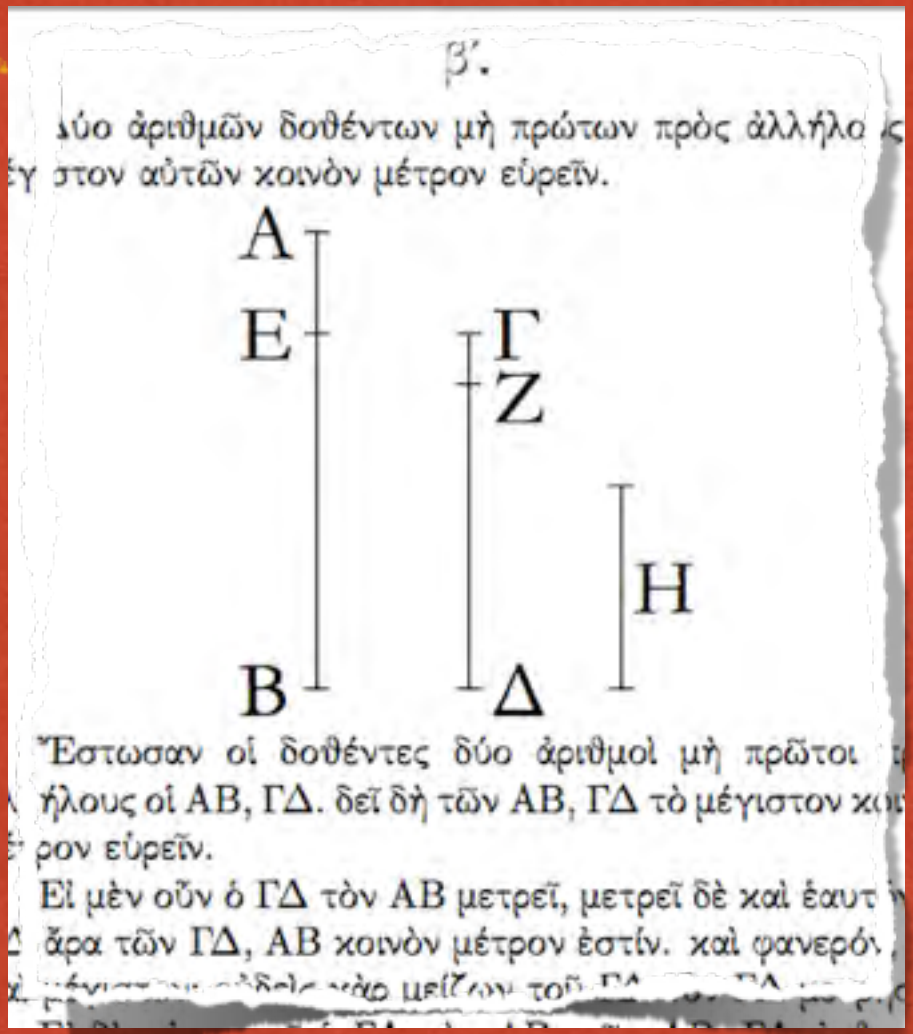
Euclid's GCD algorithm was formulated geometrically:

Used repeated subtraction of the shorter segment from the longer.

Euclid's Elements

- Finitely describable – in terms of basic compass operations





Antenaresis

Δύο ἀριθμῶν ἀνίσων ἐκκειμένων, ἀνθυφαιρουμένου δὲ ἀεὶ τοῦ ἐλάσσονος ἀπὸ τοῦ μείζονος, ἐὰν ὁ λειπόμενος μηδέποτε καταμετρῇ τὸν πρὸ ἑαυτοῦ, ἕως οὗ λειφθῇ μονάς, οἱ ἐξ ἀρχῆς ἀριθμοὶ πρῶτοι πρὸς ἀλλήλους ἔσονται

When two unequal numbers are set out, and the less is continually subtracted in turn from the greater, if the number which is left never measures the one before it until a unit is left, then the original numbers are relatively prime.

Euclid's Computer

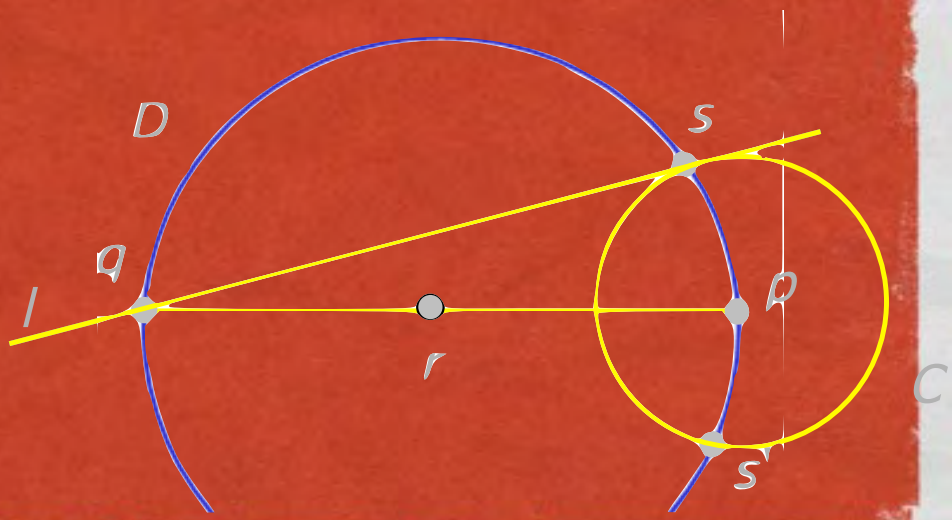
If $q \notin C$ then

r := bisect(p,q)

D := Circle(r,q)

s :∈ C ∩ D

out := Line(q,s)

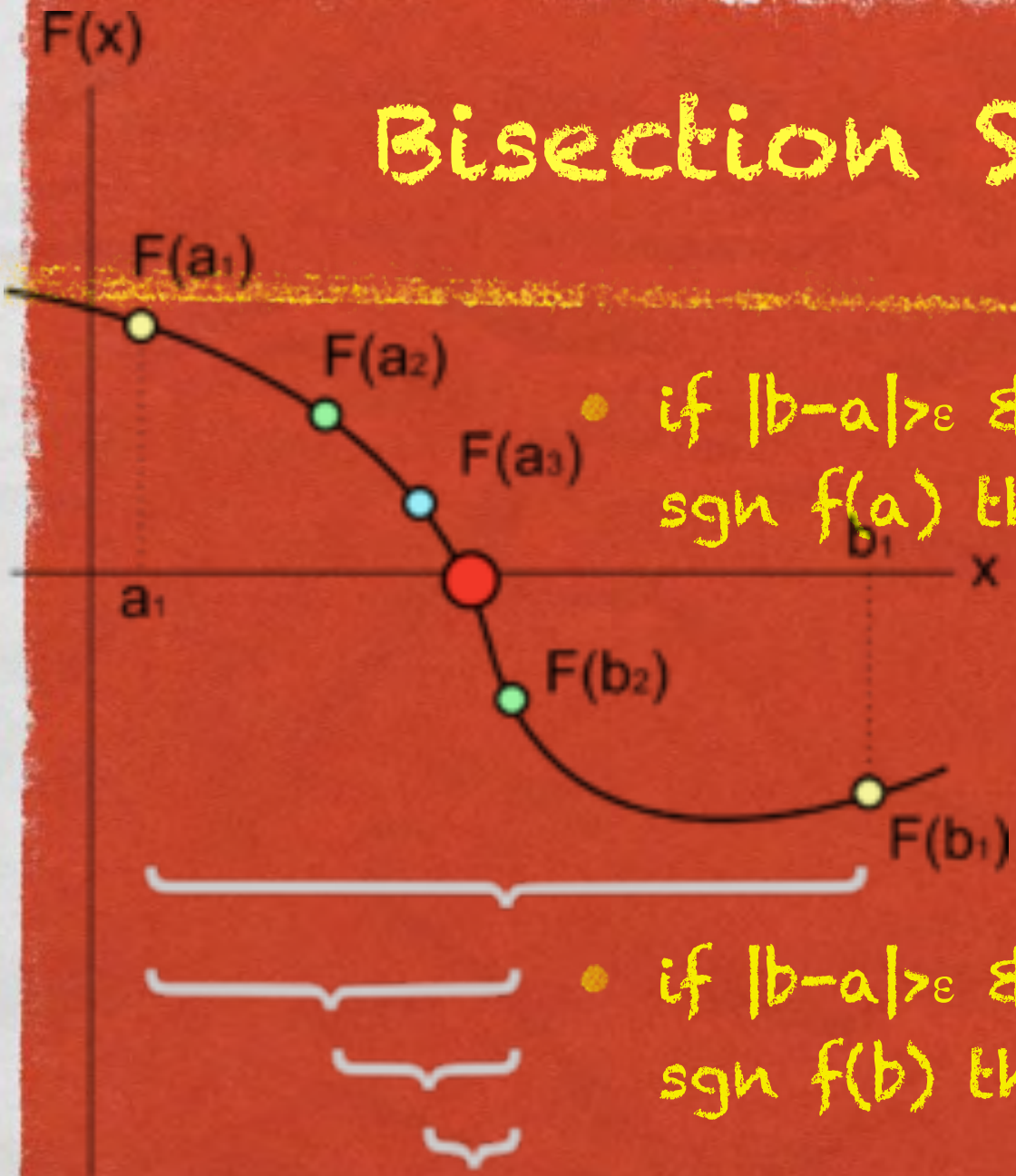


Algorithm #4 (1960)

- This procedure ... finds a root by iterated bisection and evaluation at the midpoint, halting if ... two successive approximations of the root differ by less than ε, [which] must not be less than 2 units in the last place carried by the machine or else indefinite cycling will occur due to roundoff.



Bisection Search



- if |b−a|>ε & sgn f((a+b)/2) = sgn f(a) then a := (a+b)/2

- if |b−a|>ε & sgn f((a+b)/2) = sgn f(b) then b := (a+b)/2

Bisection Search (Saul Gorn)

- Although [this procedure is] among the slowest, it is applicable to any continuous function. The fact that no differentiability conditions have to be checked makes it ... an 'old work-horse'.



Hartley Rogers

- Roughly speaking, an algorithm is a clerical (i.e., deterministic, bookkeeping) procedure which can be applied to any of a certain class of symbolic inputs and which will eventually yield, for each such input, a corresponding symbolic output.

- An example of an algorithm is the usual procedure given in elementary calculus for differentiating polynomials...



Postulates of Algorithms

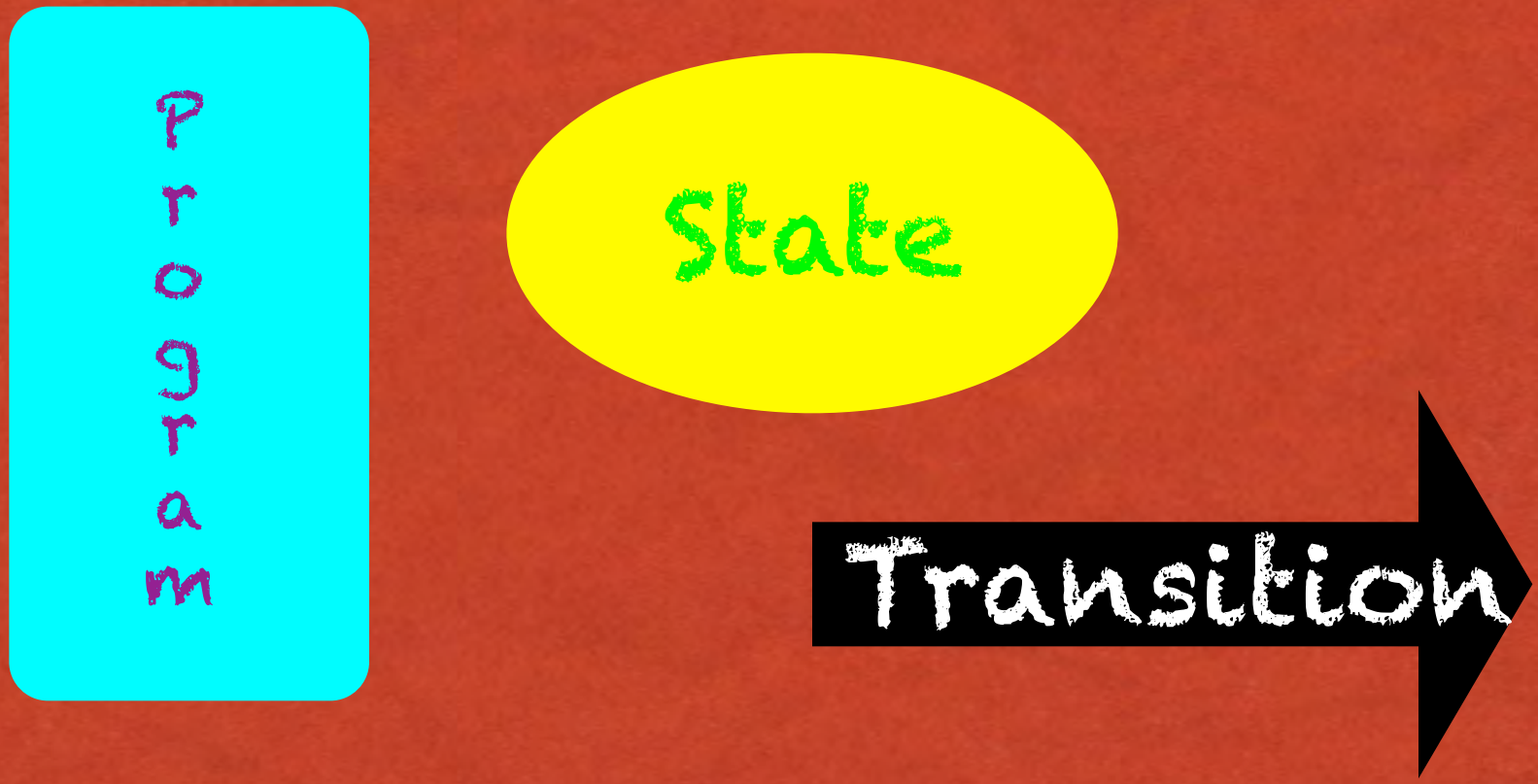
I. A classical algorithm is a discrete state-transition system

II. Logical structures capture salient aspects of its states

III. The transition relation can be described finitely

I: State Transition System

Tripartite Notion

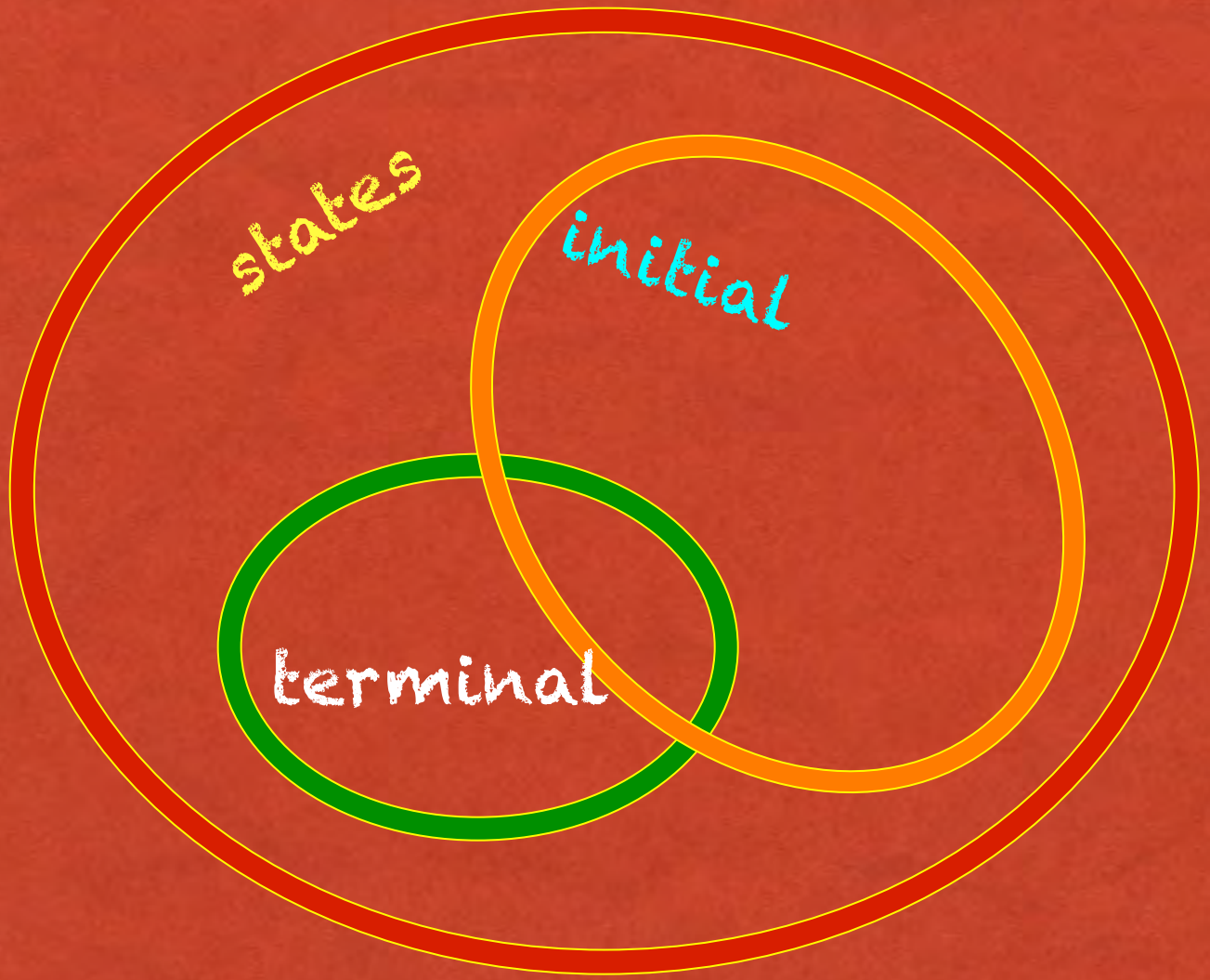


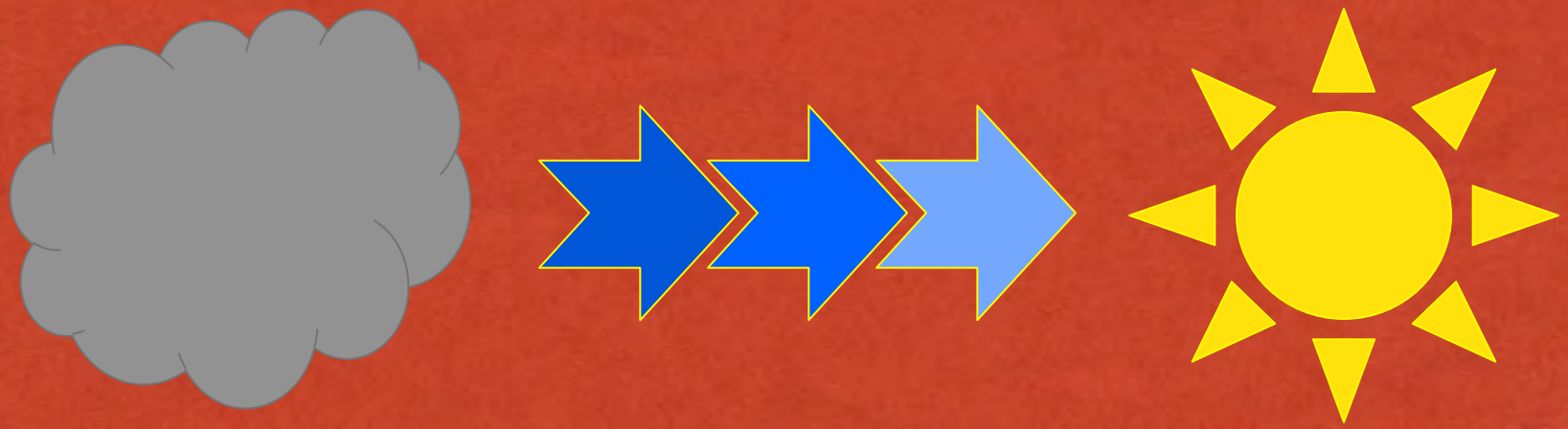
Knuth (1966)



A computational method comprises a set of states...

In this way we can divorce abstract algorithms from particular programs that represent them.


states
initial
terminal



Hilary Putnam's Limit



Hermann Weyl's Zeno Machines



John von Neumann

- Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.





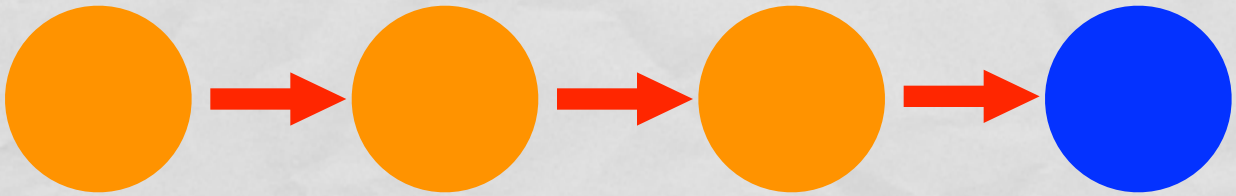
II: States as Structures





State encapsulates all relevant data!

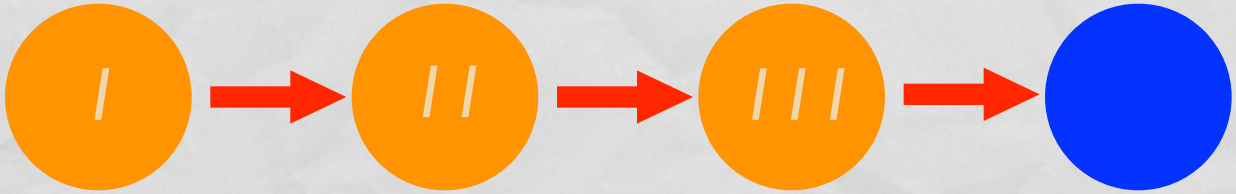
Tabula Rasa



Tabula Rasa



Tabula Rasa



States

- Everything needed to proceed (besides the algorithm)



Intermediate States

Ce genre de Peinture ... de pouvoir être interrompu quand on veut & repris de même

Paul Romain Chaperon,
Traité de la peinture au pastel (1788)









Data Structures

- Church uses terms
- Kleene uses numbers
- Turing uses strings
- Programmers use all sorts of data structures

Emil Post (1922)

- We ... assume [symbol-complexes] to be finite and ... discrete...
- [They] are completely determined by specifying all the properties and relations of its parts... Each ... can be completely described [by a conjunction of relations].



Structures/Algebras

- Domain (universe, base set, carrier)
- Vocabulary (function symbols, relations [truth-valued functions], scalars [constants])
- Operations (interpreting the symbols)

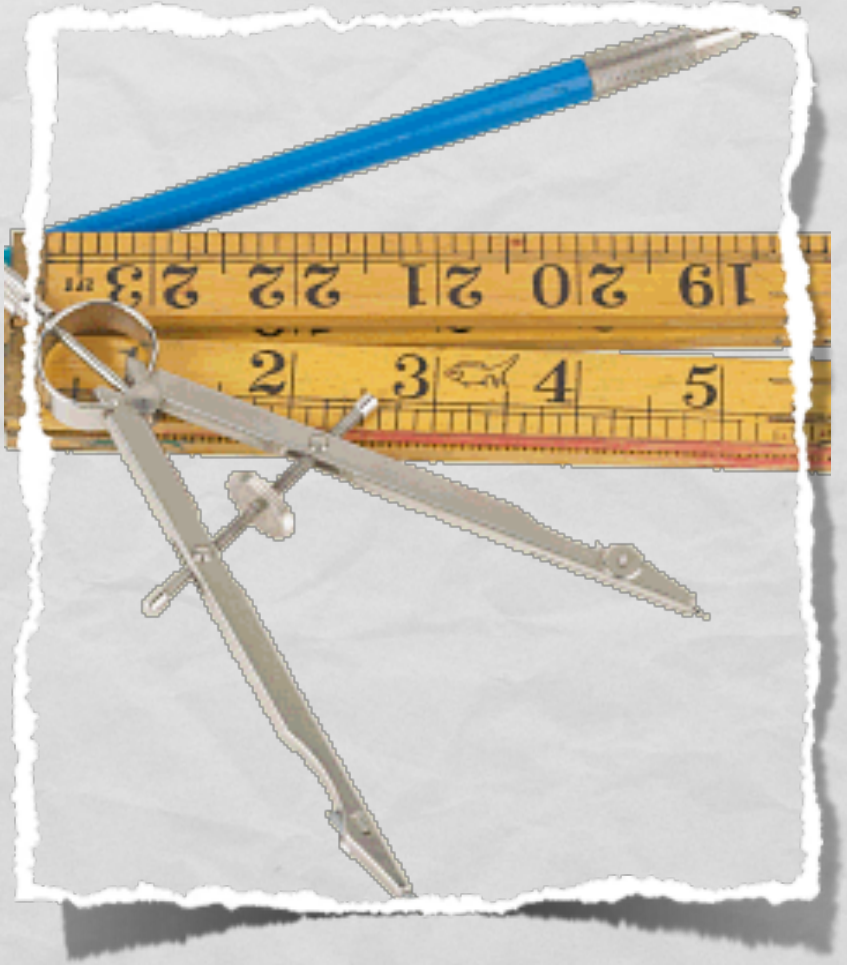
States

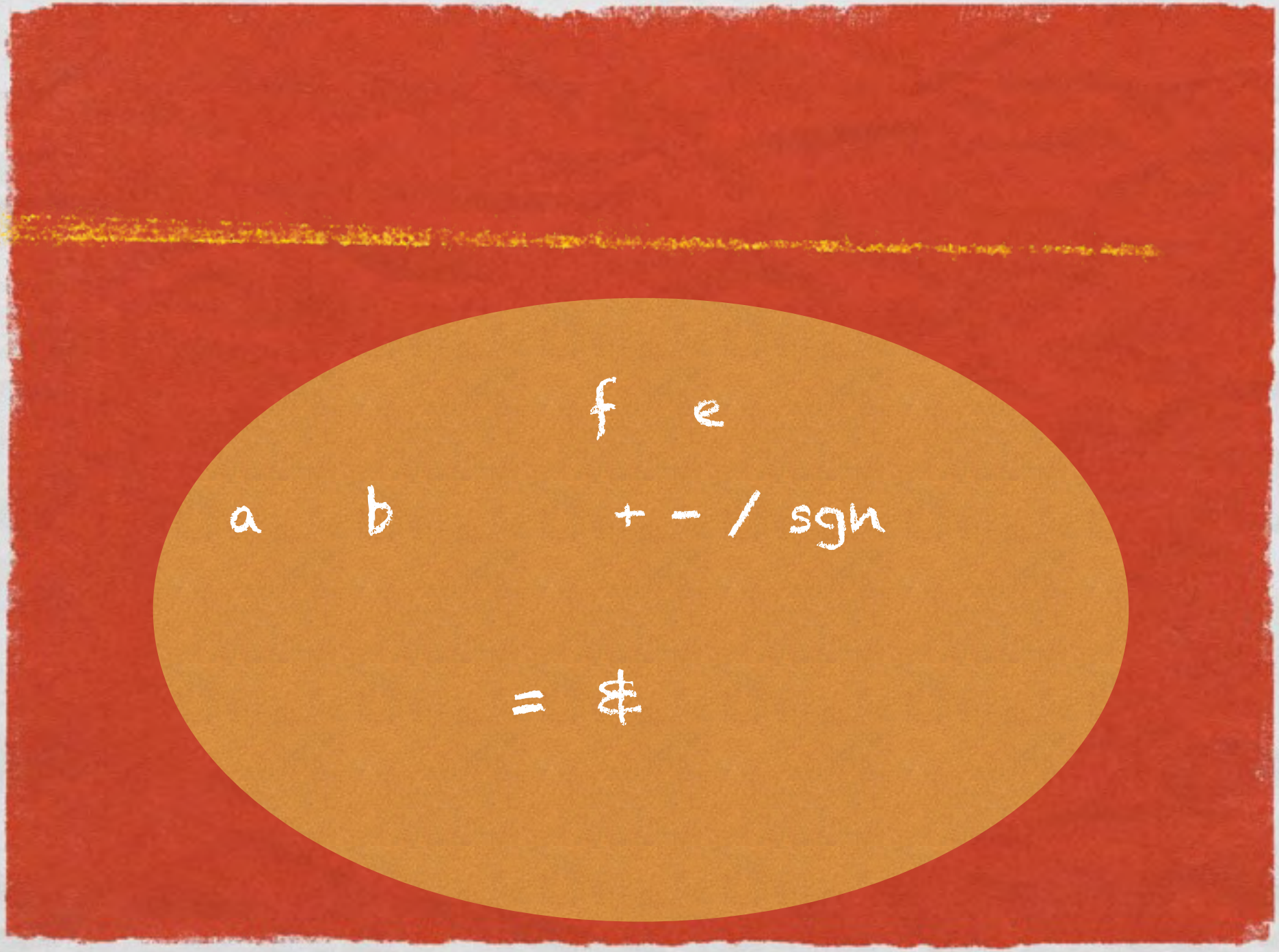
- Everything needed
- Structures capture all salient features of state: static operations & dynamic memory



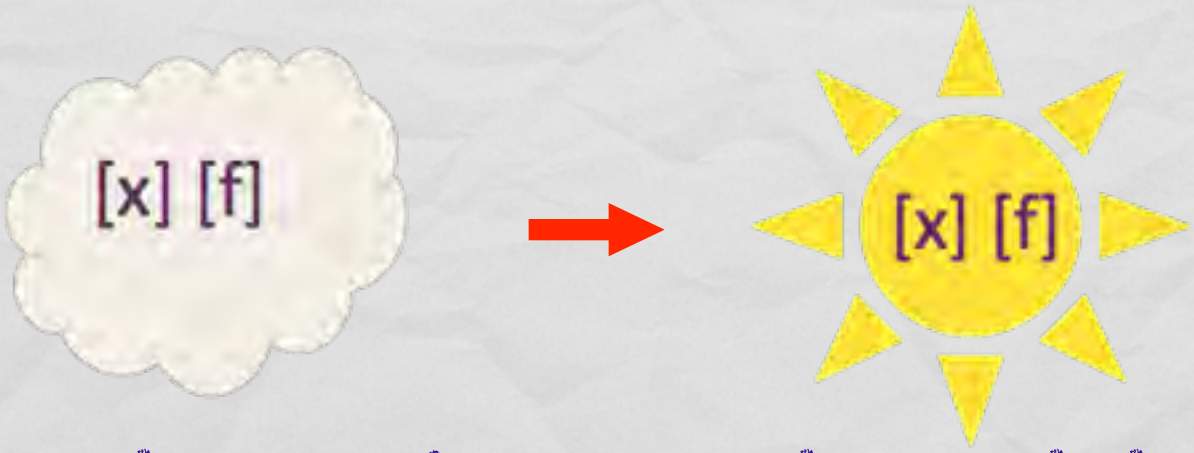
Geometry

- Domain (underlying set): points, lines, rays, circles, tuples and small bags
- Vocabulary & Operations: Compass; Ruler; =; ∩; Tuple & Bag

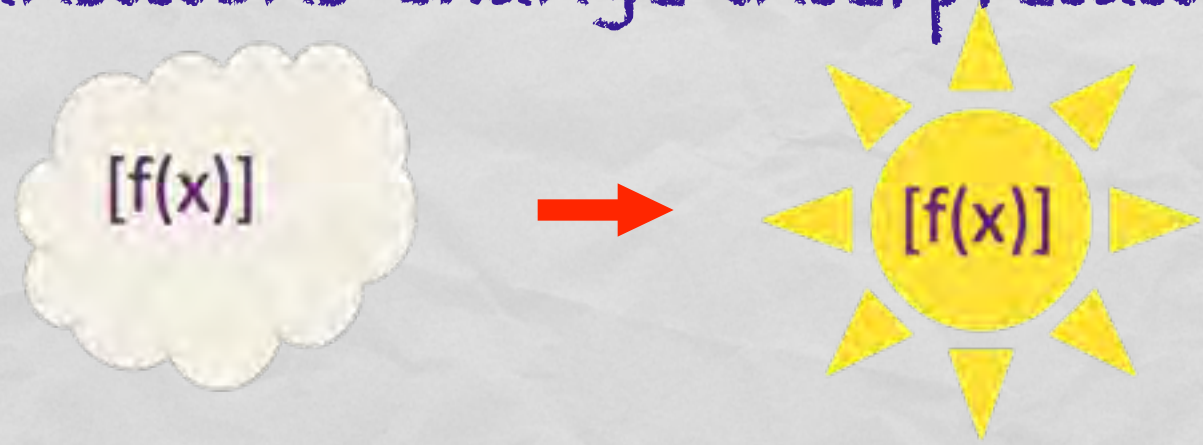



f e
a b
+ - / sgn
= ∉

Algebras



Transitions change interpretations

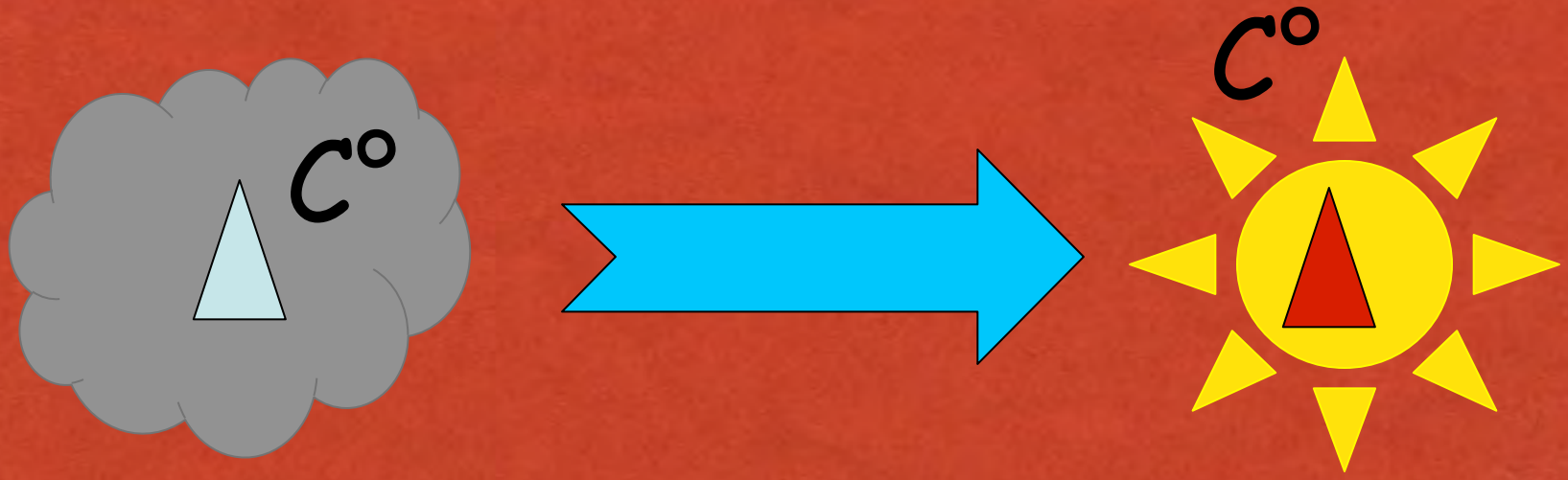






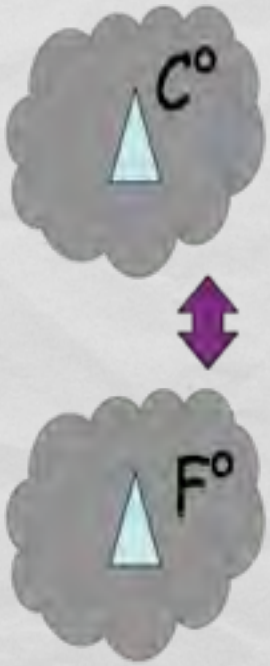
Abstract State

- States are (first-order) structures.
- All states share the same (finite) vocabulary.
- Transitions preserve the domain (base set) of states.

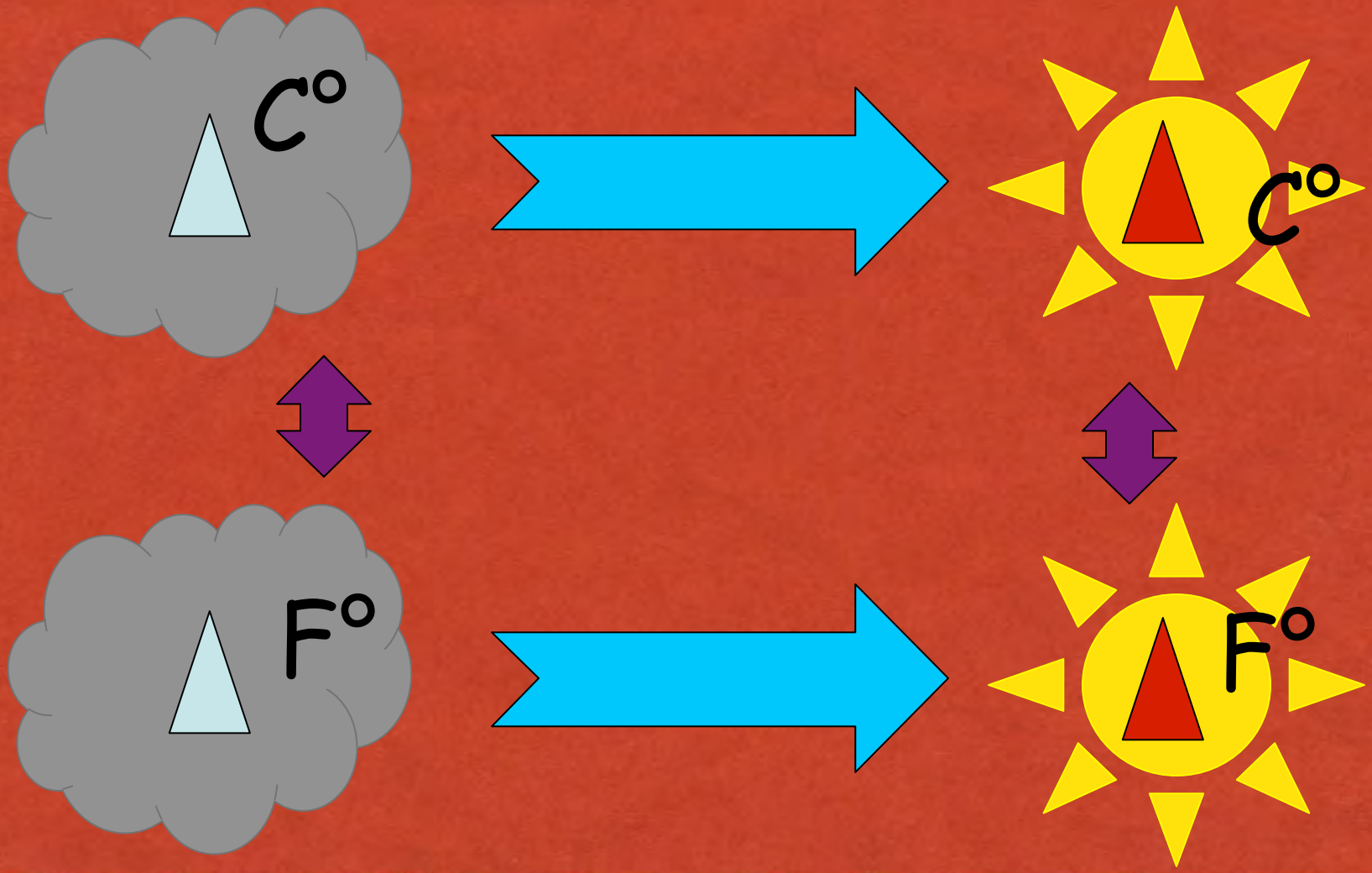

C°
C°

States & Transitions

- States are abstract (closed under isomorphism)
- Behavior does not depend on internal representation



Transitions respect isomorphisms



States are Abstract

- Data is arranged in a structure of a finite signature.



- An algorithm is abstract, thus applicable to all isomorphic structures.



Disallowed Operations

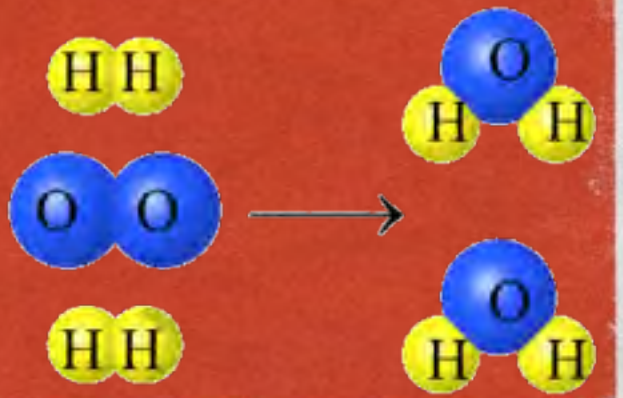
- ~~min(a1,a2,.....)~~
  - min{ai|...}
- 2^3^4^...^n
  - ↑n



III: Finite Description

What is a Transition?

- Transitions are algorithmic if they can all be described finitely (without presupposing any special knowledge).



Kleene

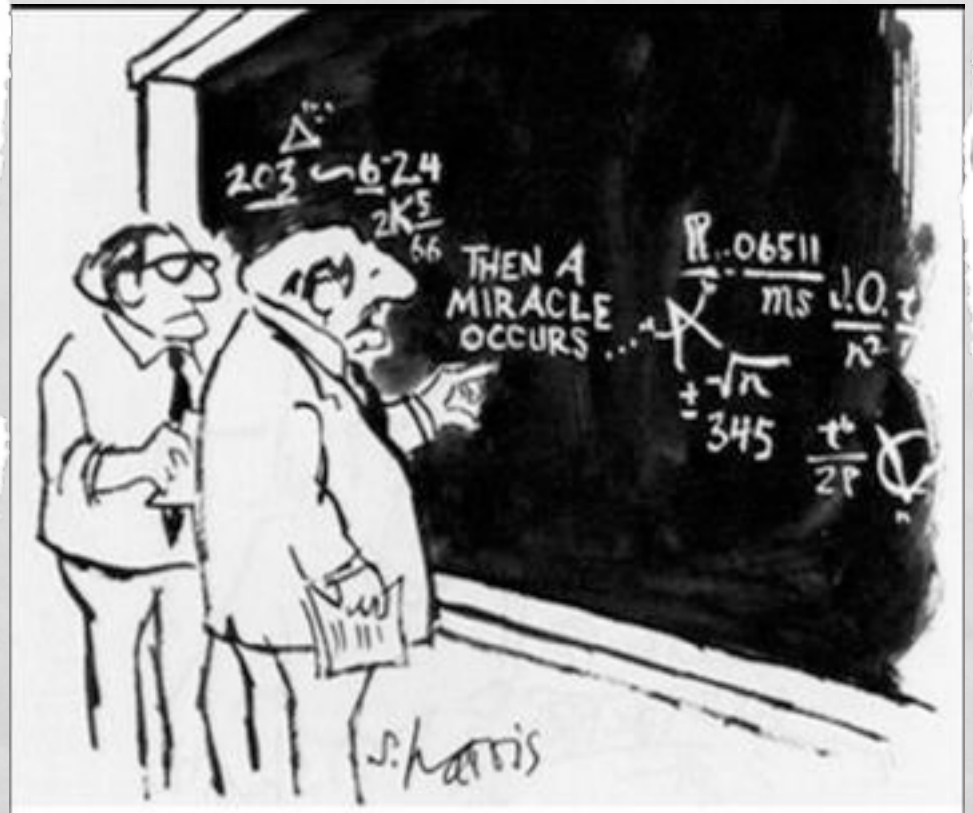
- An algorithm in our sense must be fully and finitely described before any particular question to which it is applied is selected....
- All steps must ... be predetermined and performable without any exercise of ingenuity or mathematical invention by the person doing the computing.

Joe Shoenfield

A method must be mechanical...

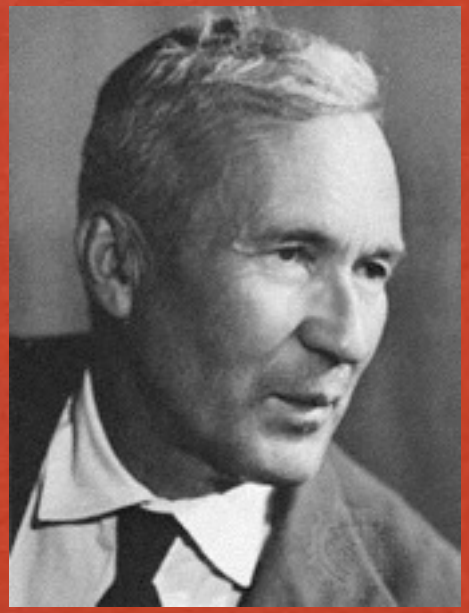
Methods that require insight are excluded.

Mechanical



"I think you should be more explicit here in step two."

Andrei Kolmogorov (1953)



The computational operations are carried out in discrete steps, where every step only uses a bounded part of the results of all preceding operations.

Algorithmic Transitions

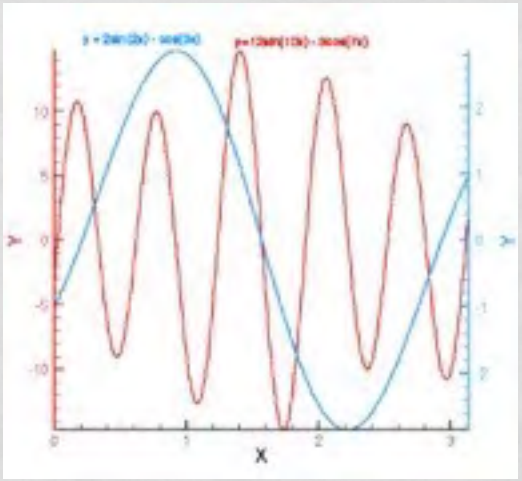


- Transitions are determined by a fixed finite set of terms, such that states that agree on the values of these terms, also agree on all state changes.

Yuri Gurevich

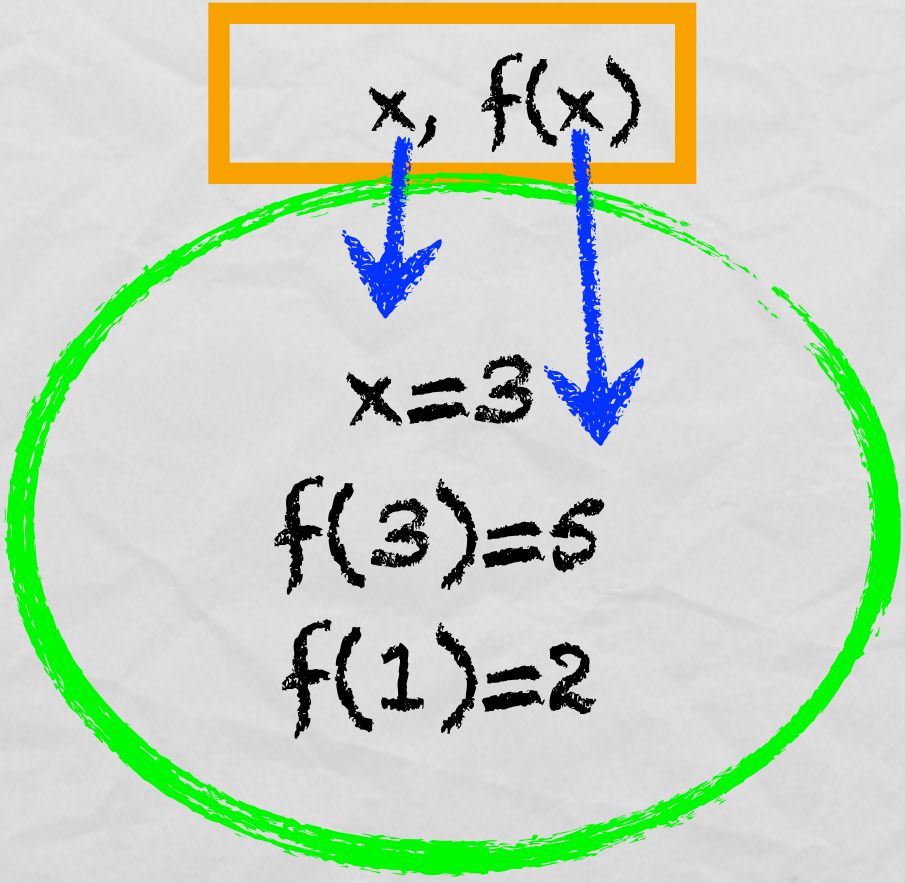
Transitions

- View state as location-value pairs
  - $f(a,b,c) \mapsto d$
- Changes to state X
  - $\Delta X = X' \setminus X$



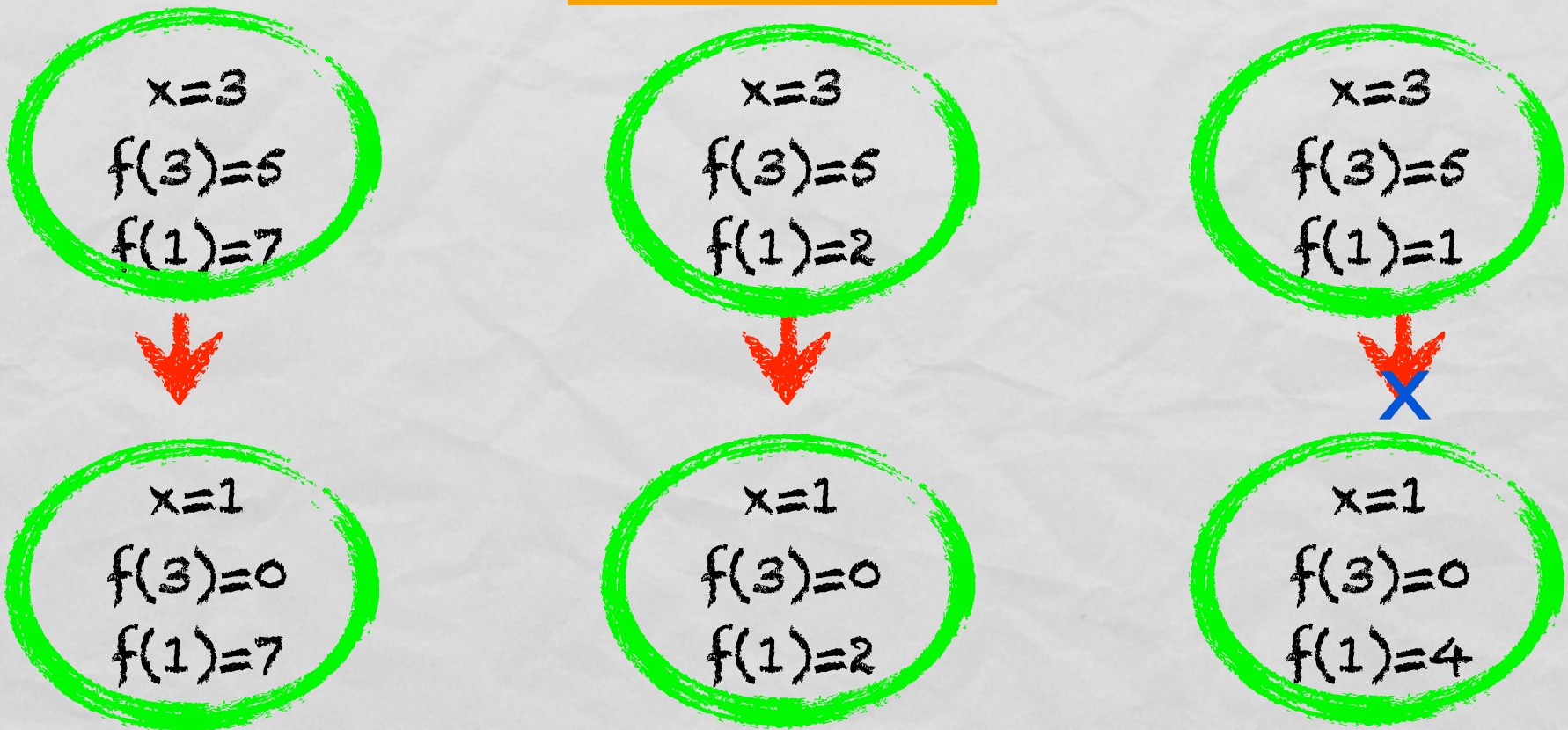


Terms & Locations

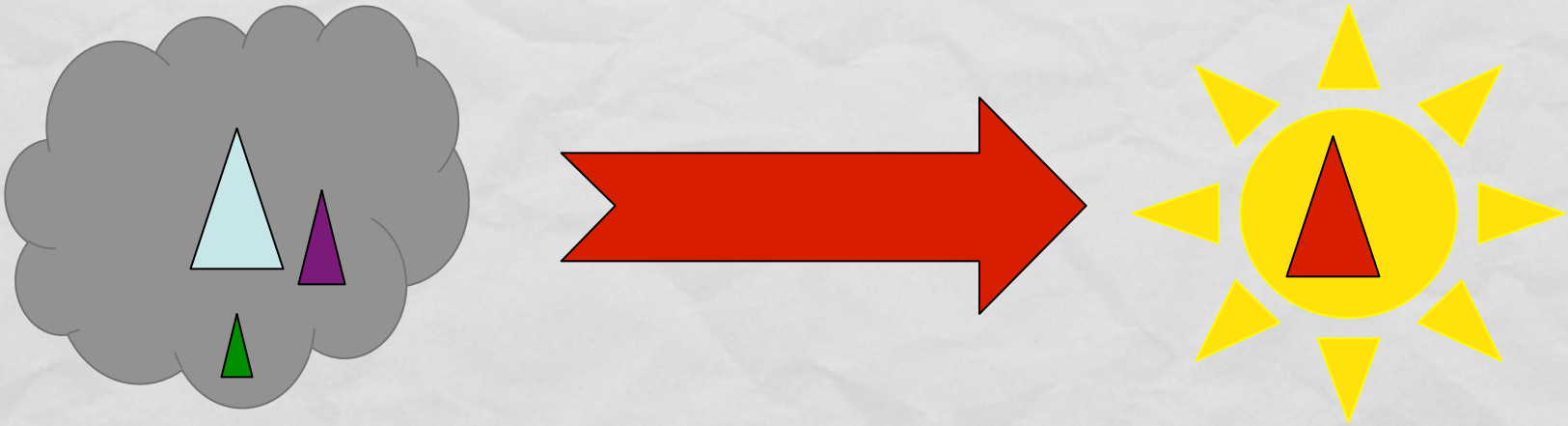


Critical Terms

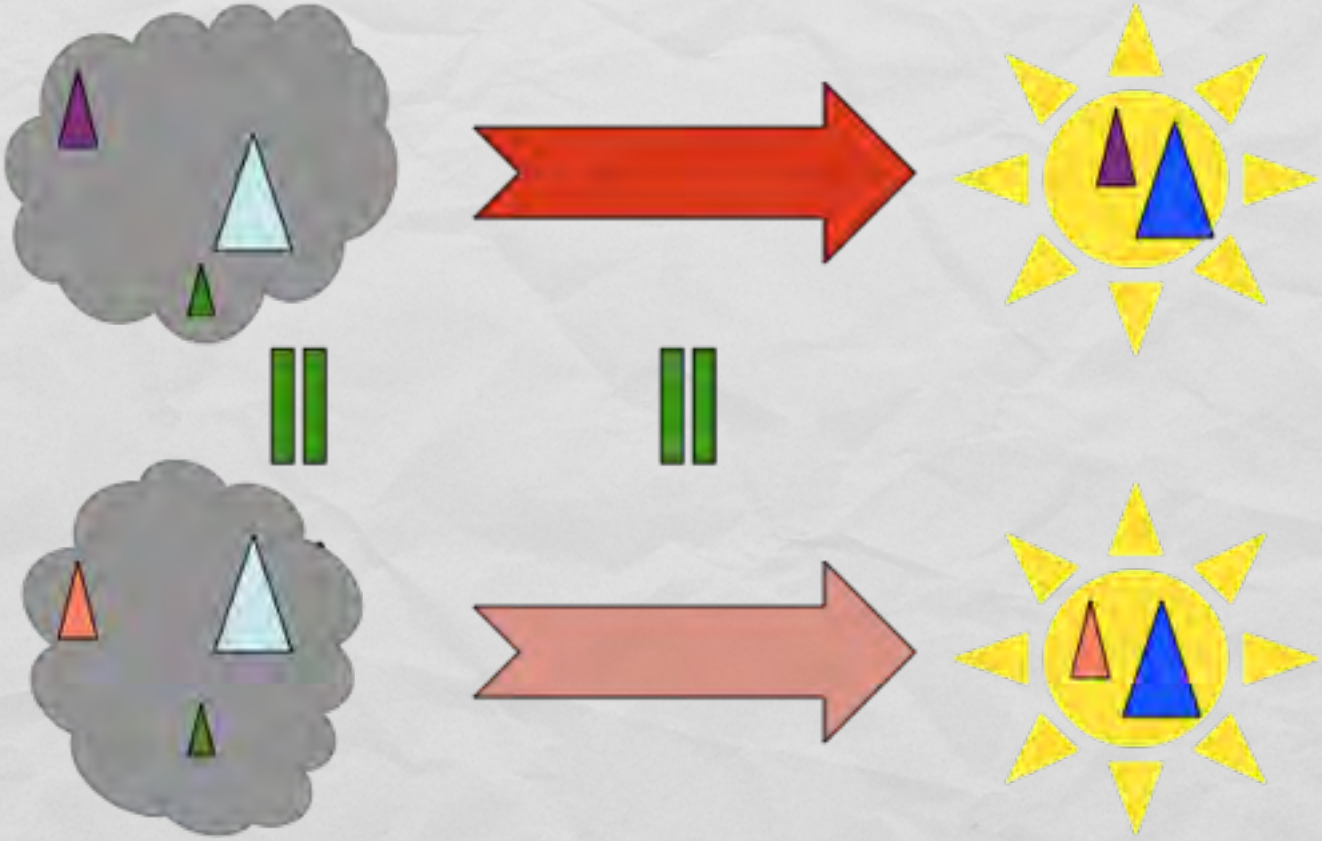
T: x, f(x)



Algorithmic Transitions



Algorithmic Transitions



ASM Theorem [Gurevich]

- Every classical algorithm can be emulated state-for-state, step-for-step by an abstract state machine (ASM).

Structured Programs

- :=
- ➡
- ;
- 

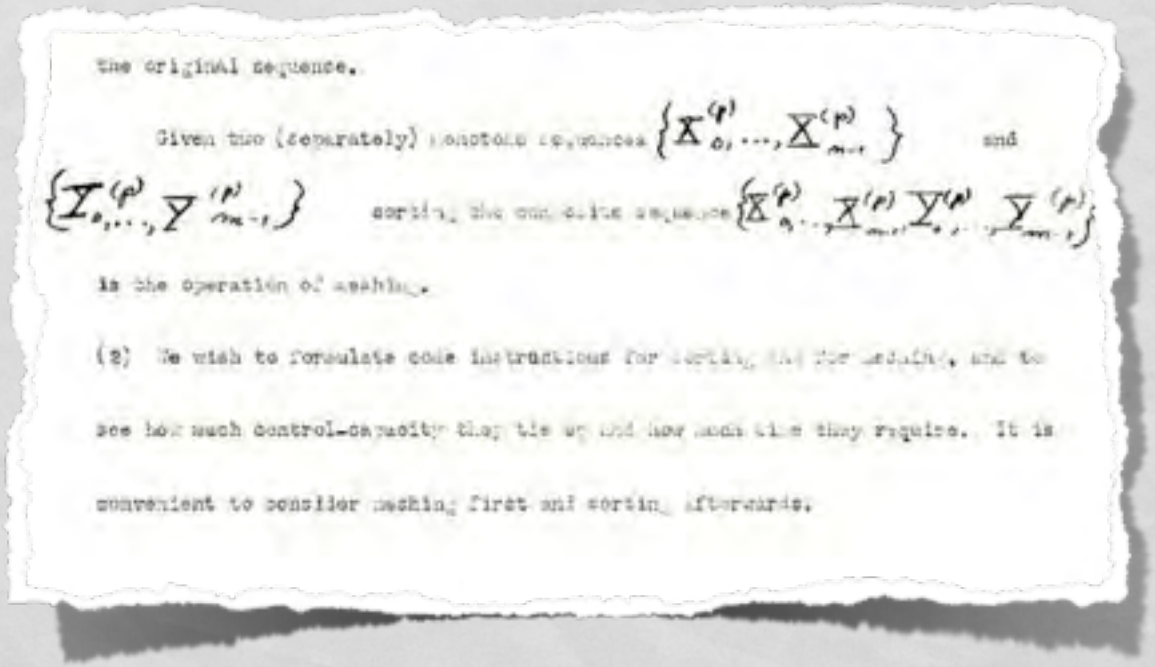
the original sequence.

Given two (separately) monotone sequences $\{X_0^{(p)}, \cdots, X_{m-1}^{(p)}\}$ and $\{Y_0^{(p)}, \cdots, Y_{n-1}^{(p)}\}$ sorting the combined sequence $\{X_0^{(p)}, \cdots, X_{m-1}^{(p)}, Y_0^{(p)}, \cdots, Y_{n-1}^{(p)}\}$ is the operation of meshing.

(2) We wish to formulate code instructions for sorting [illegible] for meshing, and to see how much control-capacity they tie up and how much time they require. It is convenient to consider meshing first and sorting afterwards.

Abstract State Machines

- :=
- ➡
- ||
- ∞



the original sequence.

Given two (separately) monotone sequences $\{X_0^{(p)}, \ldots, X_{m-1}^{(p)}\}$ and $\{Y_0^{(p)}, \ldots, Y_{n-1}^{(p)}\}$ sorting the combined sequence $\{X_0^{(p)}, \ldots, X_{m-1}^{(p)}, Y_0^{(p)}, \ldots, Y_{n-1}^{(p)}\}$ is the operation of meshing.

(2) We wish to formulate code instructions for sorting [illegible] for meshing, and to see how much control-capacity they tie up and how much time they require. It is convenient to consider meshing first and sorting afterwards.

Abstract State Machine

- $f(s_1,\ldots,s_n) := t$
- $c \rightarrow P$ if c then P [else Q]
- $\{P_1 \parallel \ldots \parallel P_k\}$

Proof

- Crux of proof:
  - Finite set of critical terms determining behavior & isomorphism condition imply that a term-based program describes transitions

Lemma

- States with the same equivalences between critical terms behave the same.

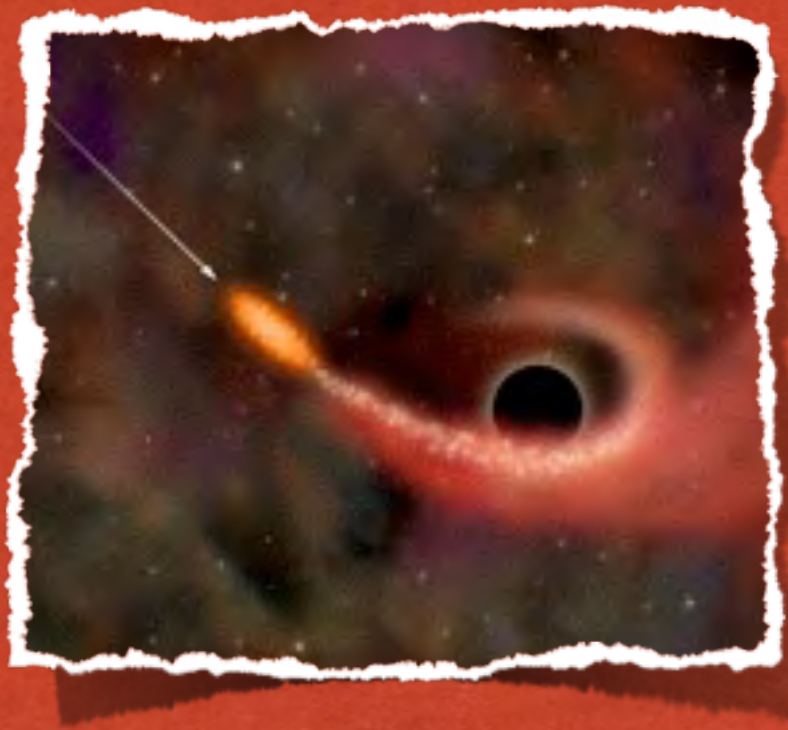
Lemma

- The location and new value of any update are given by critical terms.



Partial Operations

- Typically, a special "undefined" value captures partiality



Partial Operations

- Algorithm may hang if it looks for something that's not there



Matrix Inversion

- Computable reals
- Avoid unnecessary division by 0
- adj A / det A



Overkill



Conditions of ASM much more complicated than algorithm's.

Exactness

- Locations still to be accessed are determined by locations already accessed



Artificial Example

- if a then if b then if c then s := x
- if a then if ¬b then t := x
- if a then if ¬c then s := y

Artificial Example

- if a then if b then if c then s := x
- if a then if ¬b then t := x
- if a then if ¬c then s := y



Artificial Example

- if a then if b then if c then s := x
- if a then if ¬b then t := x
- if a then if ¬c then s := y



Artificial Example

- if a then if b then if c then s := x
- if a then if ¬b then t := x
- if a then if ¬c then s := y

Varied Changes

- $\Delta X = X' \setminus X$ updates
- $\Delta X = \bot$ terminal state
- $\Delta X =$  hang forever!

Refined Postulate

- Critical terms are localized
  - $X =_{\Gamma X} Y \rightarrow \Delta X = \Delta Y$
- Moreover, they determine exploration
  - $X =_{\Gamma X} Y \rightarrow \Gamma X = \Gamma Y$
- $\cup \Gamma X$ is finite

Actually...

- $Z_1 =_G \ldots =_G Z_n \rightarrow \Gamma Z_1 = \Gamma Z_n$
  where $G = \bigcap_i \Gamma Z_i$

Equivalently...

- Critical terms are partially ordered
- $t \in \Gamma X \setminus \Gamma Y \Rightarrow$ some s<t in ΓX has opposite truth values in X and Y

Refined Theorem

- Every "exacting" algorithm is emulated, step for step, state for state, by an ASM
- Furthermore, the exact same locations are accessed



Representation Theorem

- Every algorithm is emulated precisely by an ASM
- Furthermore, at each step, the exact same locations are accessed

Transitions

- Transitions preserve domain
- Commute with isomorphism
- Do not decrease domain of definition



Extensions

- Interaction
- Parallelism
- Distribution

What is an effective algorithm?

An algorithm is effective if its initial states have a finite description

If initial states can be described, then all states can be

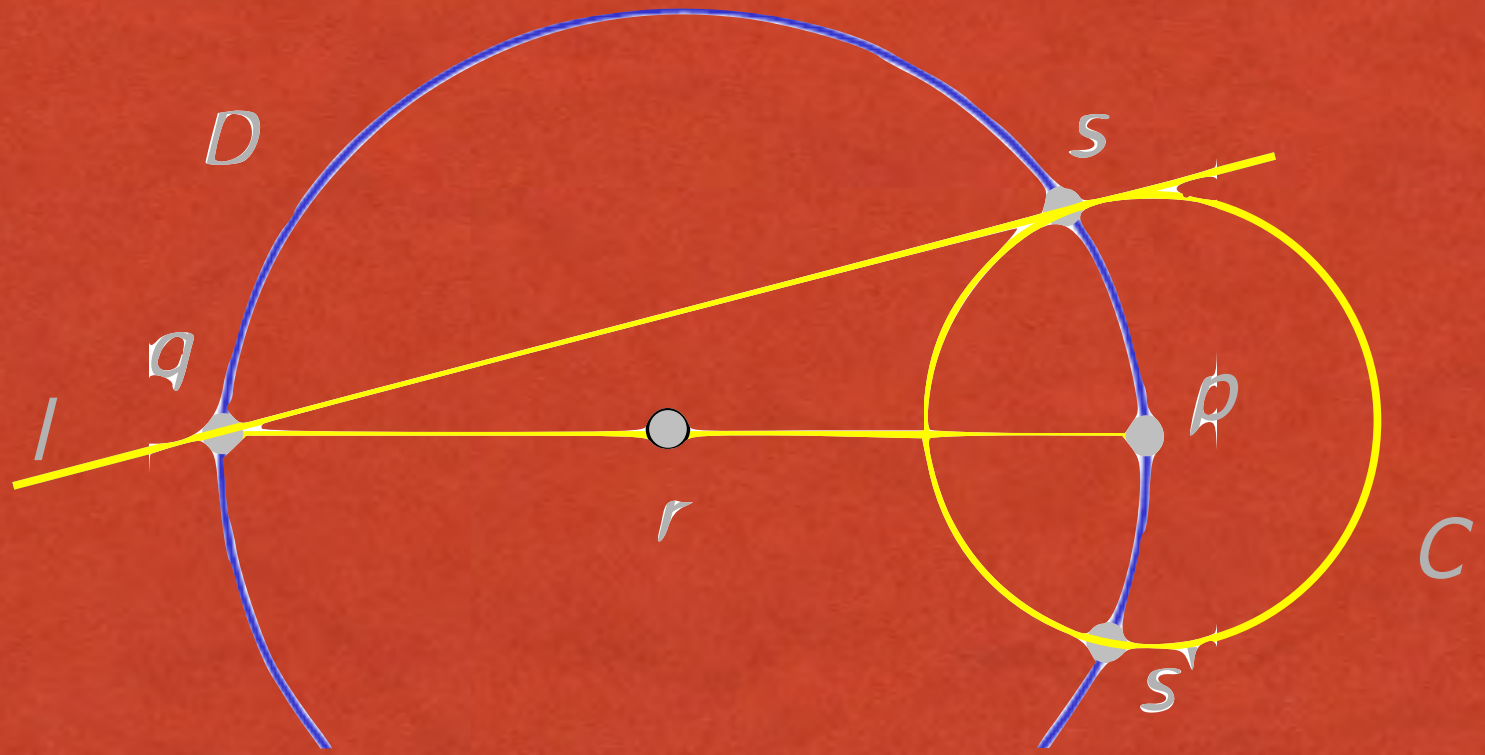
Why Study Effectiveness?

- What – in principle – can be done "mechanically"
- What – in principle – cannot be done "mechanically"

Algorithms: Effective?

- Construct tangent
- Bisection search
- Gaussian elimination
- Antenaresis

Geometry



Effective?

- Arbitrary points in the plane (not necessary finitely representable)
- Compass creates a circle with infinitely many points
- Geometric algorithms may test for intersection of lines

David Hilbert's Entscheidungsproblem



#2. Provide an effective method to determine whether a formula is valid.

Hilbert & Ackermann (1928)



The Entscheidungsproblem must be considered the main problem of mathematical logic.

The Entscheidungsproblem is solved when we know a procedure that allows for any given logical expression [in first-order predicate calculus] to decide by finitely many operations its validity or satisfiability.

Jules Henri Poincaré



We might imagine a machine where we should put in axioms at one end and take out theorems at the other, like that legendary machine in Chicago where pigs go in alive and come out transformed into hams and sausages.

The Question

- Is there an effective algorithm for doing
  - Logic?
  - Arithmetic?
  - Geometry?

Church's Thesis (1936)



Recursive functions "capture" effective computability

Alonzo Church

[We] propose a definition of effective calculability which is thought to correspond satisfactorily to [a] somewhat vague intuitive notion....

We now define ... the notion of effective calculable function of the positive integers by identifying it with the notion of a recursive function.

Kleene's Thesis I'

Every partial function which is effectively calculable (in the sense that there is an algorithm by which its value can be calculated for every n-tuple belonging to its range of definition) is potentially partial recursive.



Alan Turing (1936)

A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine

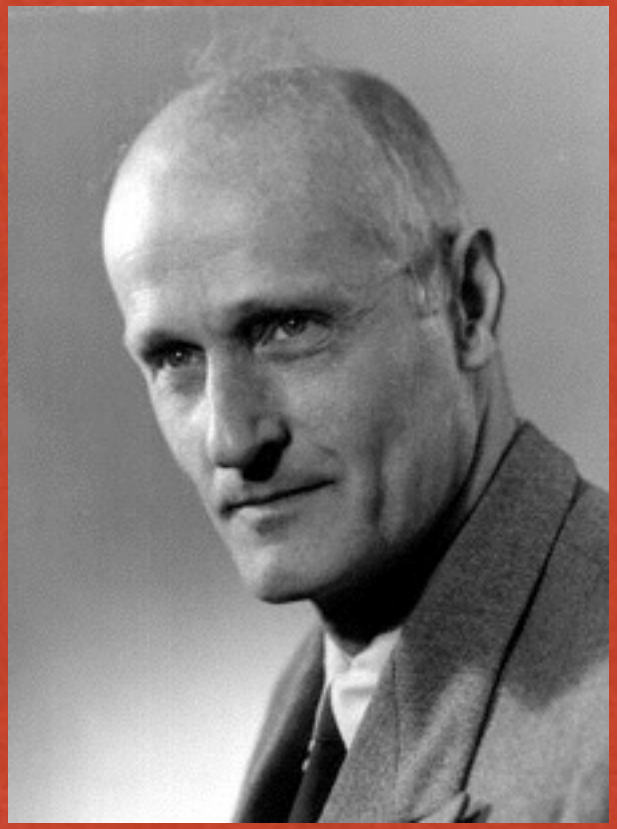


Turing's Thesis

Turing machines capture mechanical human computation



Kleene (1936)

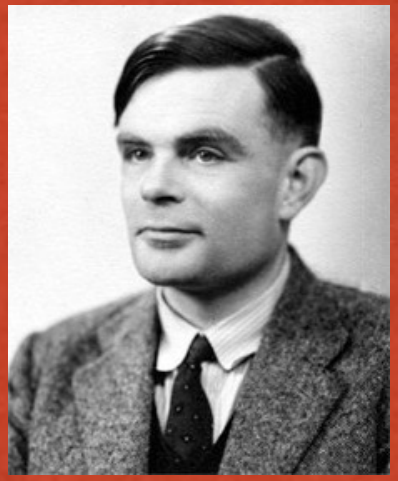


So Turing's and Church's theses are equivalent. We shall usually refer to them both as Church's thesis, or in connection with that one of its... versions which deals with "Turing machines" as the Church-Turing thesis.

Church-Turing Thesis



–



All effective computational models are equivalent to, or weaker than, Turing machines

Why Bother?

The thesis allowed Church and Turing to declare undecidability of the Entscheidungsproblem

Church

[Turing's machines have] the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately.

Kurt Gödel



- It was only by Turing's work that it became completely clear, that my proof is applicable to every formal system containing arithmetic.

Why believe?

Why Believe? (A)

- Model equivalence: Church, Turing, Post, Markov, Kolmogorov
  - What excludes a systematic error?
  - People thought primitive recursive was it

Why Believe? (B)

- Experience: programming languages compute only recursive functions
  - History is full of delayed discoveries

The Turing Tarpit

- Thue systems
- Post systems
- Lambda calculi
- Partial recursion
- Turing machines
- Markov normal algorithms
- Minsky counter machines
- Type 0 languages
- Kolmogorov-Uspenskii machines
- Neuring machines
- Wang machines
- Random access machines
- Quantum computers
- Billiard ball computers
- Least fixpoints
- Fortran, Algol, Lisp, C, Pascal, Logo, Ada, Java, ...

Why Believe? (C)

- Turing's analysis
  - This is by far the strongest argument

Turing's Premises

- Sequential symbol manipulation
- Deterministic
- Finite internal states
- Finite symbol space
- Finite observability and local action
- Linear external memory

Why Believe? (C)

- Turing's analysis
  - This is by far the strongest argument

  - Does not cover all algorithms
  - Only covers effective, classical algorithms
  - Presumes all data can be encoded as strings

How can we
prove the thesis?

Church to Kleene (1935)

[Gödel thought] that it might be possible ... to state a set of axioms which would embody the generally accepted properties of [effective calculability], and to do something on that basis.

Joe Shoenfield

It may seem that it is impossible to give a proof of Church's Thesis. However, this is not necessarily the case. We can write down some axioms about computable functions which most people would agree are evidently true.

It might be possible to prove Church's Thesis from such axioms.



Joe Shoenfield

However, despite strenuous efforts, no one has succeeded in doing this (although some interesting partial results have been obtained).



Sequential Algorithms

- Adopt the ASM postulates
- But the initial state may be non-computable

- Need a mechanism for input/output

Martin Davis



If non-computable inputs are permitted, then non-computable outputs are attainable.

What is Effective?

- Basic arithmetic certainly is
- The halting function might not be

No Magic





Implementation

- An implementation is a sequential algorithm, restricted to particular domain
  - So we may speak of the function computed by an implementation

Procedures

- A procedure is an implementation, in which the initial states of all runs are the same, except for inputs, which cover the whole domain
  - The answer is not given with the question

Postulates

I. An algorithm determines a computational sequence

II. Elements of the sequence can be arbitrary structures

III. Transitions are governed by some finite description

IV. Only 0, successor, and inputs are given initially

Church's Thesis

Any numeric function computed by a process satisfying the postulates is partial recursive.

Turing's Thesis

Any string function computed by a process satisfying the postulates can be computed by a Turing Machine.

Some Issues

- What if basic, given operations include more than successor?
- What if they include partial operations?
- What if they also manipulate non-numerical values?

Successor Plus

- No problem including any operations that can be programmed from 0 and successor.

There is more to algorithms than numbers!

Other Domains

- What should be considered effective for non-numeric or non-string domains?
- A numeric algorithm may make use of other data-structures.

Richard Montague (1960)

Turing's notion... applies directly only to functions on... natural numbers....
[For] another denumerable set... [it] depends on what correspondence... is chosen.

Natural... to restrict... to... correspondences which are in some sense 'effective'.... The sets of computable functions... correlated with two such correspondences will in general differ.



Ada Lovelace

Many persons ... imagine that because [Babbage's Analytical Engine] give[s] its results in numerical notation, the nature of its processes must ... be arithmetical and numerical rather than algebraical and analytical.

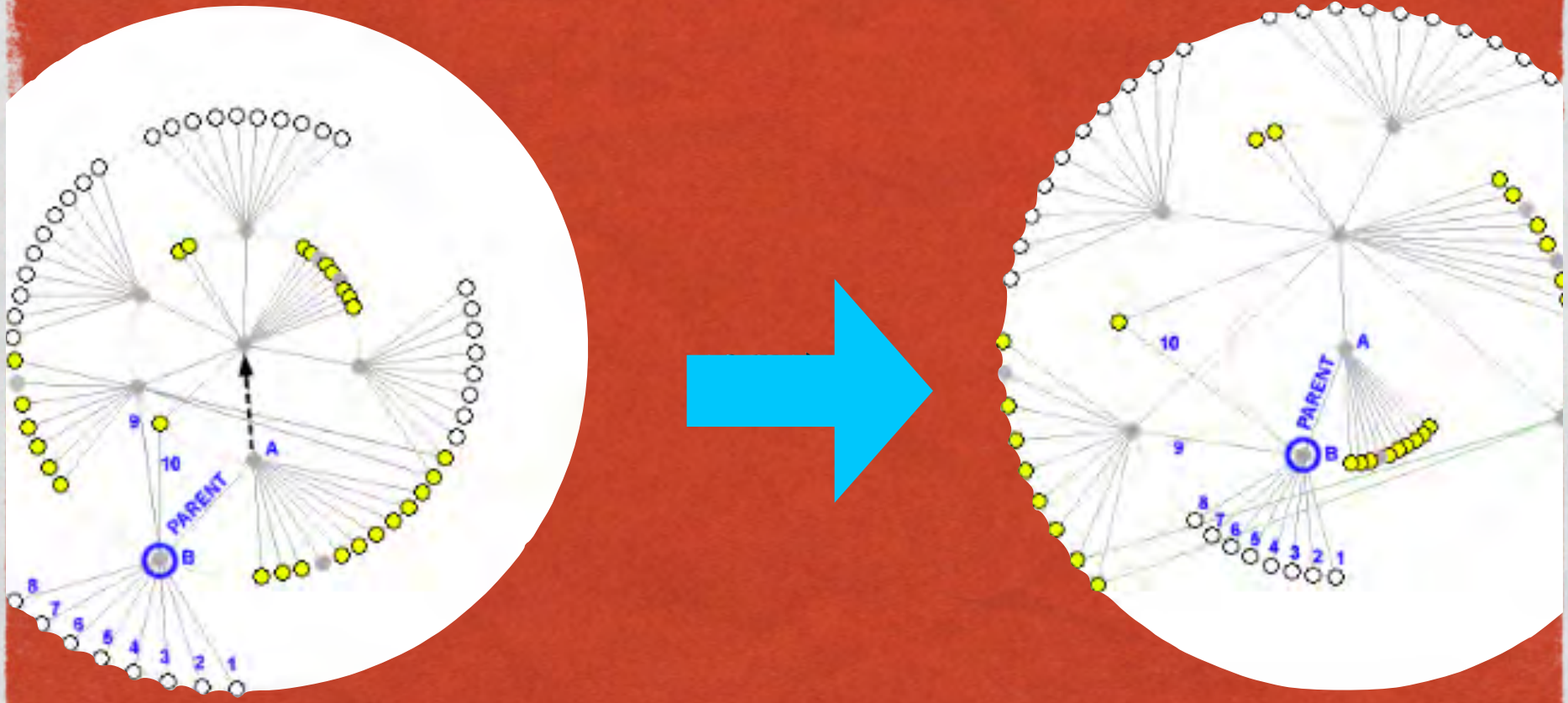
This is an error.

Ada

The engine can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols.



Graph Domain



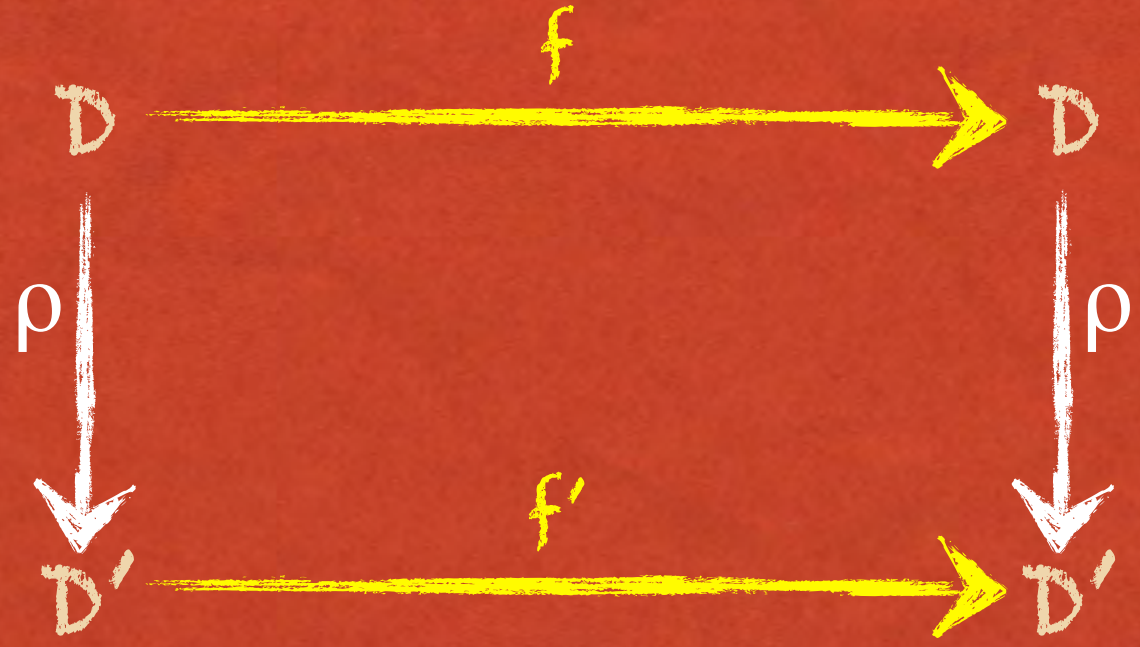
What Else?

- Consider

$$f(n) = \min i.\ p(g^{n+i}(c))$$

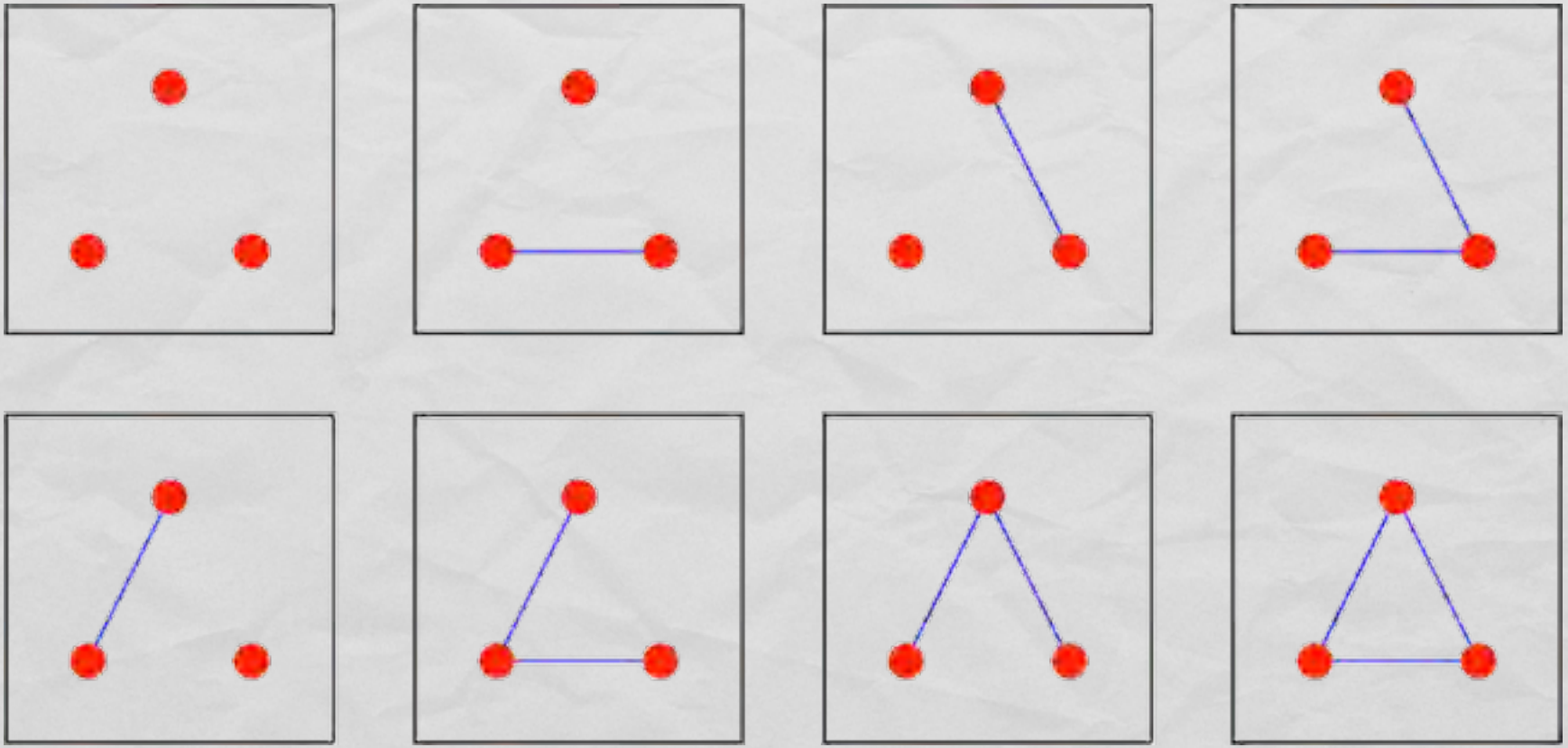
where p,g,c work over graphs, say.

Simulations

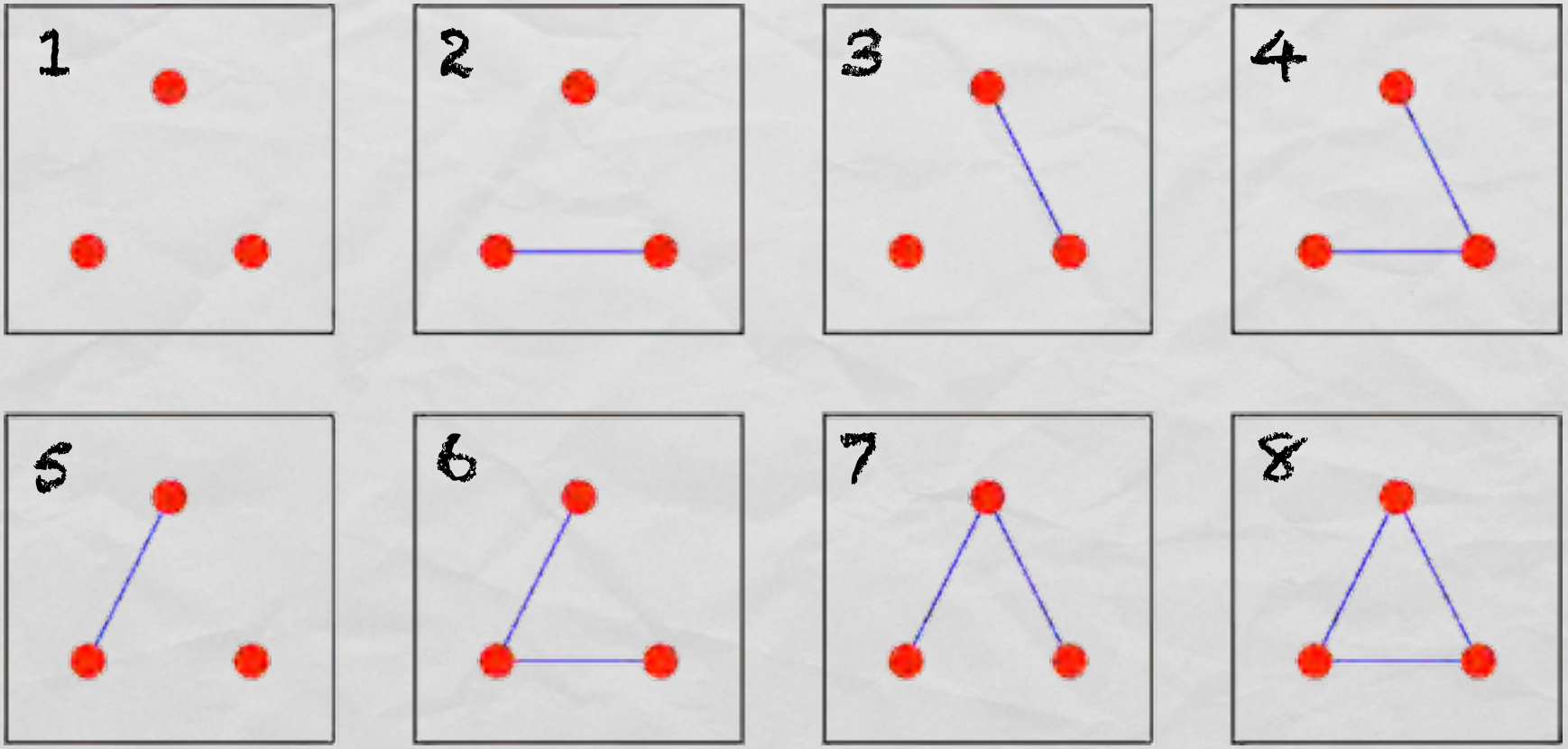


- ρ is an injection/bijection
- ρy = ⊥ iff y = ⊥

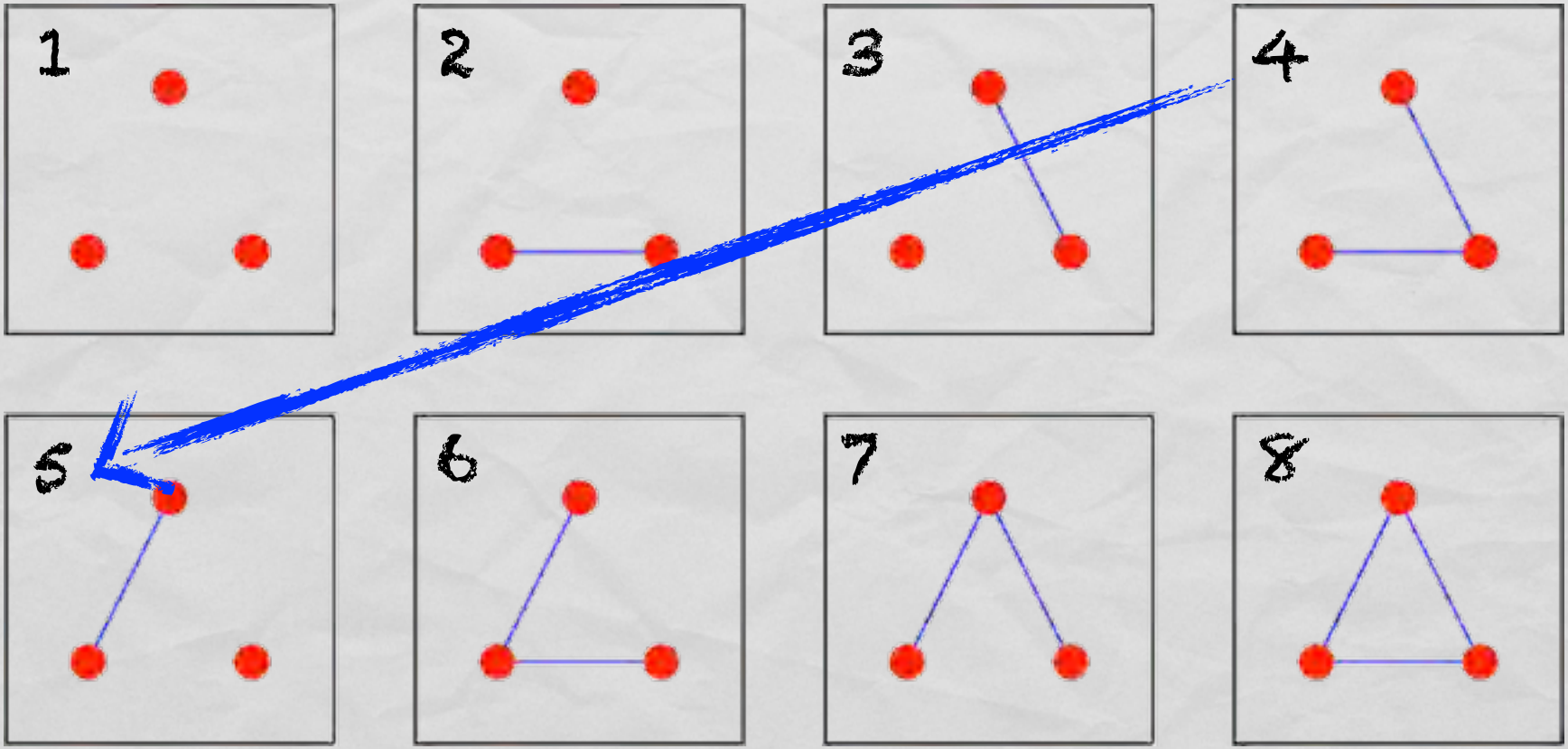
Tracking Functions



Tracking Functions



Tracking Functions



Computable Algebras

- An algebra is computable if it is simulated by (partial) recursive functions.

Constructor Domain

- Every element is the value of a unique term.
  - Isomorphic to the free term (Herbrand) algebra.
- Free constructors are effective.
  - So is anything that can be "bootstrapped" from them.

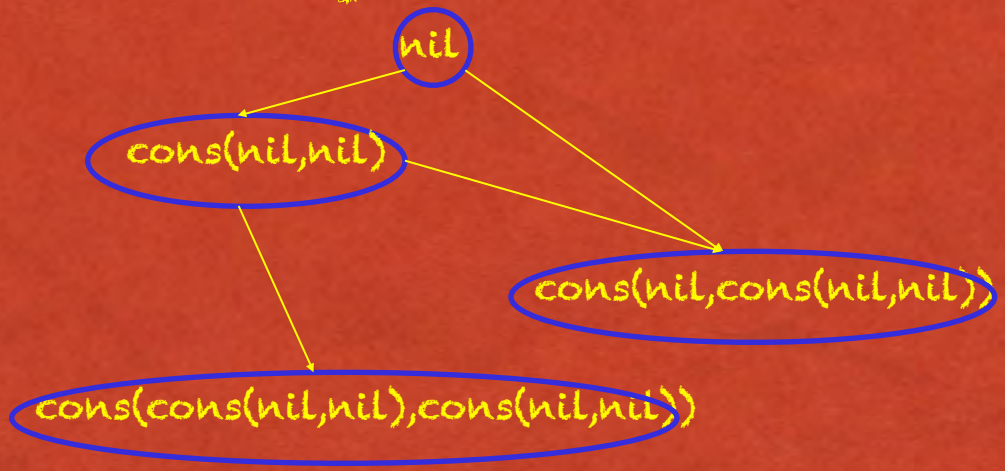
Examples



- A = { N; 0, S }



- B = { binary trees; nil, cons }



Constructive Algorithms

- A algorithm is constructive if its initial states consist of free constructors, plus operations (inductively) constructed in the same way.

Thesis

Turing Machines simulate (up to isomorphism) all effective models (under some representation).

And under no representation can an effective model do more.

Distinguishing States

- A state (X) is distinguishing if its induced equivalence relation (X |= s=t) is Turing-decidable.

[W. Reisig]

Effectiveness...

- The three notions of effectiveness are identical (up to isomorphism)
- There is only one maximal effective model (up to isomorphism)

Data Structures

- Church uses numbers
- Turing uses strings
- Kolomogorov uses graphs
- Gandy uses finite sets
- RAMs use arrays of numbers
- ASMs use any structure

Turing (1948)

Logical Computing Machines can do anything that could be described as "rule of thumb" or "purely mechanical". This is sufficiently well established that it is now agreed amongst logicians that "calculable by means of an LCM" is the correct rendering of such phrases.

Data Structures

The fact that all effective algorithms can be simulated by Turing machines justifies Turing's contention that effective computation is symbolic manipulation

Oracles

What if you dont believe the thesis?



Nagel &
Newman

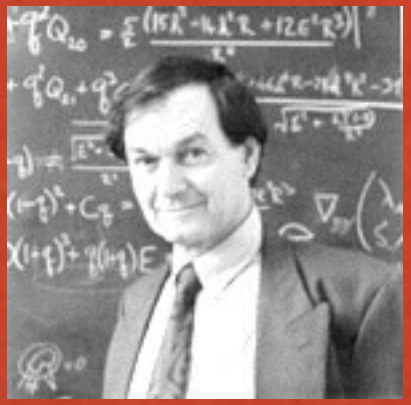




Gödel



Kalmár



Penrose





Lucas

Bringsjord

Allen Newell

- That there exists a most general formulation of machine and that it leads to a unique set of input-output functions has come to be called Church's thesis.





Robert Geroch & James Hartle



- It is difficult to see how any language that could actually be run on a physical computer could do more than Fortran can do. The idea that there is no such language is called Church's thesis.

Robin Gandy

- Thesis M: Whatever can be calculated by a machine (working on finite data in accordance with a finite program of instructions) is Turing-machine-computable.



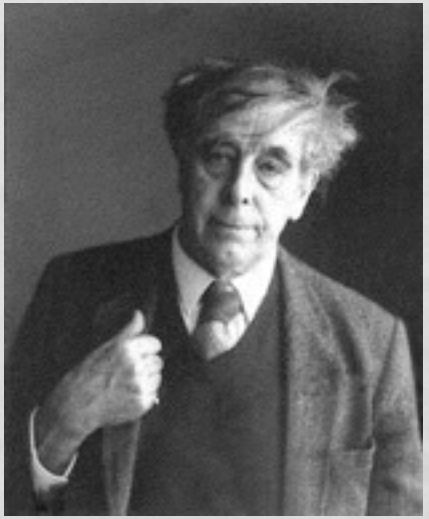
Laszlo Kalmár (1959)

There are pre-mathematical concepts which must remain [so].... Among these belong ... such concepts as that of effective calculability ... the extension of which cannot cease to change during the development of mathematics.



John Lucas (1961)

- Thanks to Gödel's theorem, the mind always has the last word.
- Mechanism is false.... Minds cannot be explained as machines.
- We cannot hope ever to produce a machine that will be able to do all that a mind can do: we can never, not even in principle, have a mechanical model of the mind.



Martin Davis



How can we ever exclude the possibility of our being presented, some day (perhaps by some extraterrestrial visitors), with a (perhaps extremely complex) device or "oracle" that "computes" an uncomputable function?"













Moral

Hypercomputation would require some of:

- Not a discrete sequence of transitions
  - transfinite or continuous computation
- Infinitary operations
  - $h(f,x,0) \vee h(f,x,1) \vee h(f,x,2) \vee \ldots$
- Infinite program
  - if x=0 then T || if x=1 then F || ...
- Uncomputable initial information
  - $h(f,x)$