Generic Parallel Algorithm

by Jenny Falkovich

Tel-Aviv University

December 2014

A concurrent system is an evolving collection of “cells”.

The evolution of the state of each depends on its “neighbourhood”.

An algorithm is a process whose possible evolutions have a finite description.

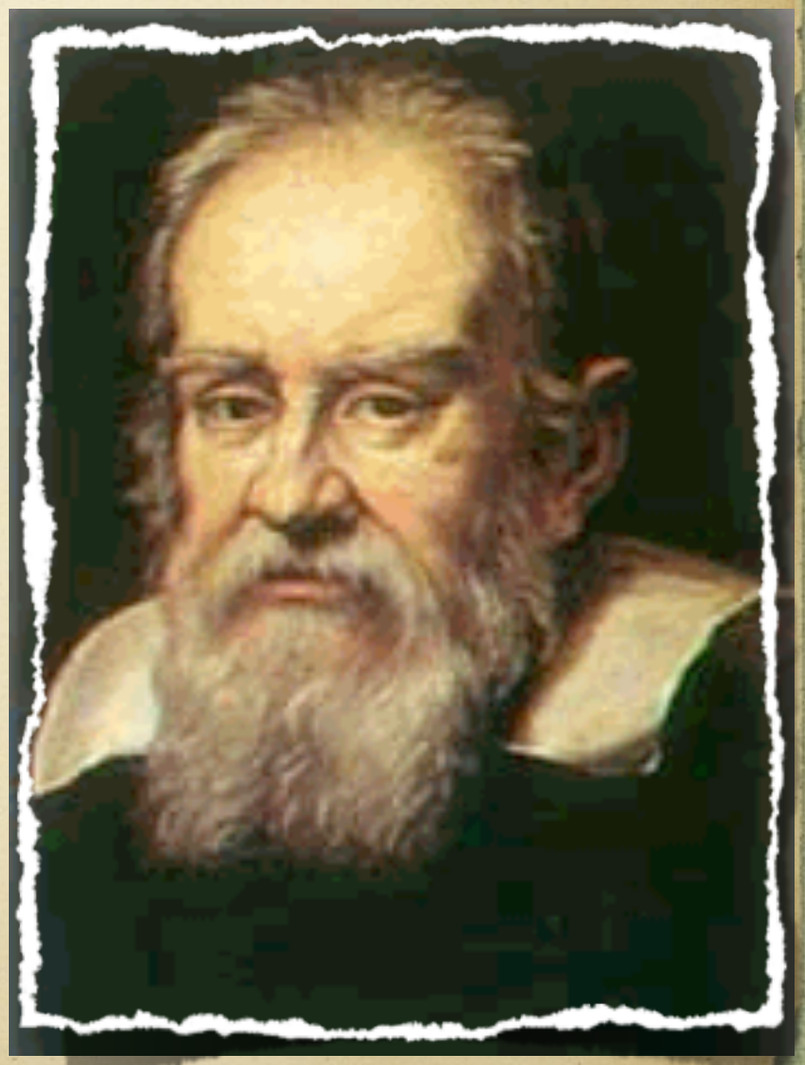
The notion of what it a computation (and algorithm) has developed rapidly

From sequential deterministic (Turing machines) to Interactive and Concurrent

Today we understand that merely any natural process may be considered as computation

Galileo Galilei(1564-1642)

La filosofia è scritta in questo grandissimo libro che continuamente ci sta aperto innanzi a gli occhi (io dico l'universo) ma non si può intender se prima non s'impara a intender la lingua e conoscere i caratteri né quali è scritto. Egli è scritto in lingua matematica e i caratteri sono triangoli, cerchi, ed altre figure geometriche senza i quali mezi è impossibile a intenderne umanamente parola; senza questi è un aggirarsi vanamente per un oscuro aberinto.

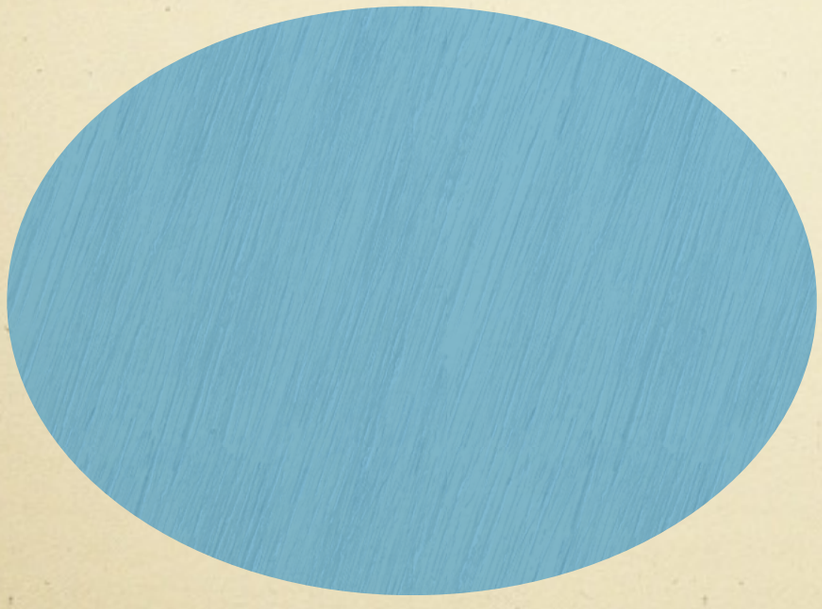


Levels of Abstraction

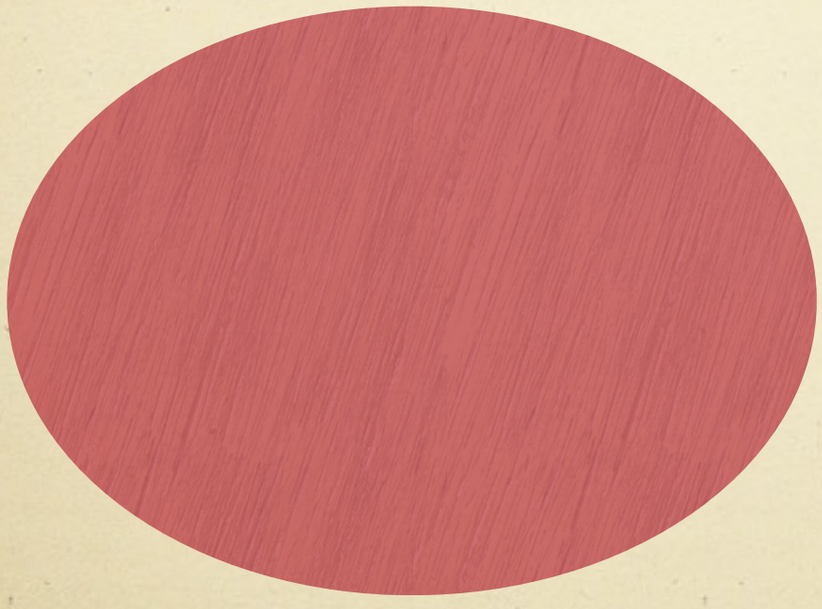
- Atoms -Physicist's point of view
- Molecules- Chemist's point of view
- Cells & Organisms - Biologist's point of view
- Populations - Sociologist's point of view
- Ecology - Ecologist's point of view

Evolving systems on any level of abstraction

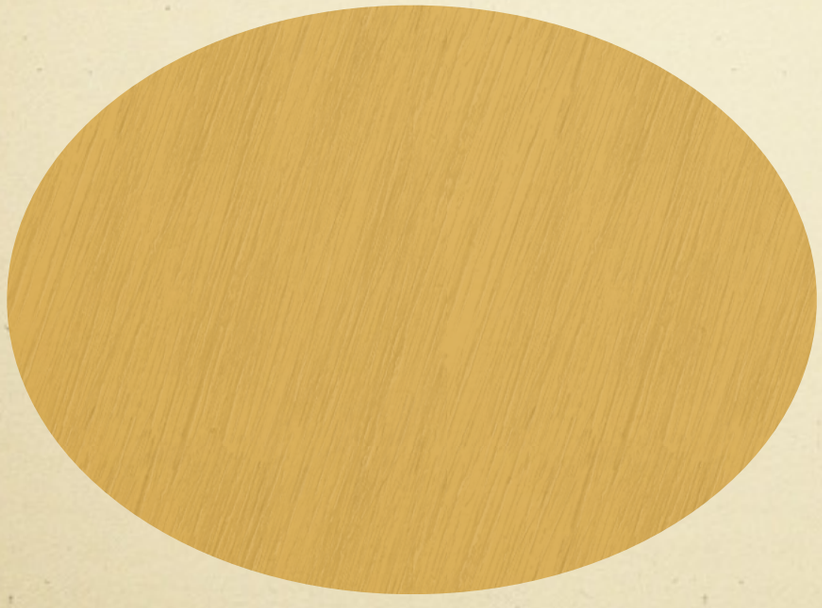
An Evolving State



An Evolving State

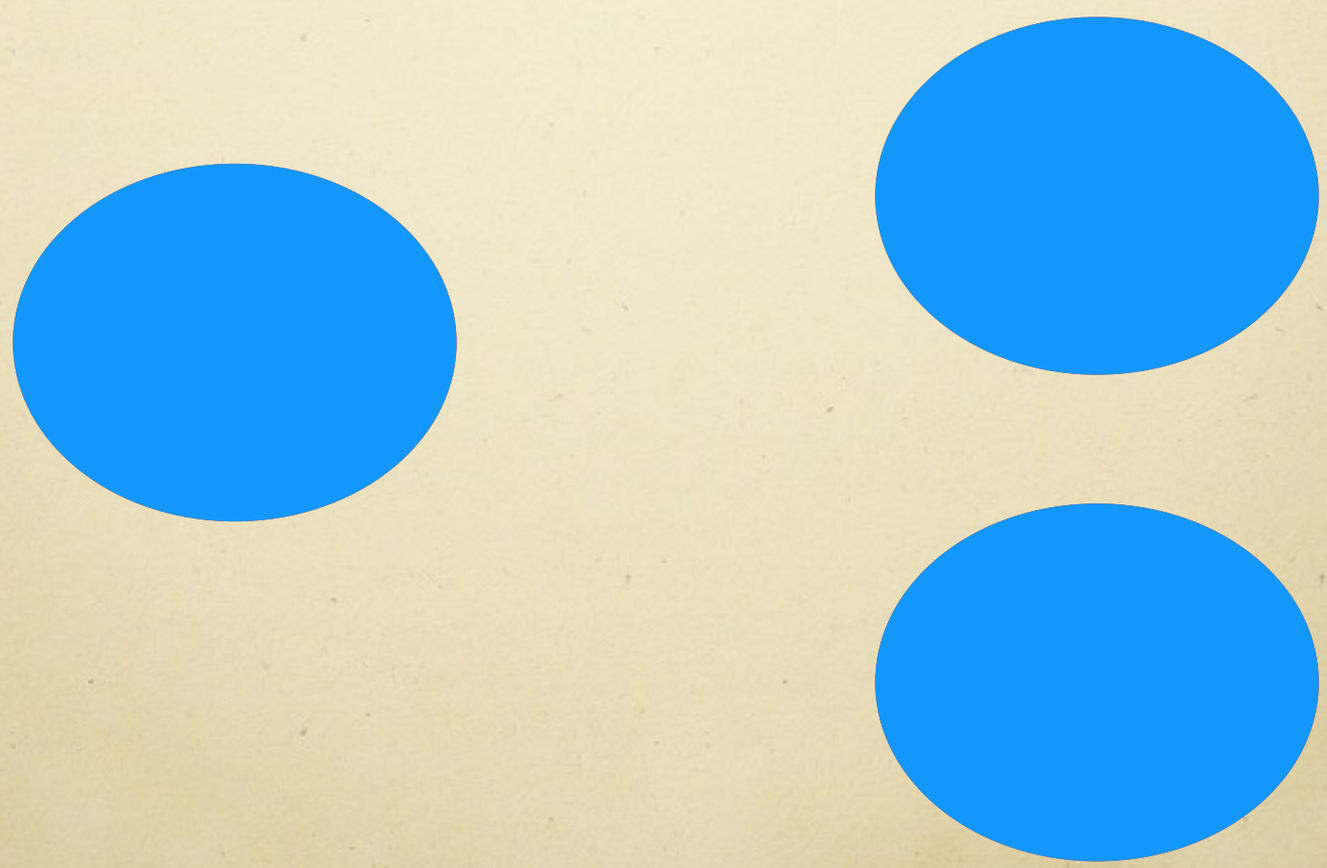


An Evolving State

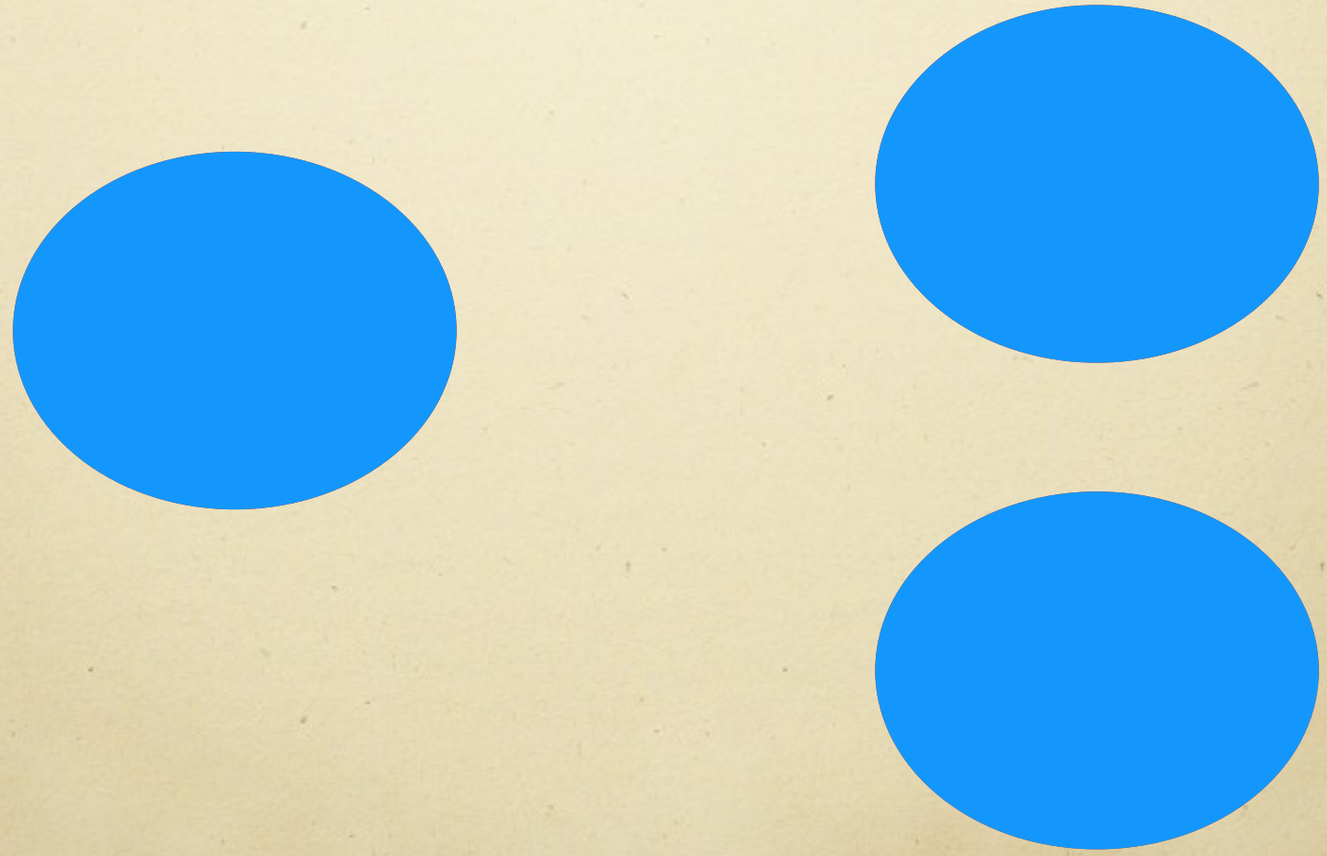


Discrete Evolution

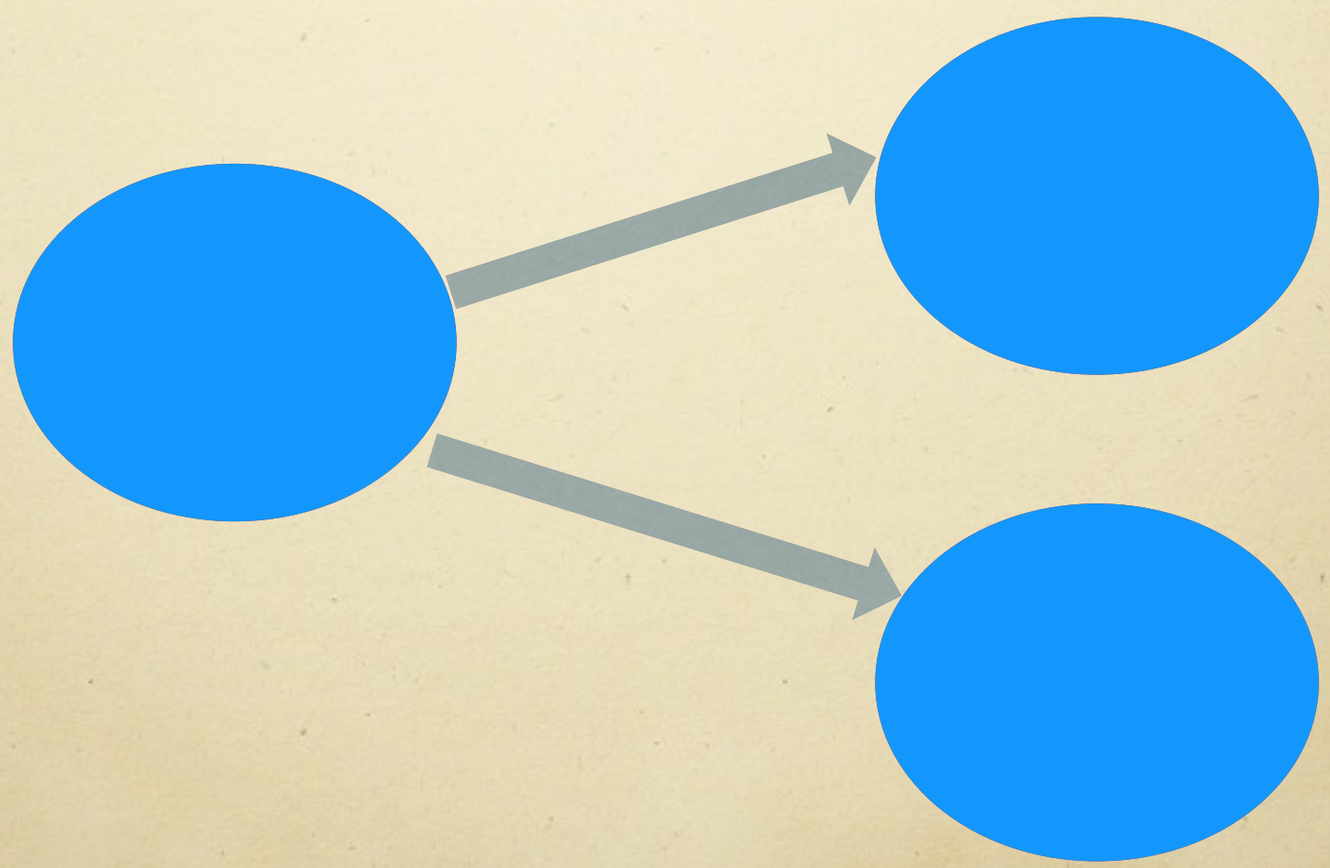
Parallel Evolution



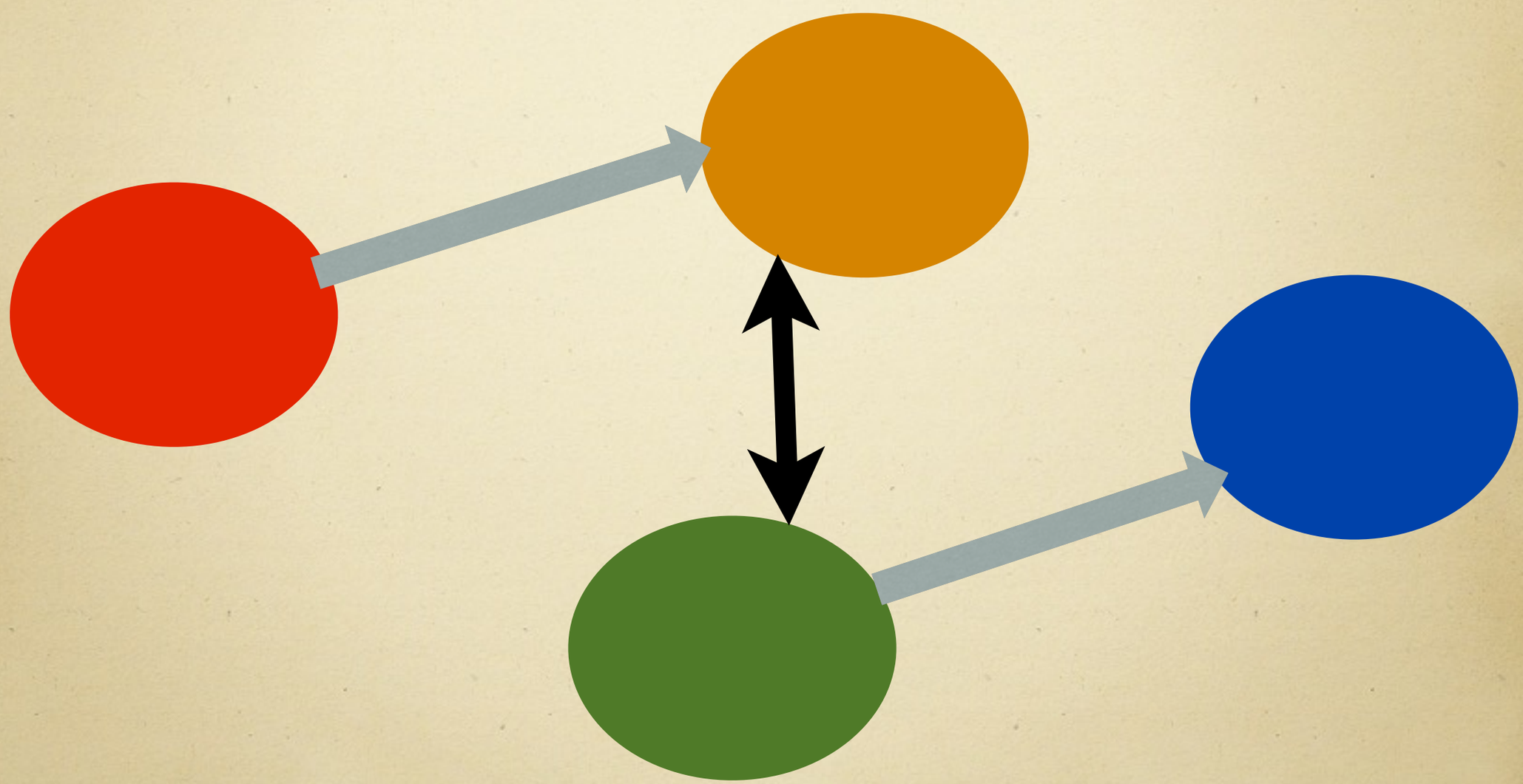
Concurrent Evolution



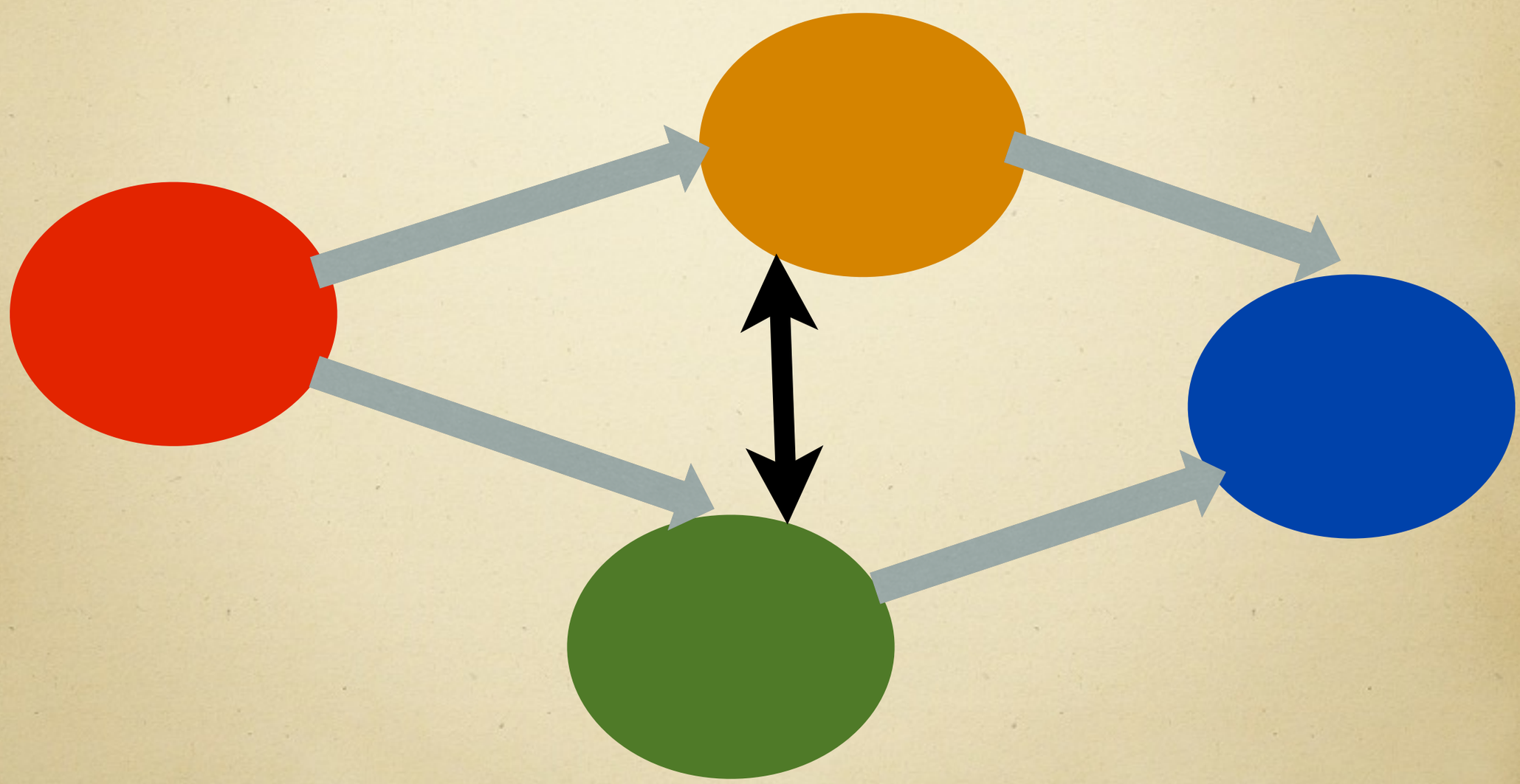
Interaction



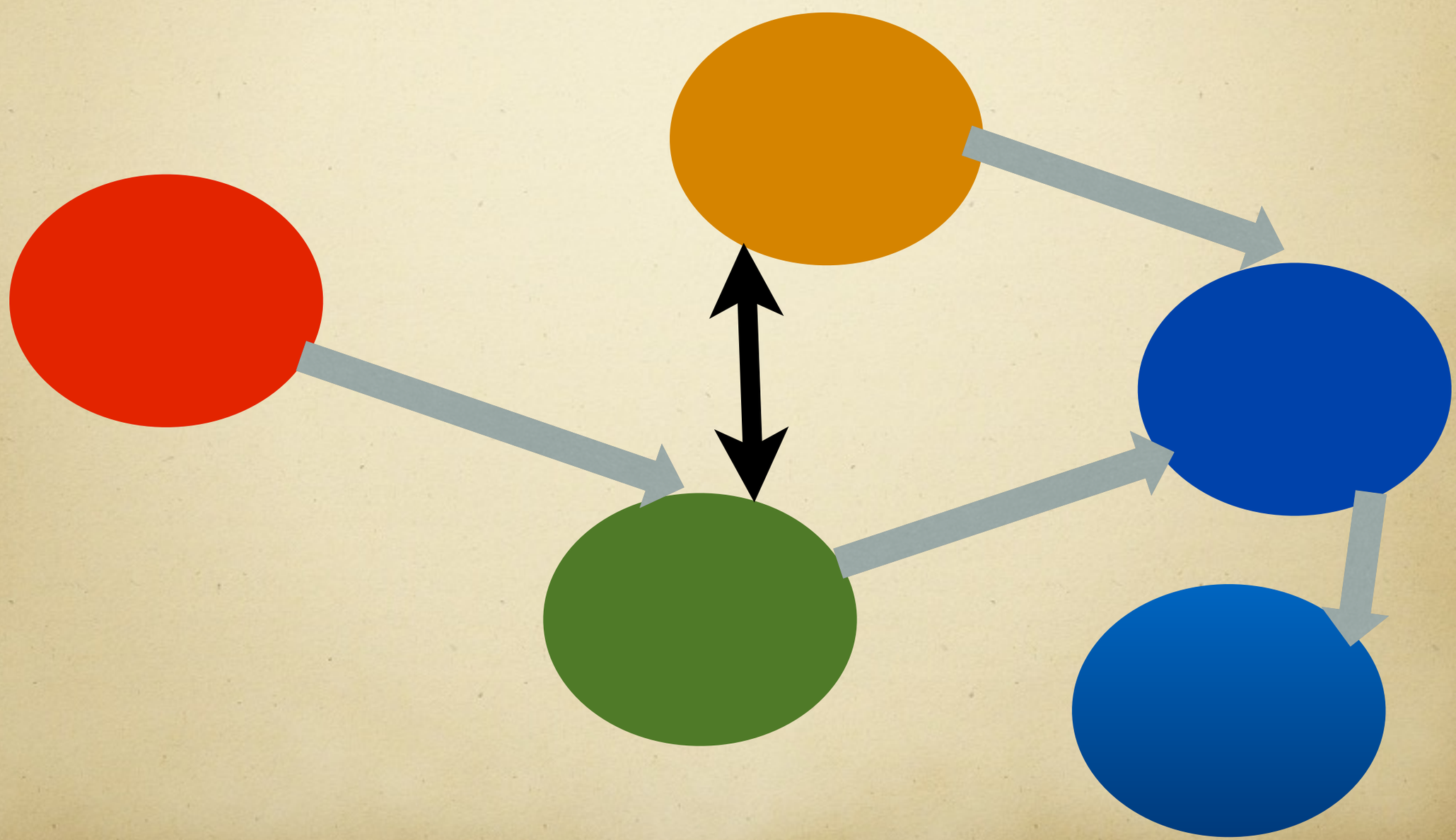
Evolving System



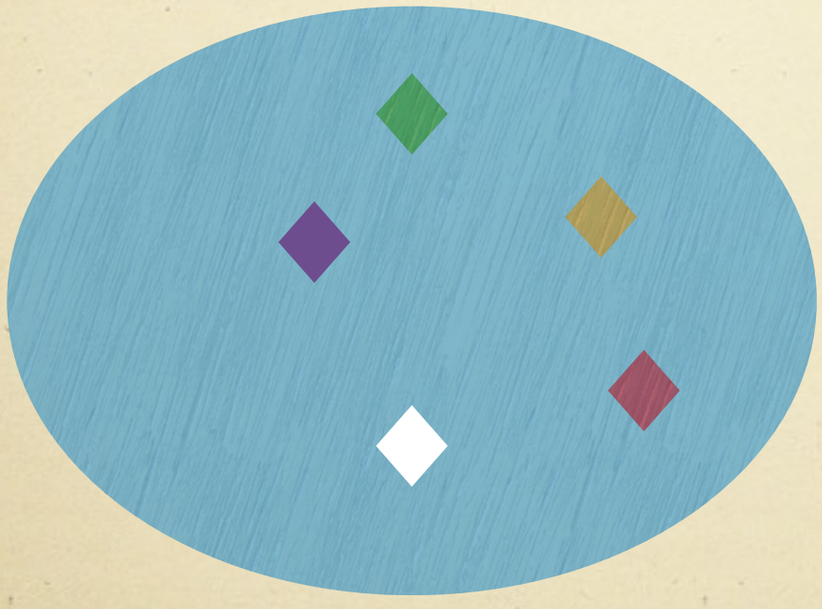
Evolving Topology



Childbirth



Inside a State



Evolving system is algorithmic if its evolution can be described finitely

Algorithms

Turing's Thesis

Turing machines capture mechanical human computation

Turing - Post

- Look — at any of a fixed number of locations
- Draw — at any of those locations
- Move — one of a fixed number of “heads”

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.

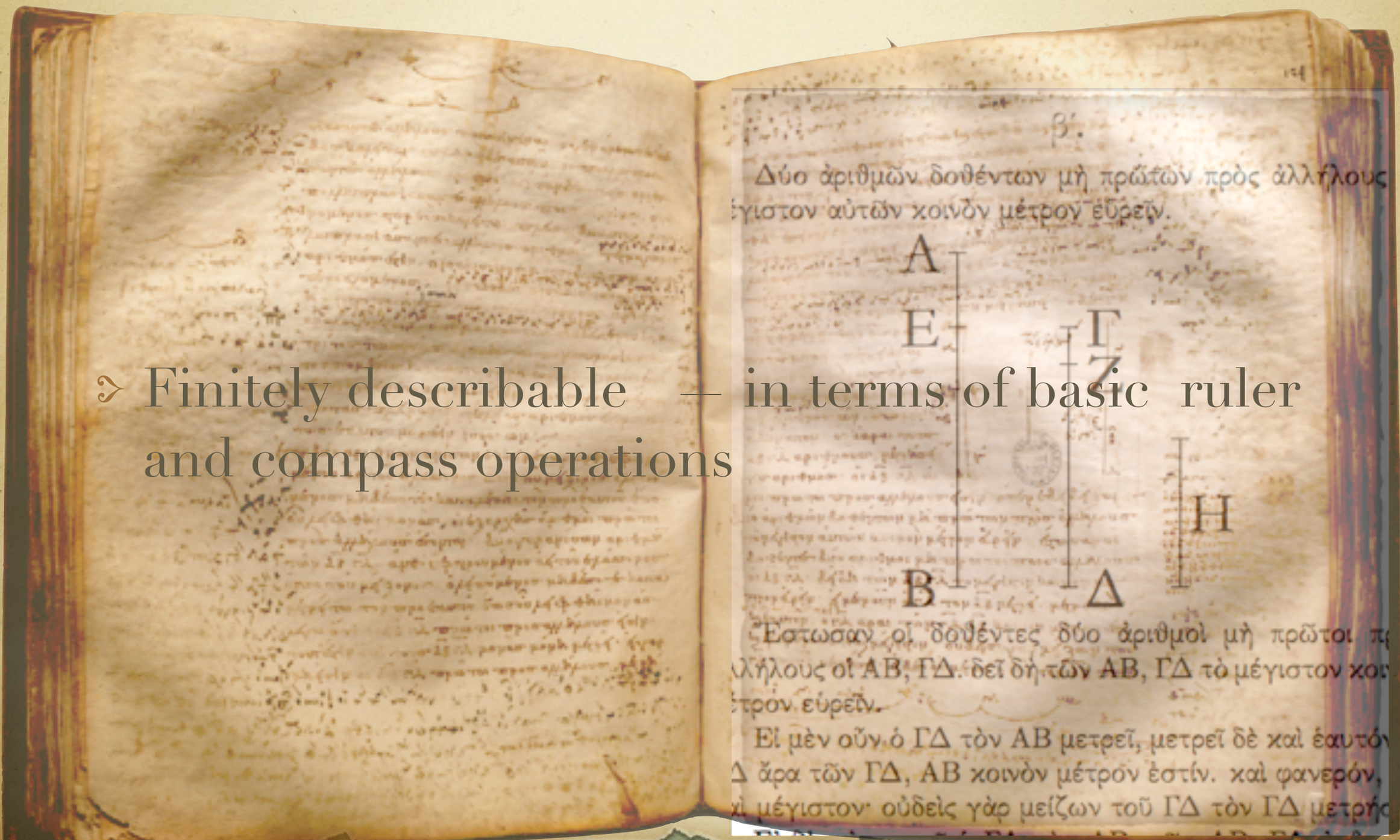
A Neolithic Algorithm

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

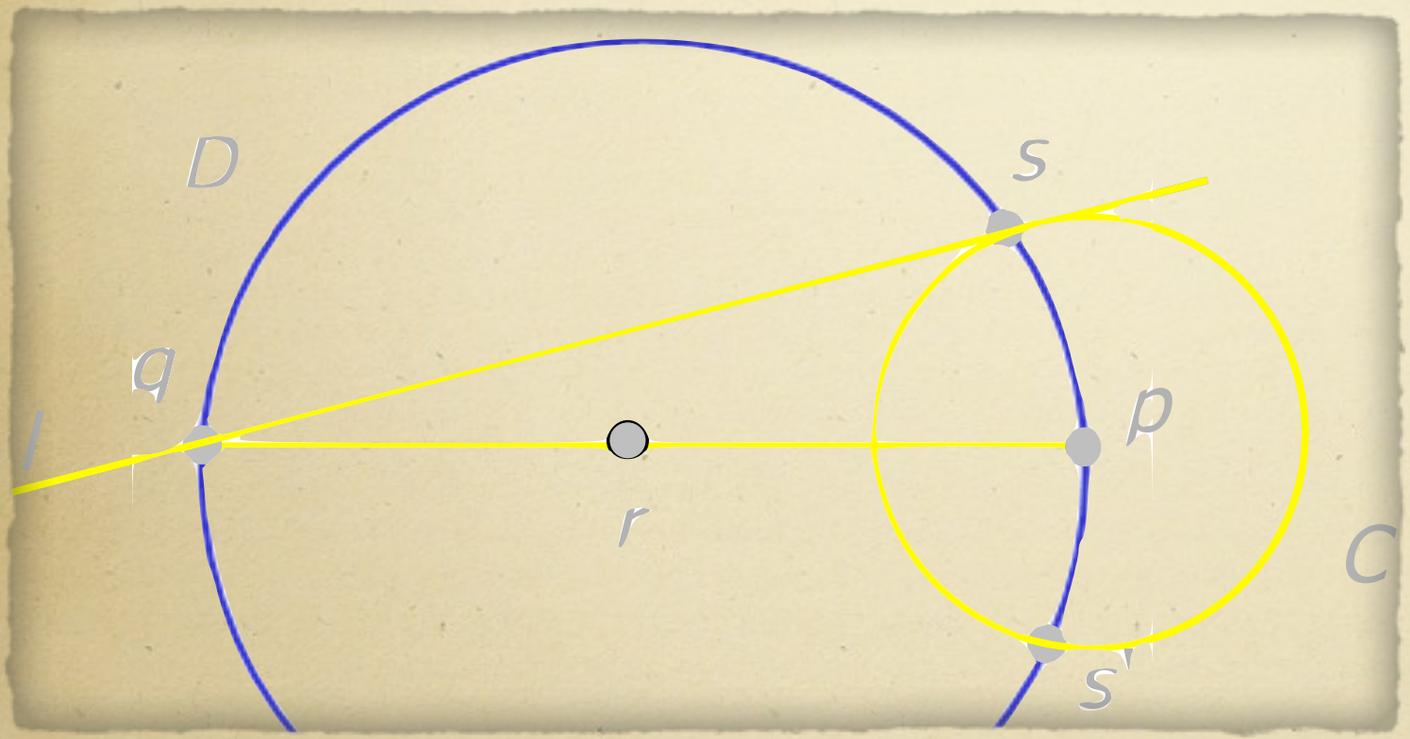
Euclid (c. -300)

Euclid's Antenaresis



- Finitely describable — in terms of basic ruler and compass operations

Euclid's Computer



If q ∉ C then

r := bisect(p,q)

D := Circle(r,q)

s :∈ C ∩ D

out := Line(q,s)

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.

Generic Algorithms

Generic

- Language independent
- Data independent

Genericity

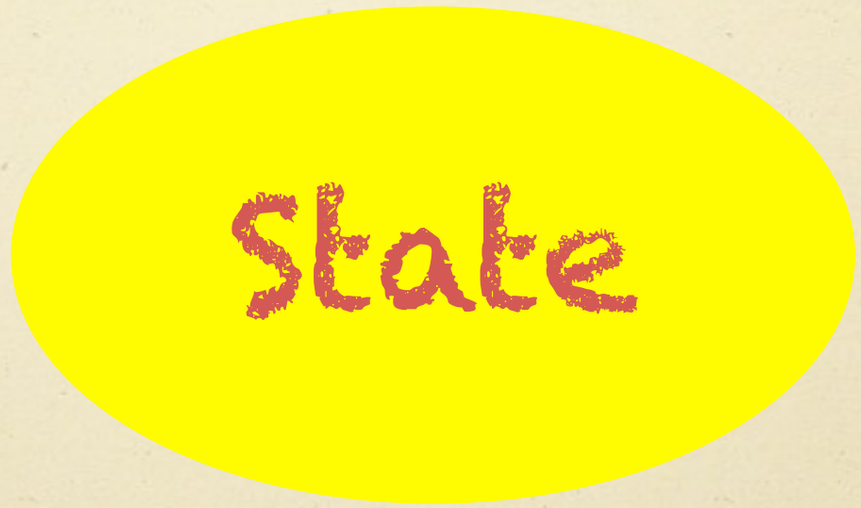
- Classical (sequential)
- Parallel (lock-step)
- Cellular (distributed control)
- Interactive (message passing)
- Analog (continuous time)

Generic Classical Algorithms

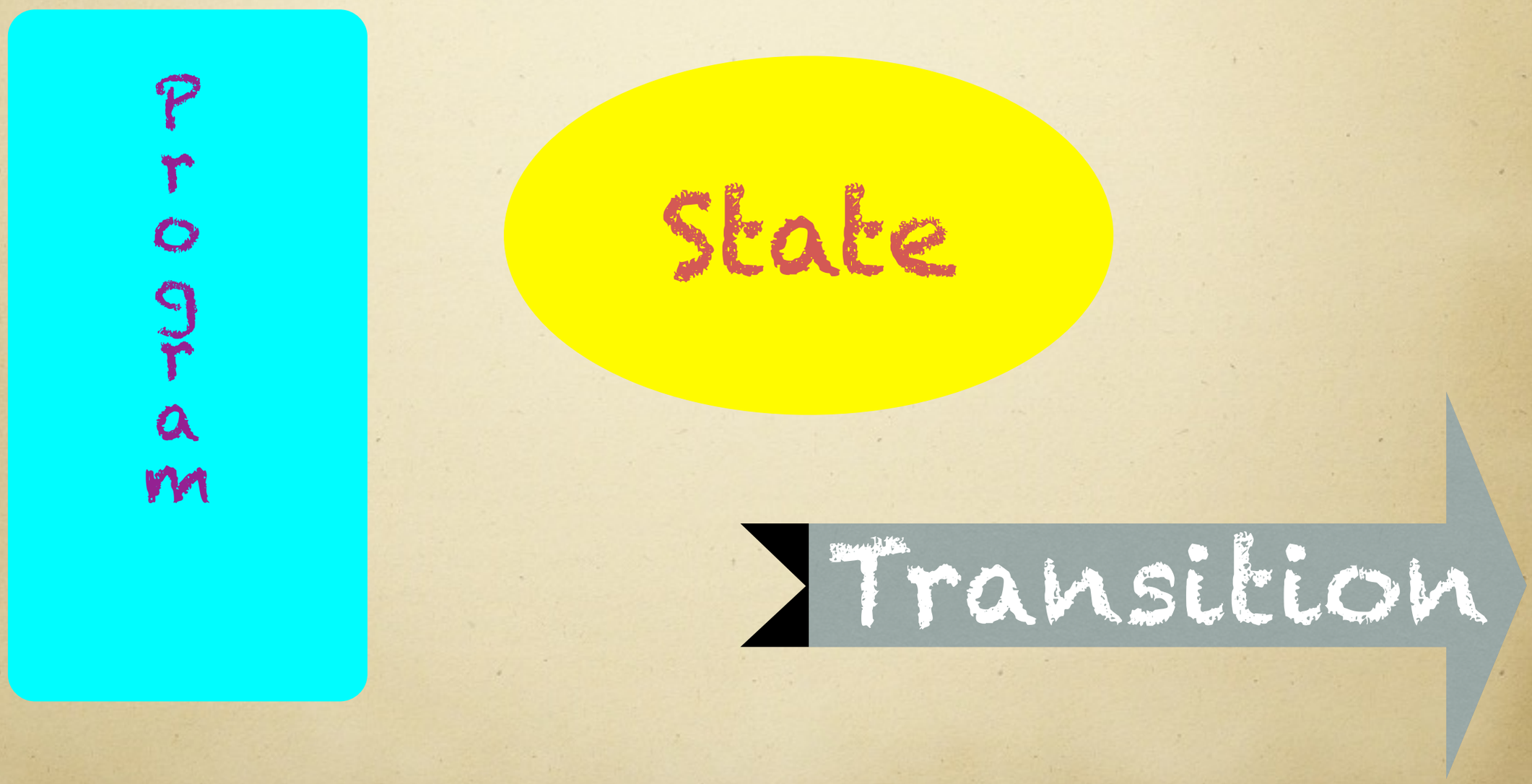
Knuth (1966)

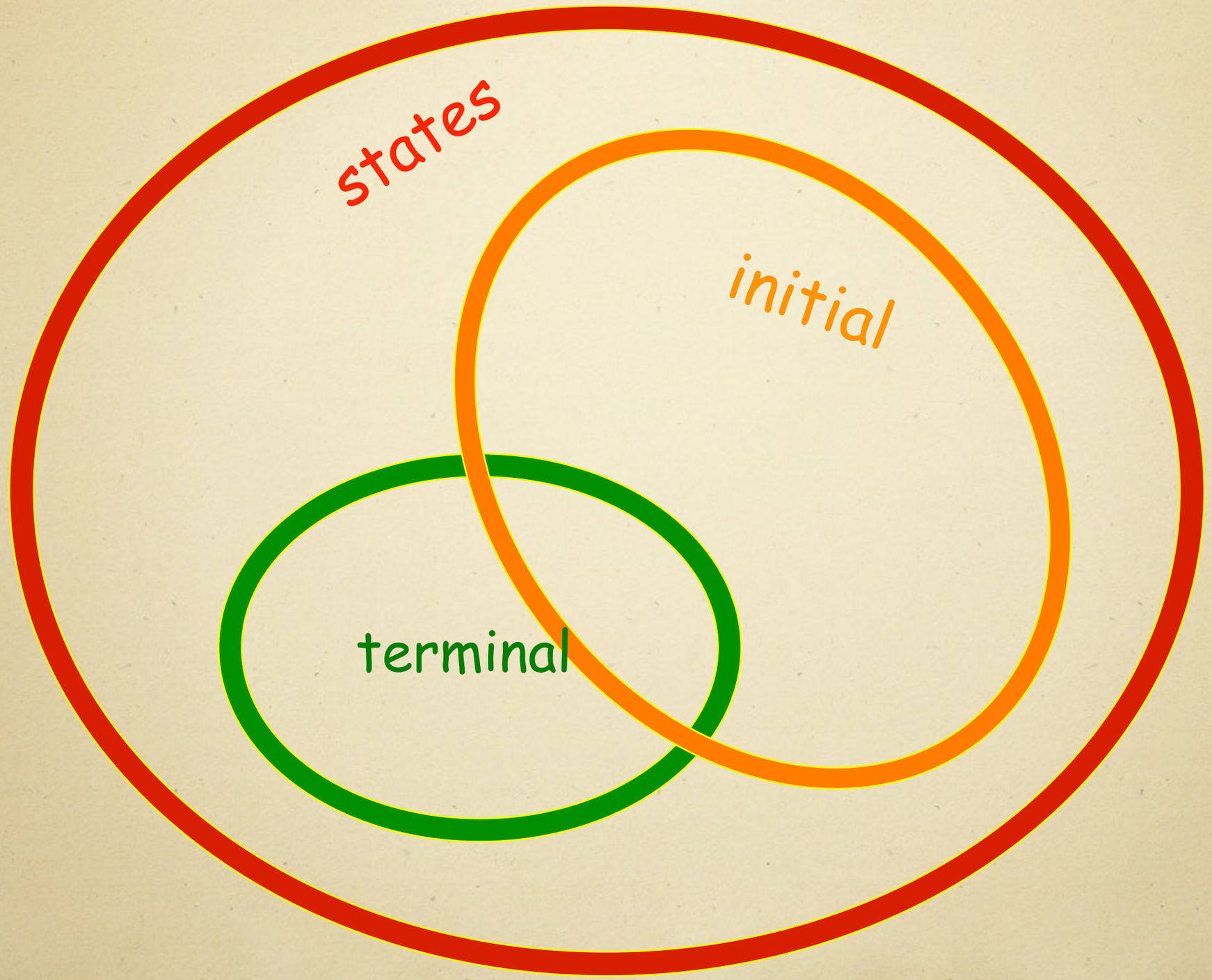
A computational method comprises a set of states…
In this way we can divorce abstract algorithms from particular programs that represent them.

Transition System



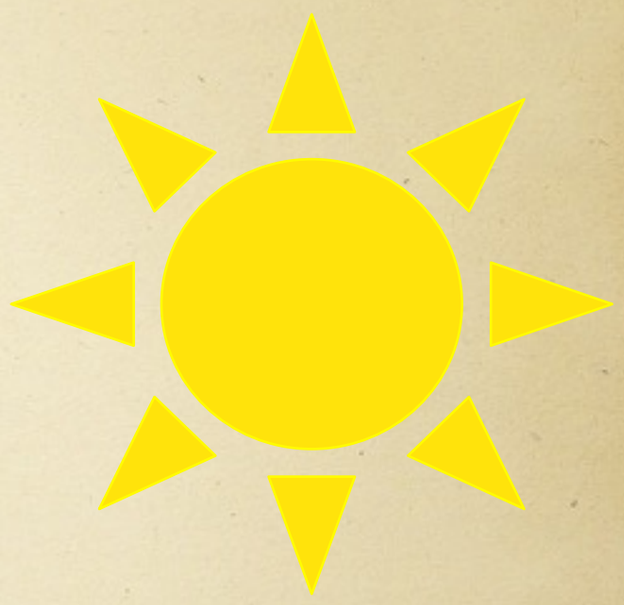
Algorithmic System




states
initial
terminal

Emil Post (1922)

[States] are completely determined by specifying all the properties and relations of its parts… Each … can be completely described [by a conjunction of relations].



State encapsulates all relevant data (and operations)!

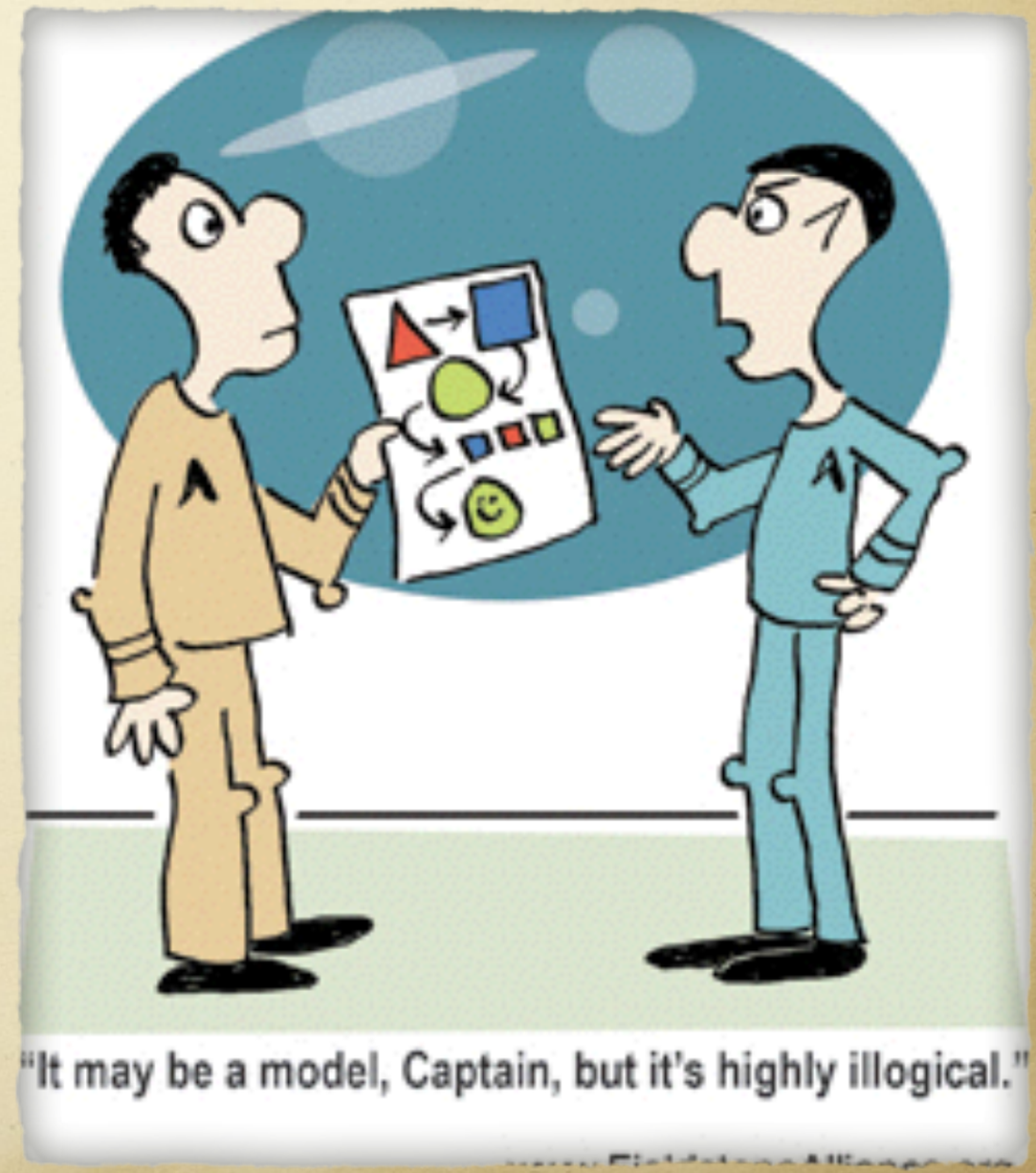
Self-Contained States

Student's Excuse Letter:

"I was working on my home task for Algebraic Topology course for several hours, when I decided to take a short break. I made for myself a cup of coffee with a donut.
But then I spent the whole night, trying to understand, which one of them to eat and which to drink.
That is why I did not finish my home work…

States

- First-Order Structures capture all salient features of state: static operations & dynamic memory



Data Structures

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

Hartley Rogers

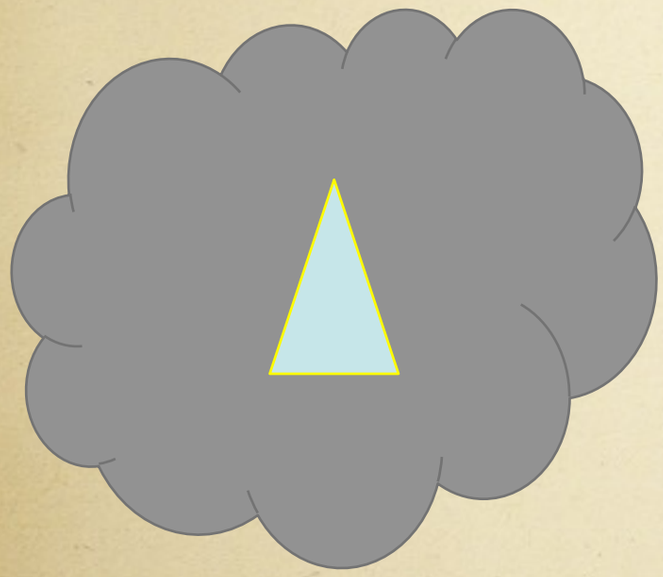
- *Roughly speaking, an algorithm is a clerical (i.e., deterministic, bookkeeping) procedure which can be applied to any of a certain class of symbolic inputs and which will eventually yield, for each such input, a corresponding symbolic output.*

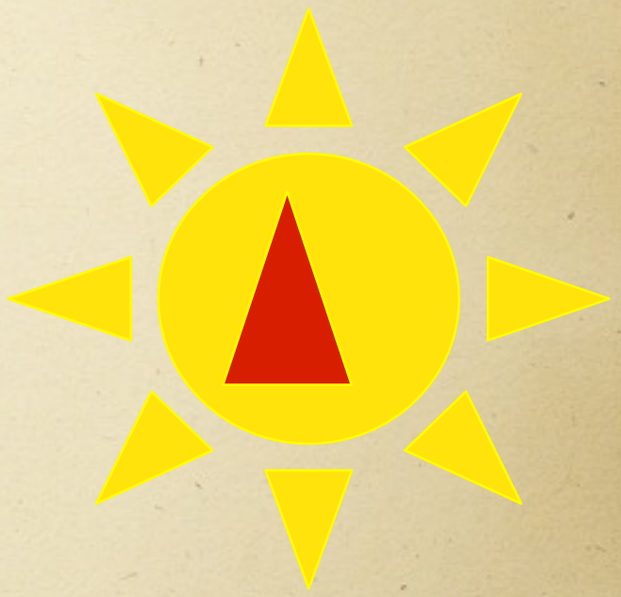
Geometry

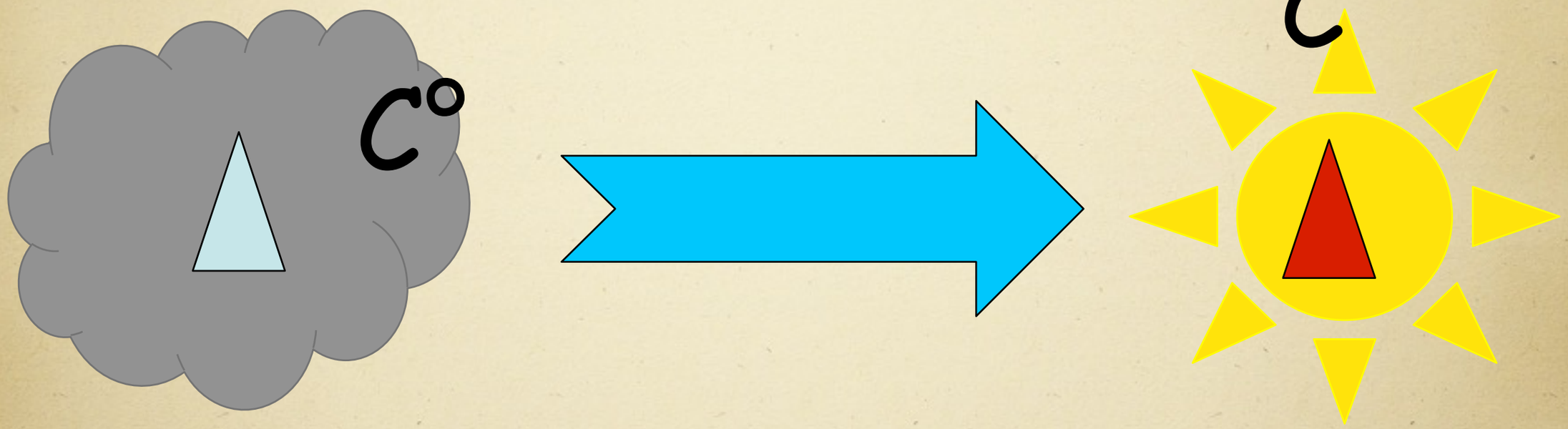
- Domain (underlying set): points, lines, rays, circles, tuples and small bags
- Vocabulary & Operations: Compass; Ruler; =; ∩; Tuple & Bag

Abstract State

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

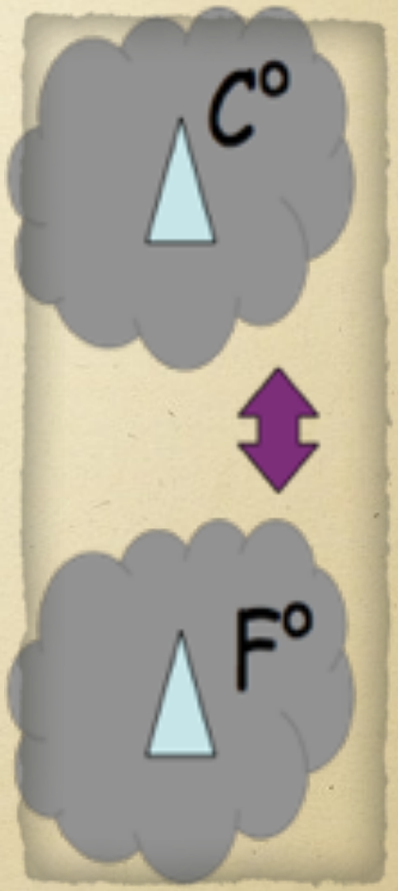




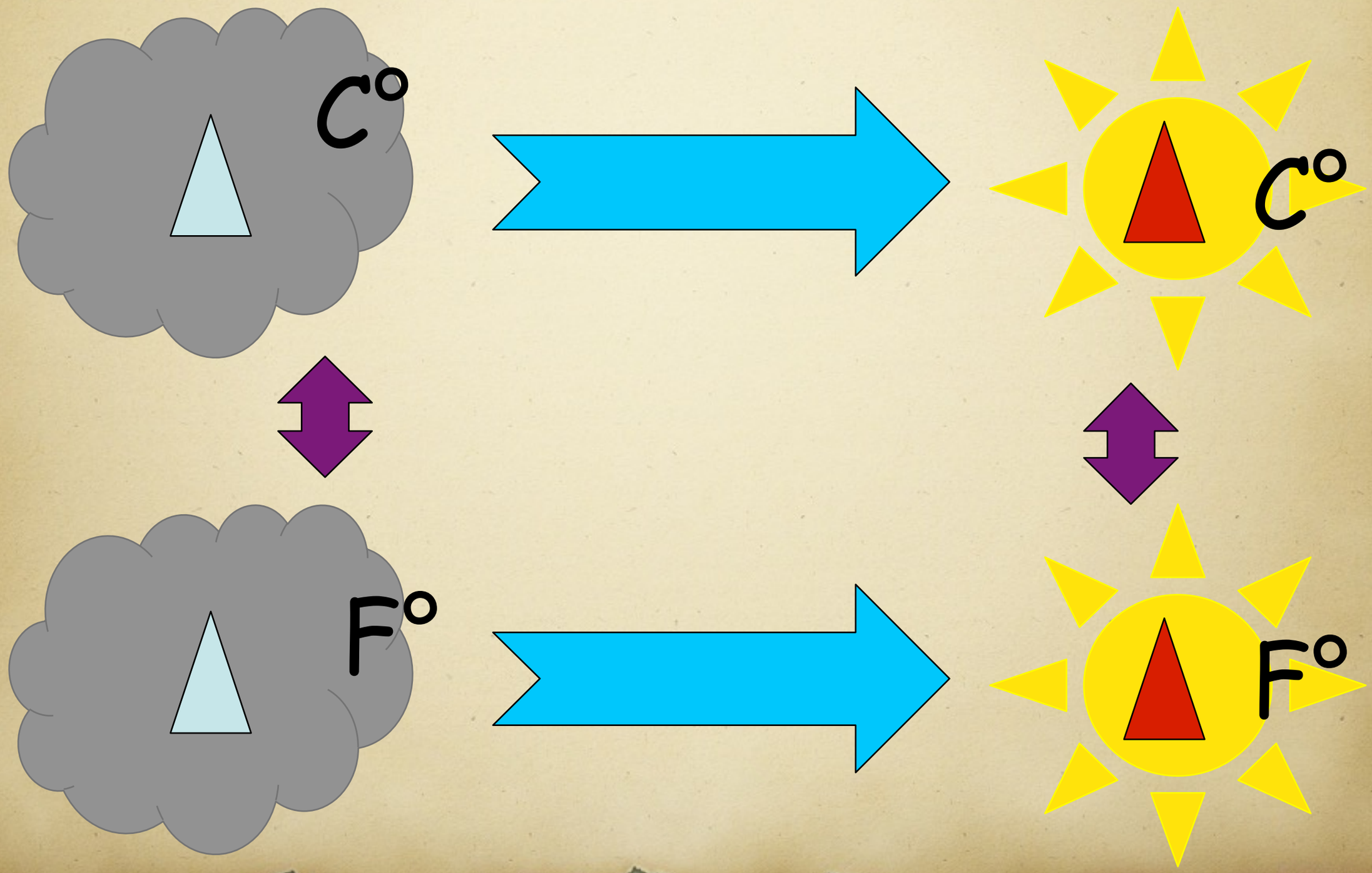

C°
C°

States & Transitions

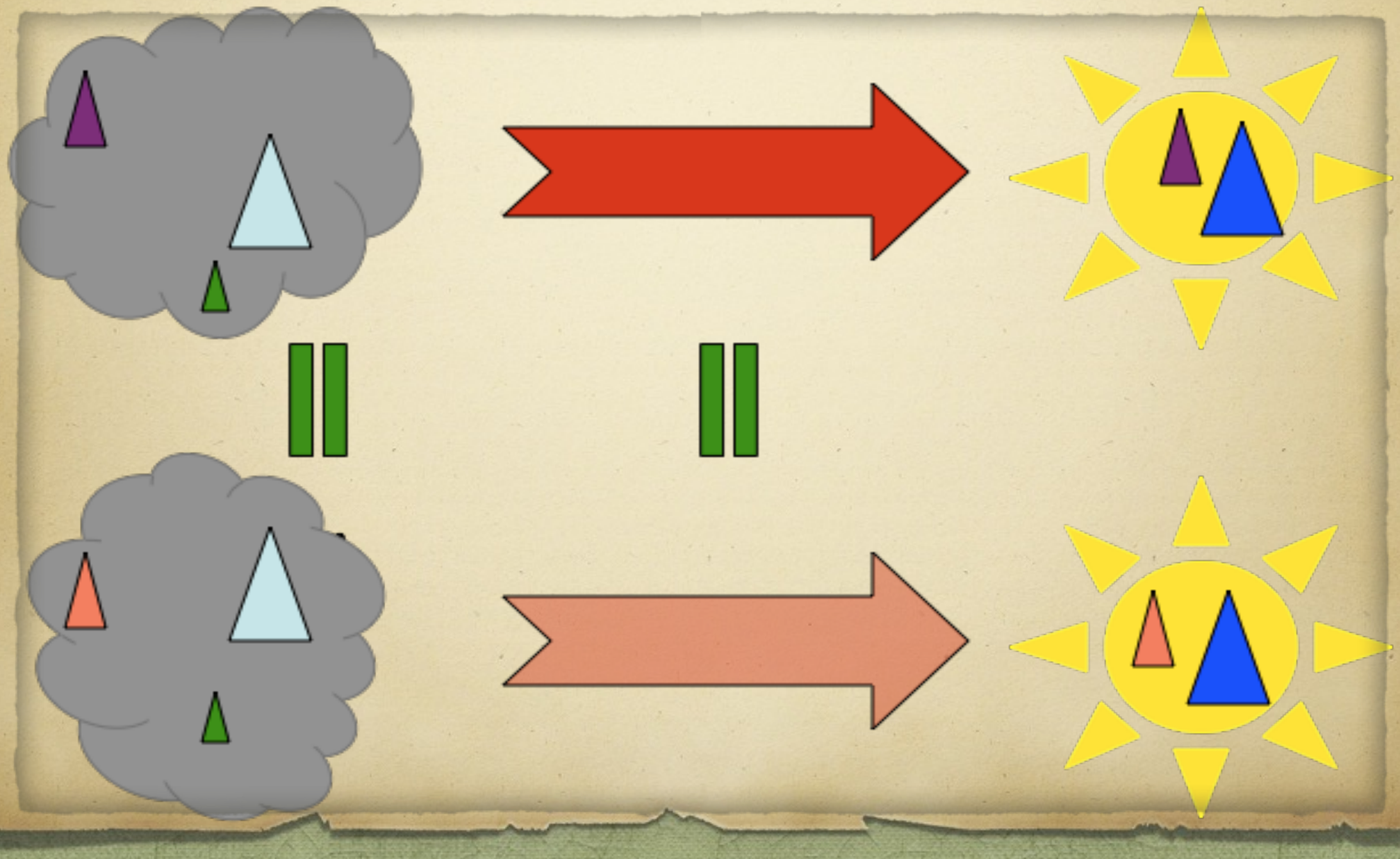
- States are abstract (closed under isomorphism)
- Behavior does not depend on internal representation



Transitions respect isomorphisms



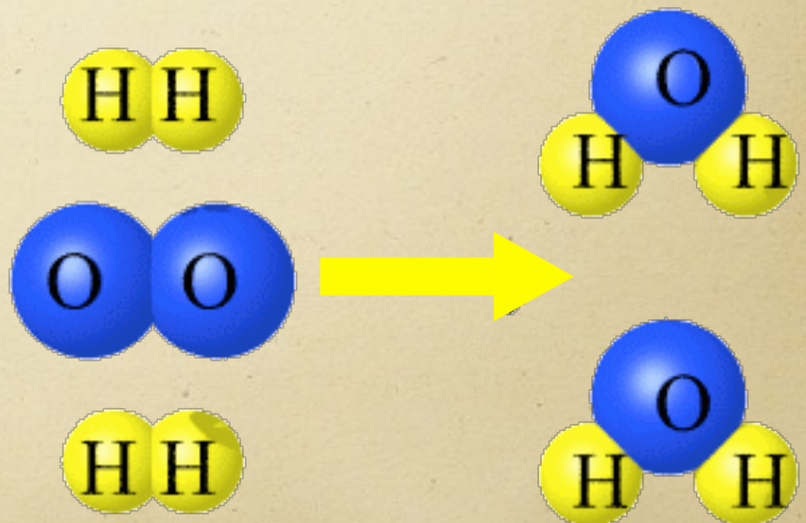
Algorithmic Transitions



States are Abstract

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

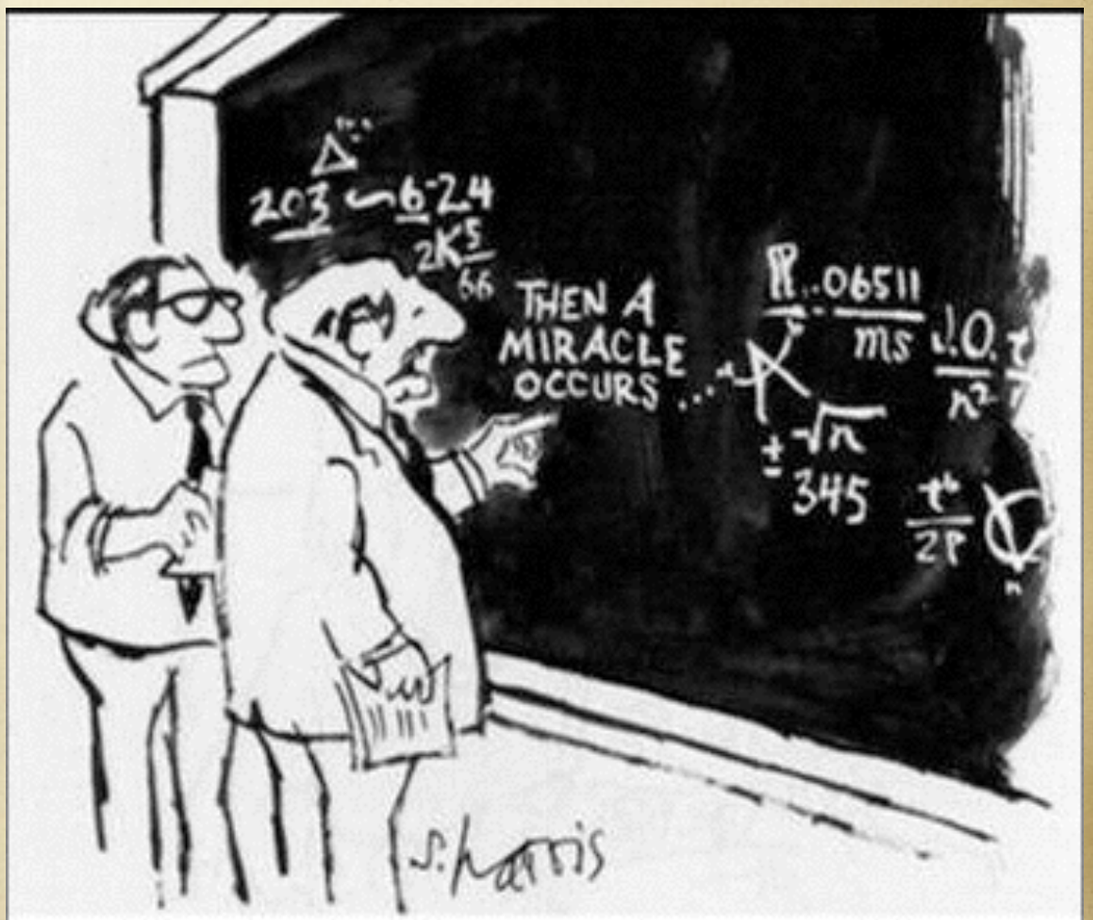
What about Transitions?



Mechanical

A method must be mechanical…

Methods which involve chance procedures are excluded; … methods which involve magic are excluded; … methods which require insight are excluded.
- Joe Shoenfield



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

Émile Borel

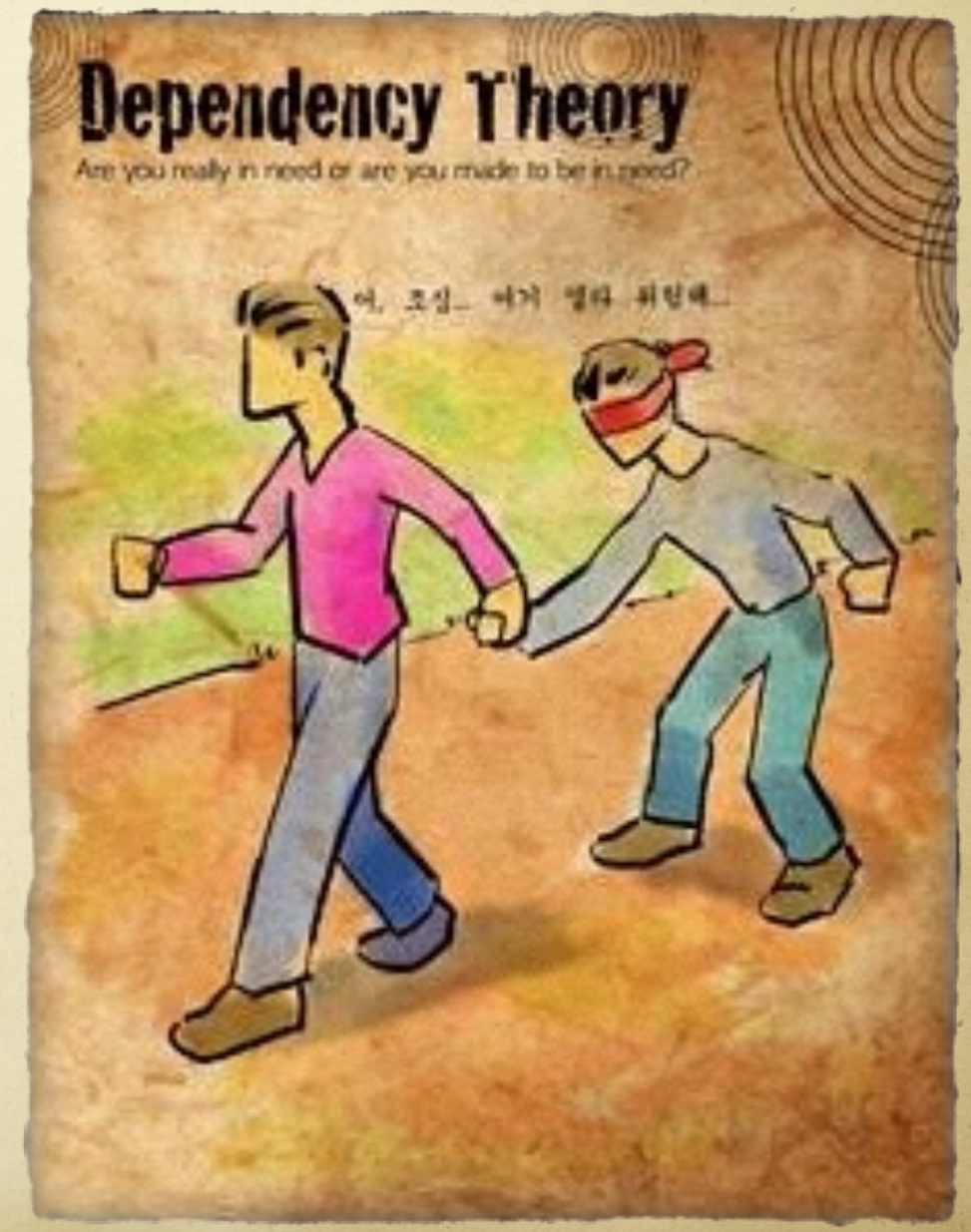
Les calculs qui peuvent être réellement effectues…

Je laisse intentionnellement de côté la plus ou moins grande longueur pratique des opérations l'essentiel est que chacune de ces opérations soit exécutable en un temps fini, par une méthode sûre et sans ambiguïté.

Bounded-in-Time Step

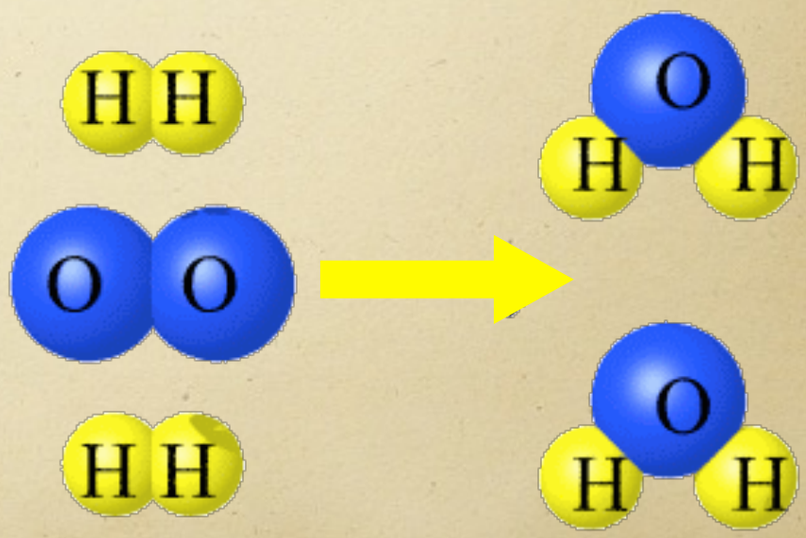

SPEED
LIMIT
∞

Disallowed


Dependency Theory
Are you really in need or are you made to be in need?

What is a Transition?

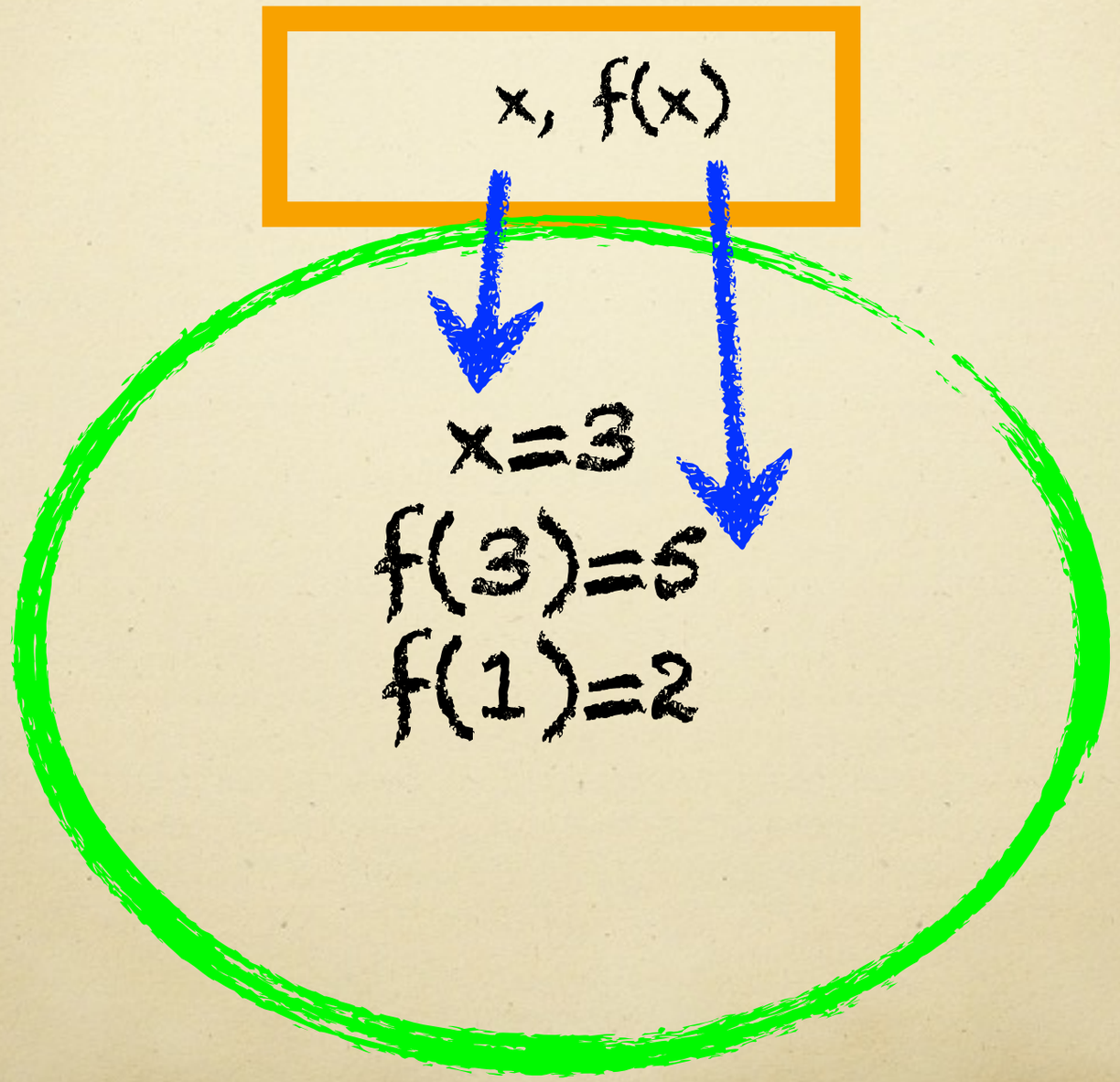
Transitions are algorithmic if they can all be described finitely (without presupposing any special knowledge).



Yuri Gurevich

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

Terms & Locations



Critical Terms

T: x, f(x)

x=3
f(3)=5
f(1)=7

x=1
f(3)=0
f(1)=7

x=3
f(3)=5
f(1)=2

x=1
f(3)=0
f(1)=2

x=3
f(3)=5
f(1)=1

x=1
f(3)=0
f(1)=1

Postulates [Gurevich]

I. An algorithm is a state-transition system

II. Logical structures capture salient aspects of states

III. The transition relation can be described finitely

ASM Theorem [Gurevich]

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

Abstract State Machines

- :=
- ?
- ||
- *

Abstract State Machine

- $f(s1,\ldots,sn) := t$
- *if c then P else Q*
- *do {P1 || ... || Pk}*

Sorting Program

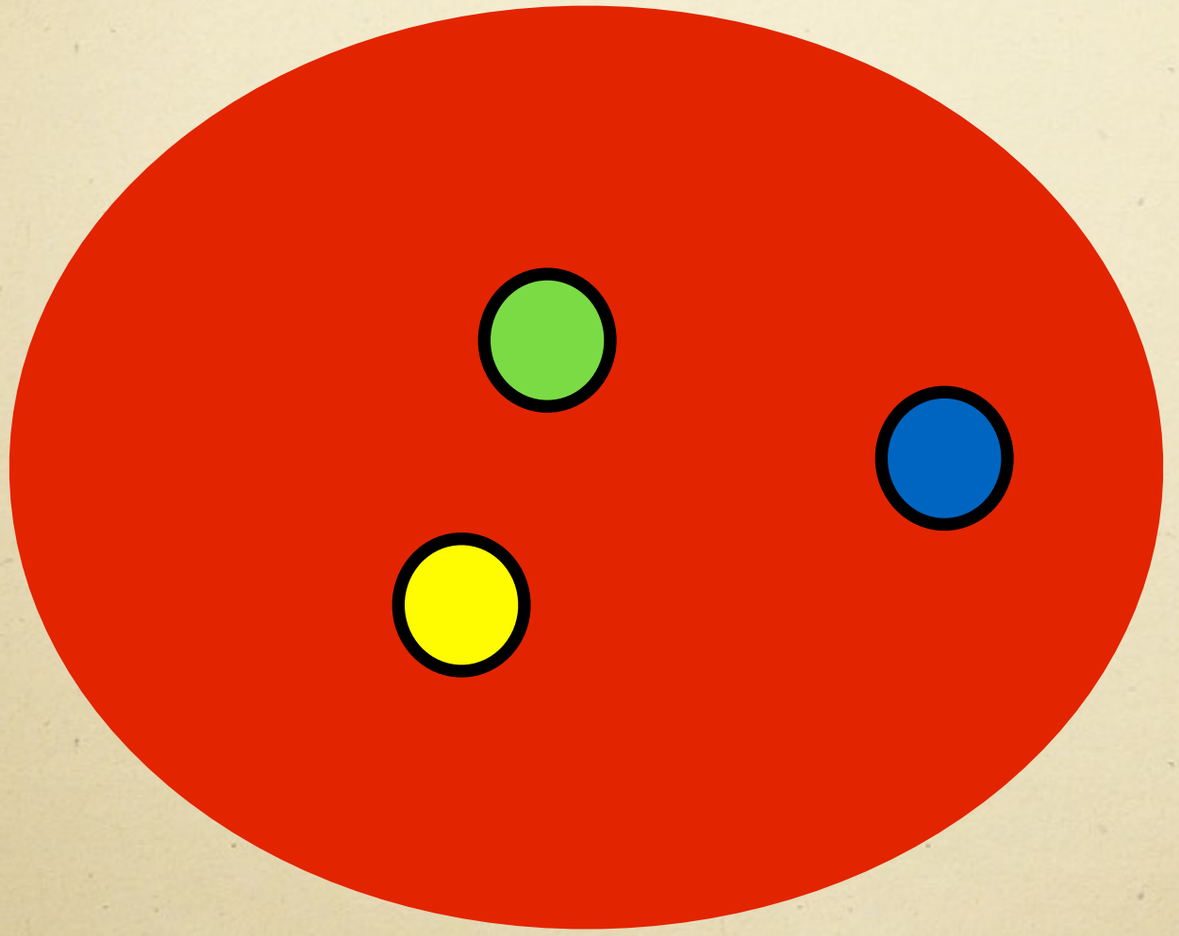
$$\textbf{if } j = n \textbf{ then if } i+1 \neq n \textbf{ then do} \begin{cases} i := i+1 \\ j := i+2 \end{cases}$$

$$\textbf{else do} \begin{cases} \textbf{if } F(i) > F(j) \textbf{ then do} \begin{cases} F(i) := F(j) \\ F(j) := F(i) \end{cases} \\ j := j+1 \end{cases}$$

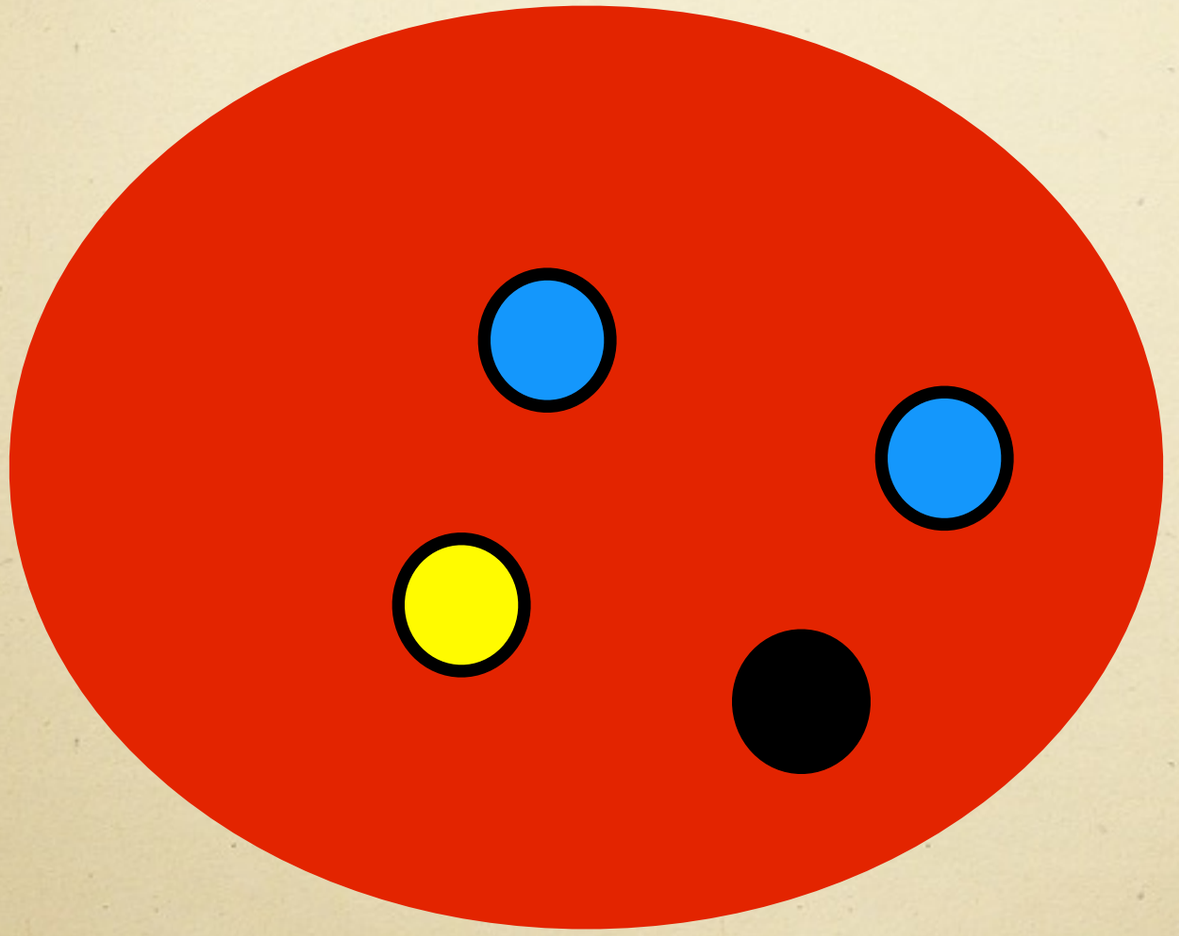
Moreover

- If critical terms determine which locations are accessed, then there is a program that accesses only those locations.

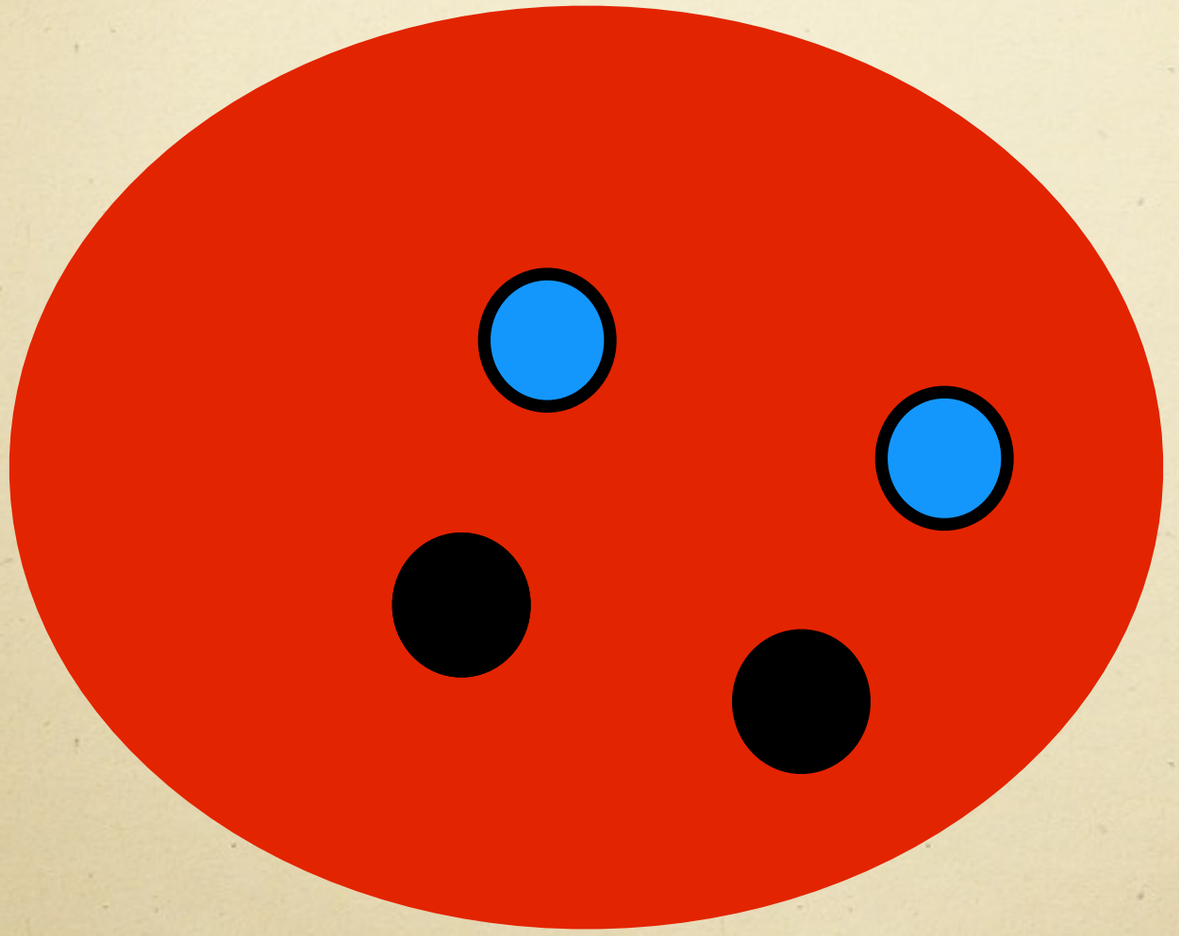
Locations & Values



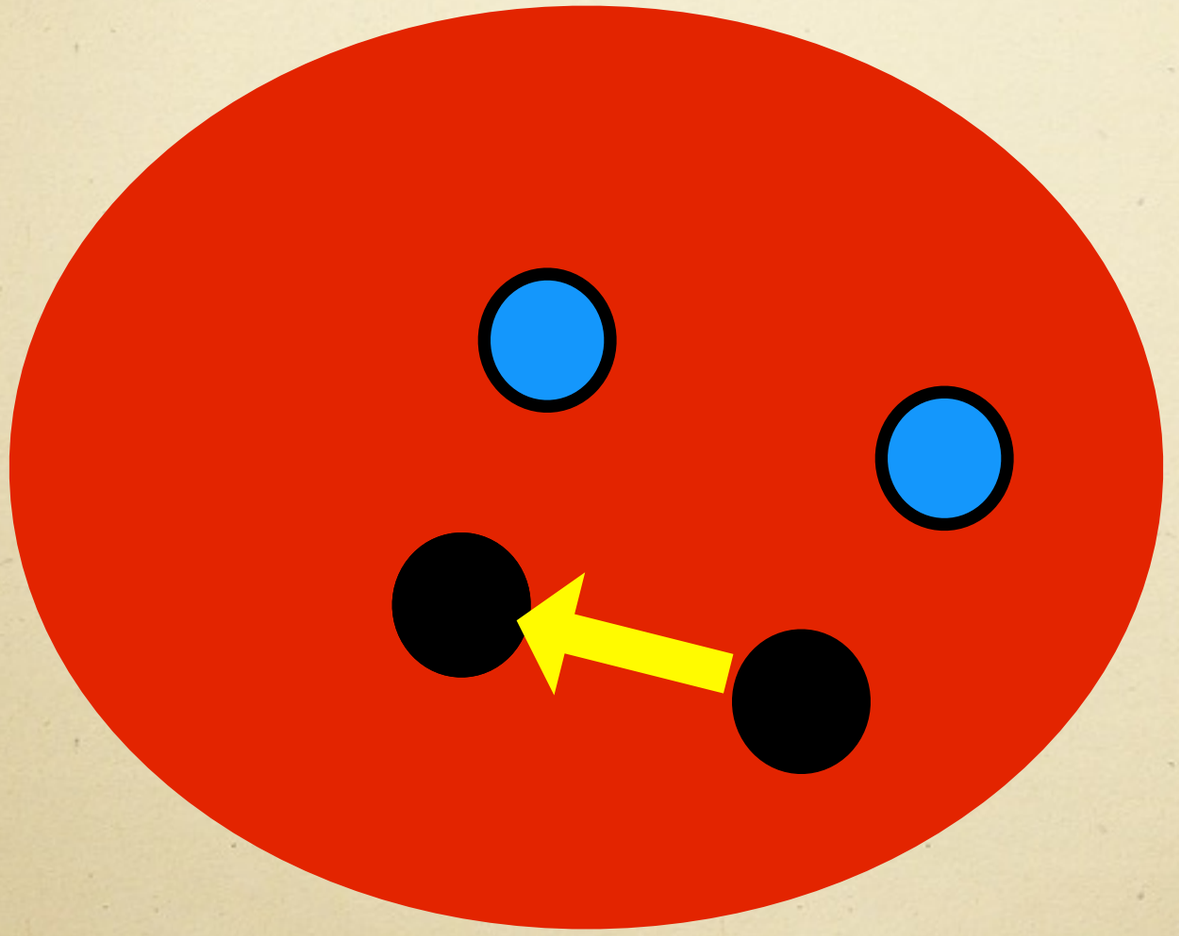
Look



Draw



Move



Generic Parallel Algorithms

Parallel ASM

Andreas Blass

ASMs can be extended to capture unbounded synchronous parallelism

Yuri Gurevich

What is a Parallel Algorithm?

I. Global states with multiple cells, each with a local state.

II. All cells execute same classical algorithm A.

III. In one global step, each cell performs exactly one step of A.

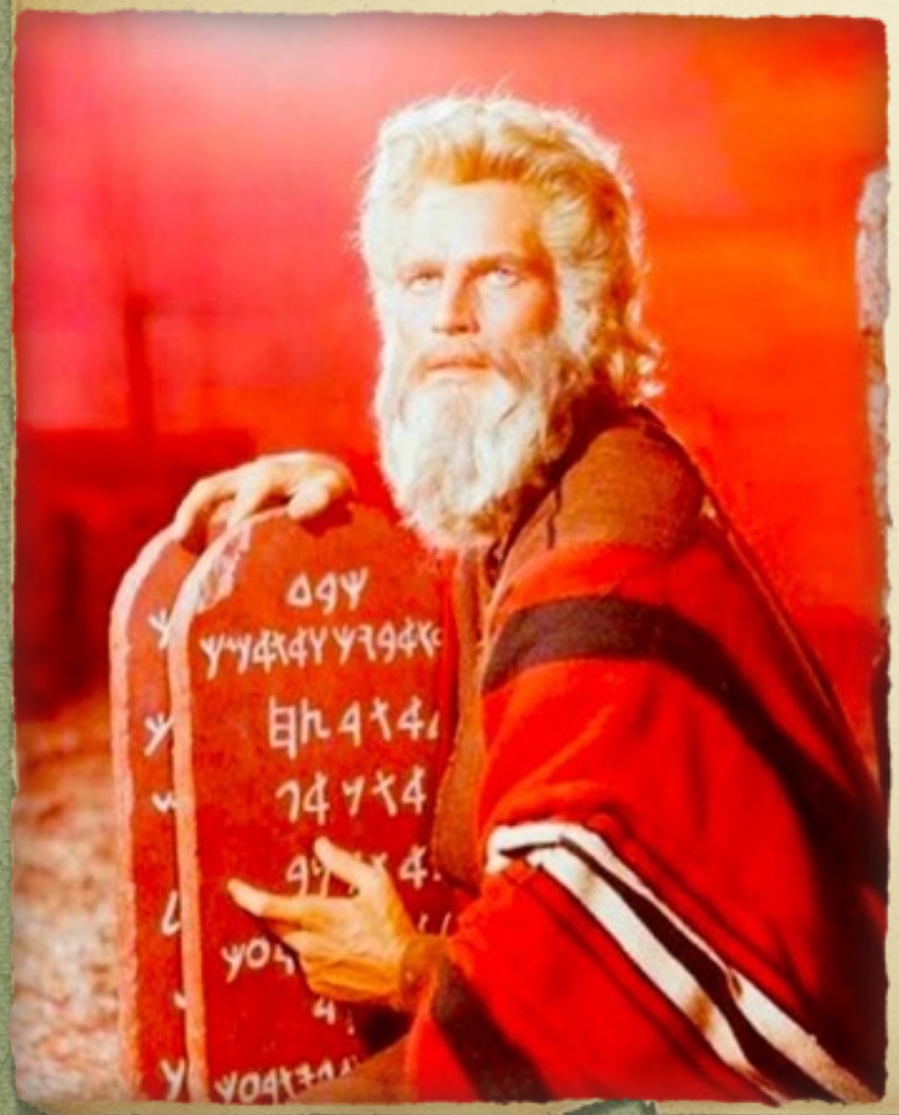
IV. Cells may communicate through either messages or/and shared memory.

V. Cells may create new cells with some initial data.

Postulates

- Each global step is the combined effect of all local cellular steps
- Local transitions can be finitely described in terms of “templates”
- Each cell take full responsibility on its local updates
- Each newborn cell has a unique mother from whom it derives its state
- A mother can have a bounded number of children in

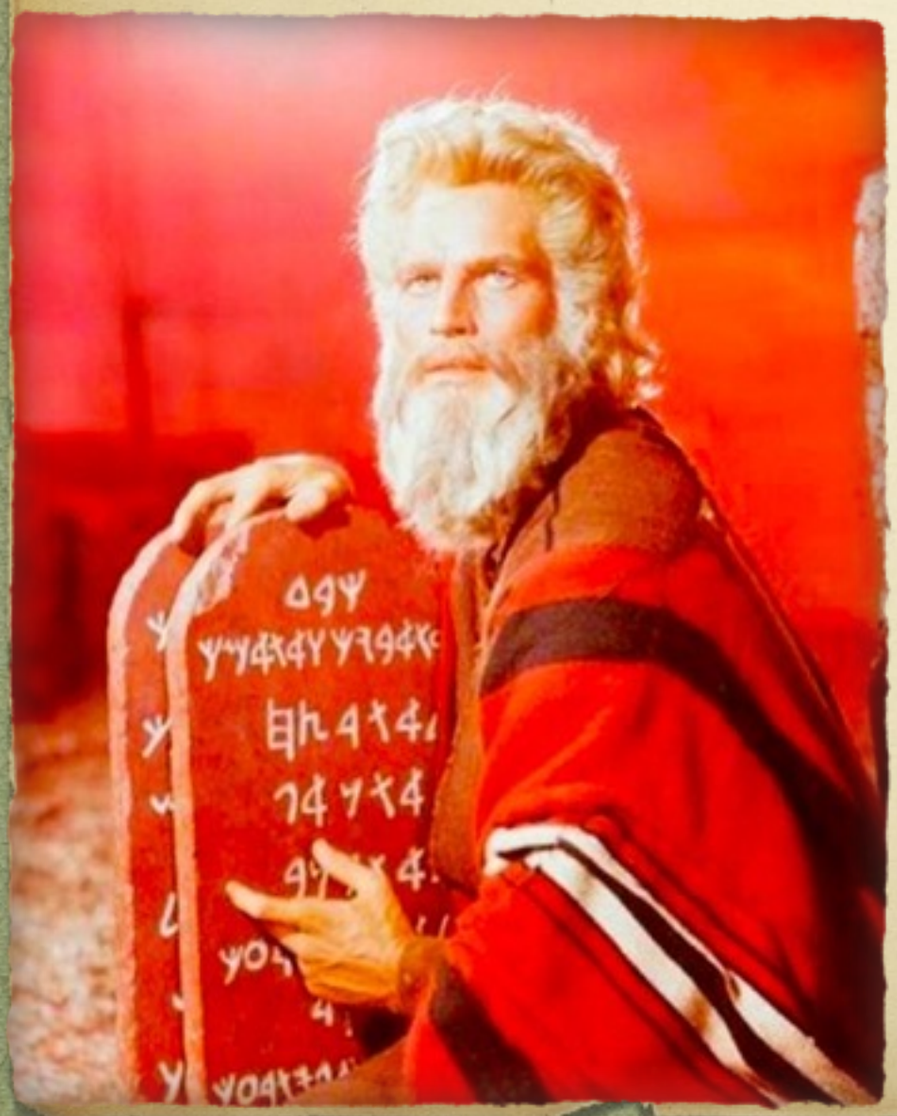
Cells…



Data

Cells…

Powers



Data

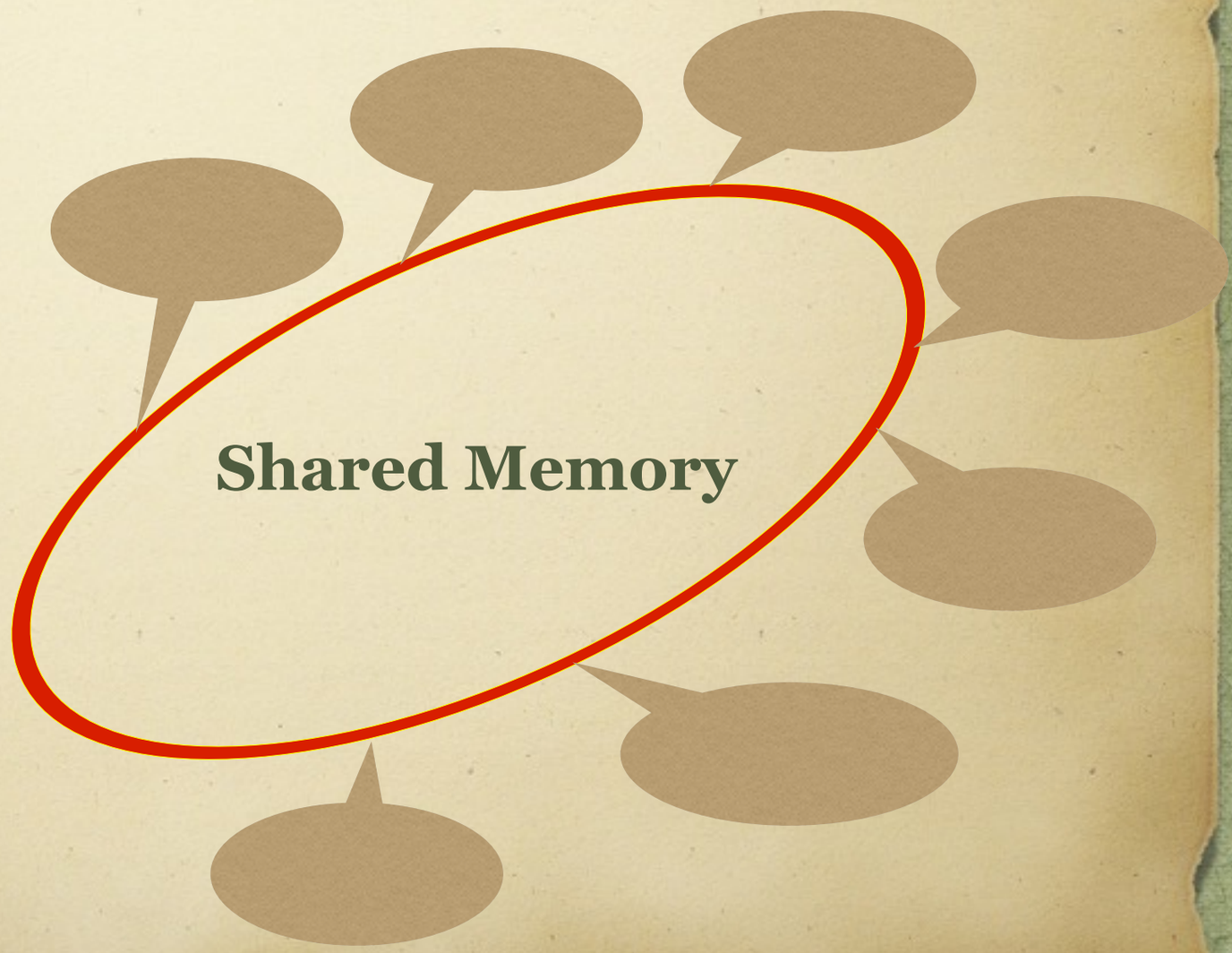
A parallel process is algorithmic if its local transitions (childbirth included) have a finite description

Informal View

I. State:

1. Shared Memory

2. Local Cells



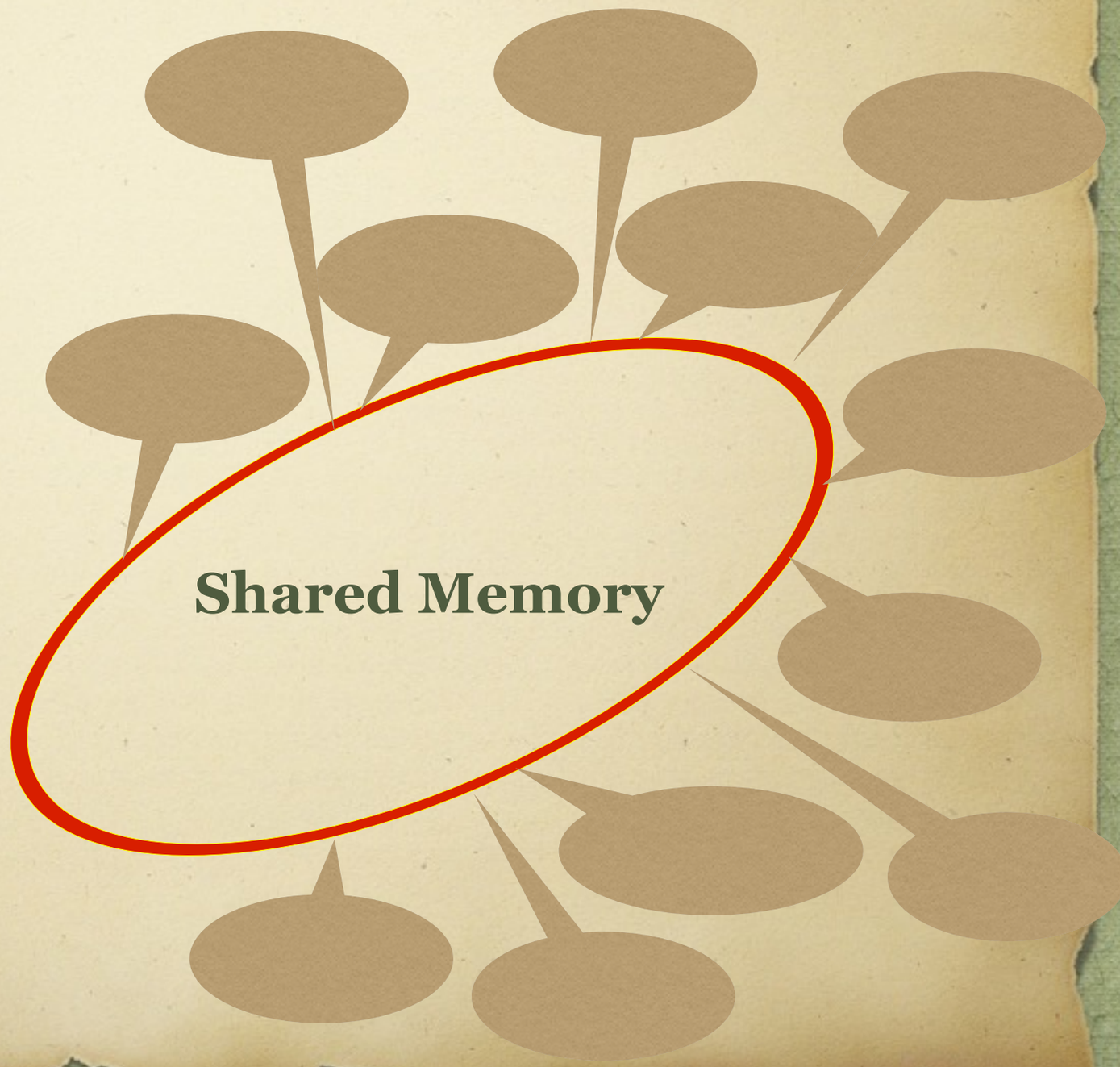
Informal View

II. Evolution:

1. One cell in one step:

   Read/write shared & local memory

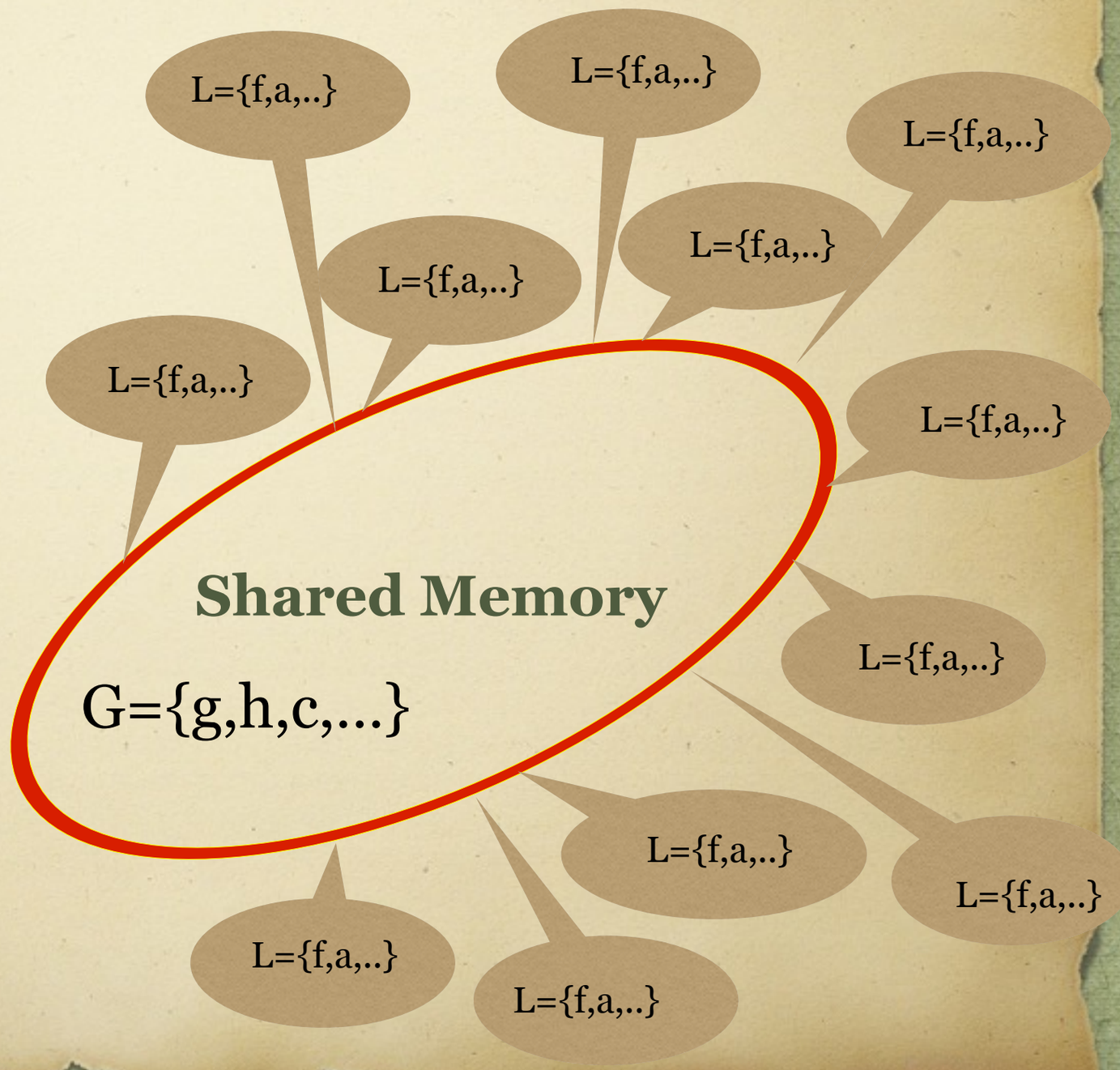
   Create new cells with some initial data



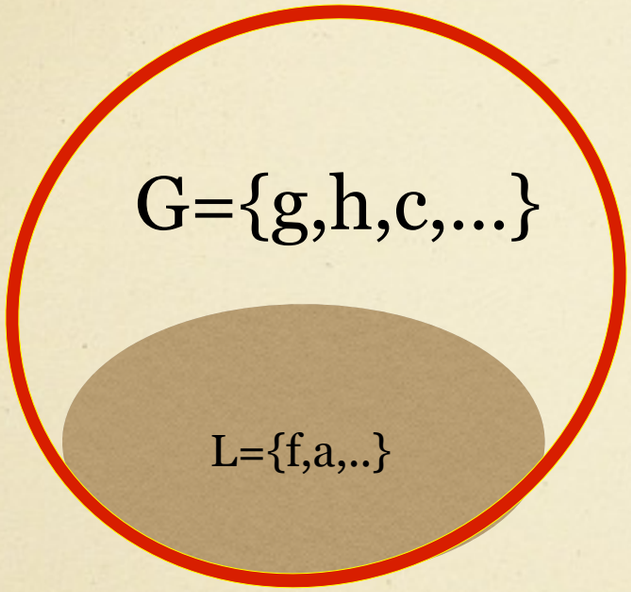
Formal View

I. State - Algebra:

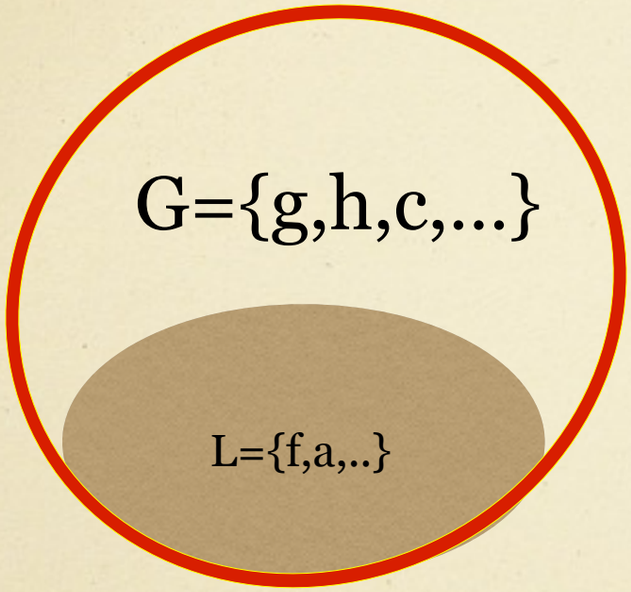
1. Shared Memory
   - Global Vocabulary
2. Local Cells - Local Vocabulary



Formal View

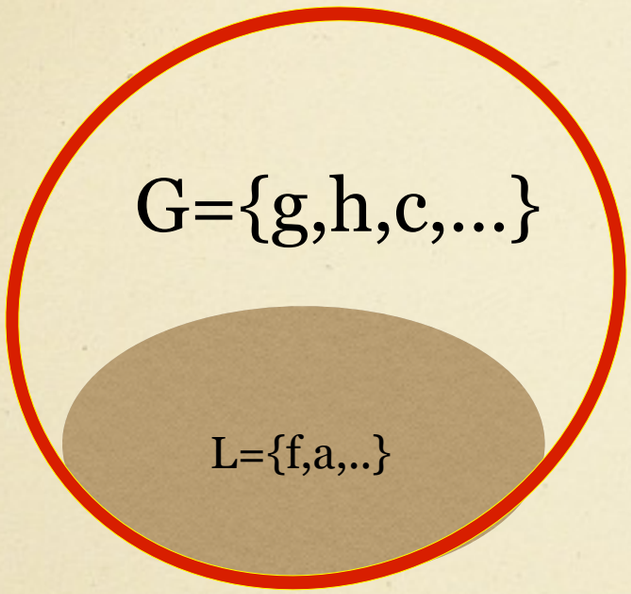


Formal View

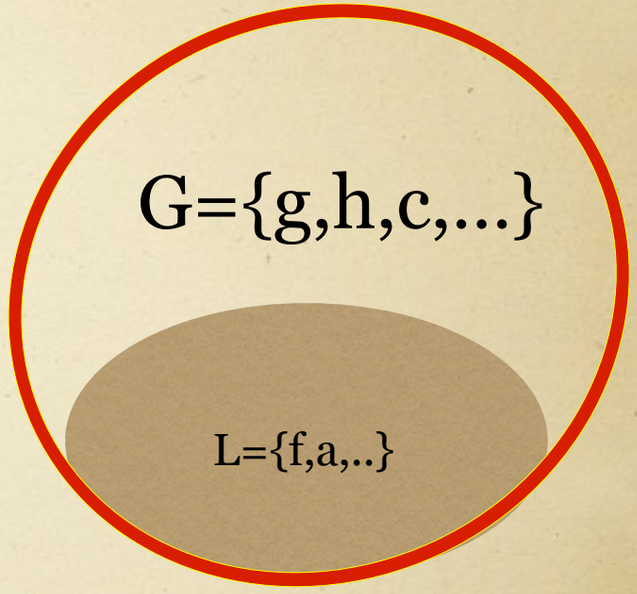


Algebra over G∪L

Formal View

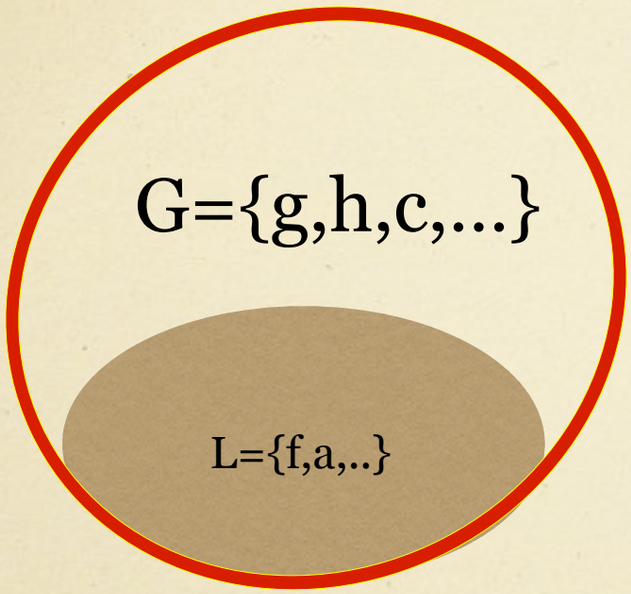


Algebra over G∪L

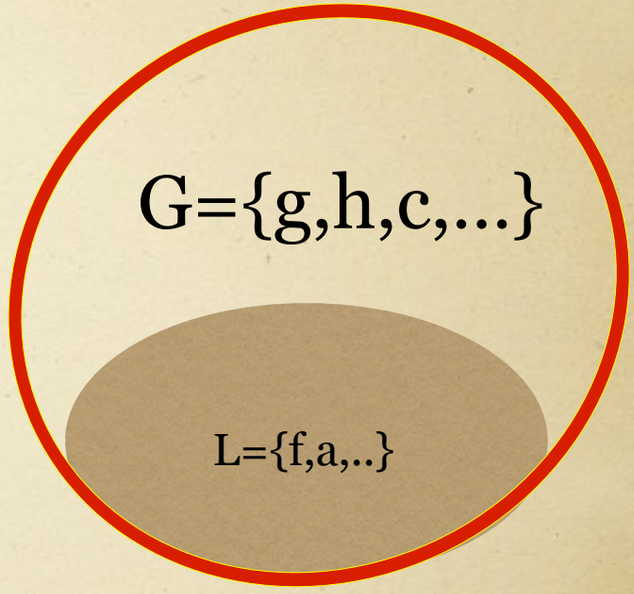


Algebra over G∪L

Formal View



Algebra over G∪L

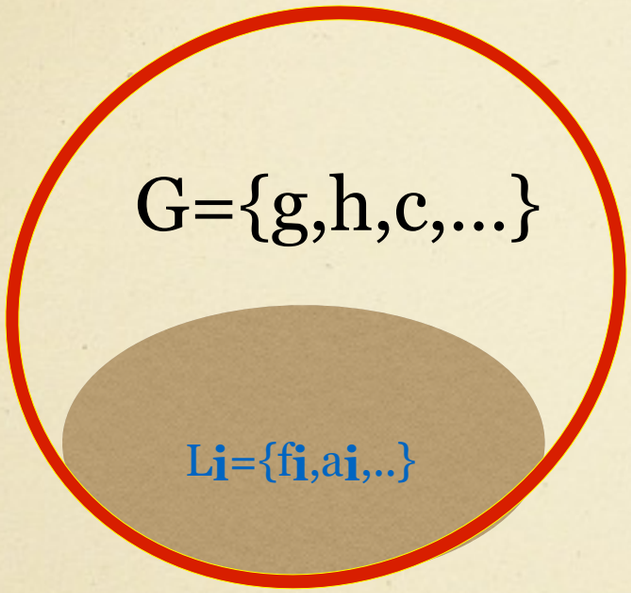


Algebra over G∪L

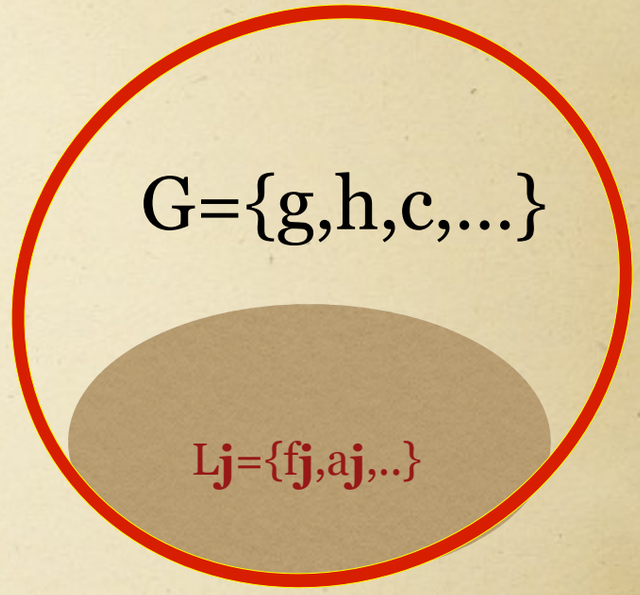
Wanted: *Algebra describing global state (with many cells)*

Formal View

Global algebra over $G \cup L_i \cup L_j$



Algebra over $G \cup L_i$



Algebra over $G \cup L_j$

Wanted: Algebra describing global state (with many cells)

Solution: Unique colors (ids) for cells

The Global State

Is an algebra over the vocabulary (possibly infinite):

$V = G \cup F^*$,

$F^* = \bigcup_{i \in \mathtt{I}} F_i$,

$F_i = \{f_i^1, \ldots, f_i^k\}$

$\mathtt{I}$ is a set (class) of any cardinality

$G \cup F_i$ describes i-th cell (i-th local statte)

Parallel Algorithm

- Is a state transition system of global states, all over the same vocabulary, satisfying axioms of abstract state, “algorithmicity” and:

Templates

- Are terms over unadorned vocabulary $G \cup \{f^1, \ldots, f^k\}$
- For each $i \in I$ a template t induces a term t_i, obtained by substituting each f^j with f_i^j
- We say that $X \equiv_T Y$ if $t_i X = t_i Y$ for any $t \in T$ and any $i \in I$

Algorithmicity

- There exists a finite set of templates T such that
  - $X \equiv_T Y$ then $\Delta X = \Delta Y$

Postulates

- Each global step is the combined effect of all local cellular steps
- Local transitions can be finitely described in terms of "templates"
- Each cell take full responsibility on its local updates
- Each newborn cell has a unique mother from whom it derives its state
- A mother can have a bounded number of children in

Localizations

- Xi is a local i-th cell of global state X, (restriction of X to $G \cup F^i$

  - $\Delta_i X$ - Local updates of i-th cell of global state X

Globality Postulate

- Each global step is the combined effect of all local cellular steps:

- $\Delta X = \cup \Delta_{i \in I} Xi$

Locality Postulate

Each cell take full responsibility on its local updates

- $X_i \equiv_T Y_j$ then $\Delta_i X = \Delta_j Y$

Fertility Postulate

- A mother can have a bounded number of children in a single step:
  - There is $n \in \mathbb{N}$ such that:
    if X is a global step with only one cell,
    then ΔX has at most $n+1$ cells

Motherhood Postulate

- Each newborn cell has a unique mother from whom it derives its state

- If Xi is a trivial cell and $\Delta_i X$ is non-empty then there is $j \in I$ such that $\Delta_i X \subseteq \Delta X_j$

Parallel ASM Theorem

- An evolution of any system that satisfies postulates may be described by a parallel ASM program

Abstract State Machine

- $f(s1,\ldots,sn) := t$
- *if c then P else Q*
- *do {P1 || ... || Pk}*

P-Abstract State Machine

- $f(s_1,\ldots,s_n) := t$
- $\mathit{if}\ c\ \mathit{then}\ P\ \mathit{else}\ Q$
- $\backslash new \bullet f(s_1,\ldots,s_n) := t$
- $do\ \{P_1 \| \ldots \| P_k\}$

$$\backslash new \bullet f(s_1, \ldots, s_n) := t$$

Creates a new cell with update
$(f, s_1, \ldots, s_n, t)$

Parallel ASM Theorem

- An evolution of any system that satisfies postulates may be described by a parallel ASM program

Proof Sketch

- $(S,S_0,\boldsymbol{\tau})$
- T
- We want: $P(X)=\boldsymbol{\tau}(X)$ for any state X
- By globalization: $\boldsymbol{\tau}(X)=\cup\boldsymbol{\tau}(Xi)$
- Conclusion: it is enough to prove:
  - $P(X)=\cup P(Xi)$ for any state X
  - $P(X)=\boldsymbol{\tau}(X)$ for any state X with one non-trivial cell

Let's start

- We assume a cell may have at most one child at one step (generalisation is trivial, since fertility requires uniform bound)

- By localization: ids of non-trivial cell and child do not metter

- Assume X has non-trivial i-th cell
- and $\boldsymbol{\tau}(X)$ has non-trivial i-th and j-th cells

Ordinary point of view

- Ordinary states:
  - X: G∪{fi} Ti
  - **τ**(X): G∪{fi, fj} Ti∪Tj

- Updates are described by ordinary ASM program
- We replace updates of Tj with **new** operation
- We replace replace terms with templates(still works for X)

- We have assignment rule A_X s.t. $A_X(X)=\boldsymbol{\tau}(X)$

- We bound the assignment with boolean condition which is True on X:
  - This gives us a P-ASM rule R s.t. $R(X)=\boldsymbol{\tau}(X)$

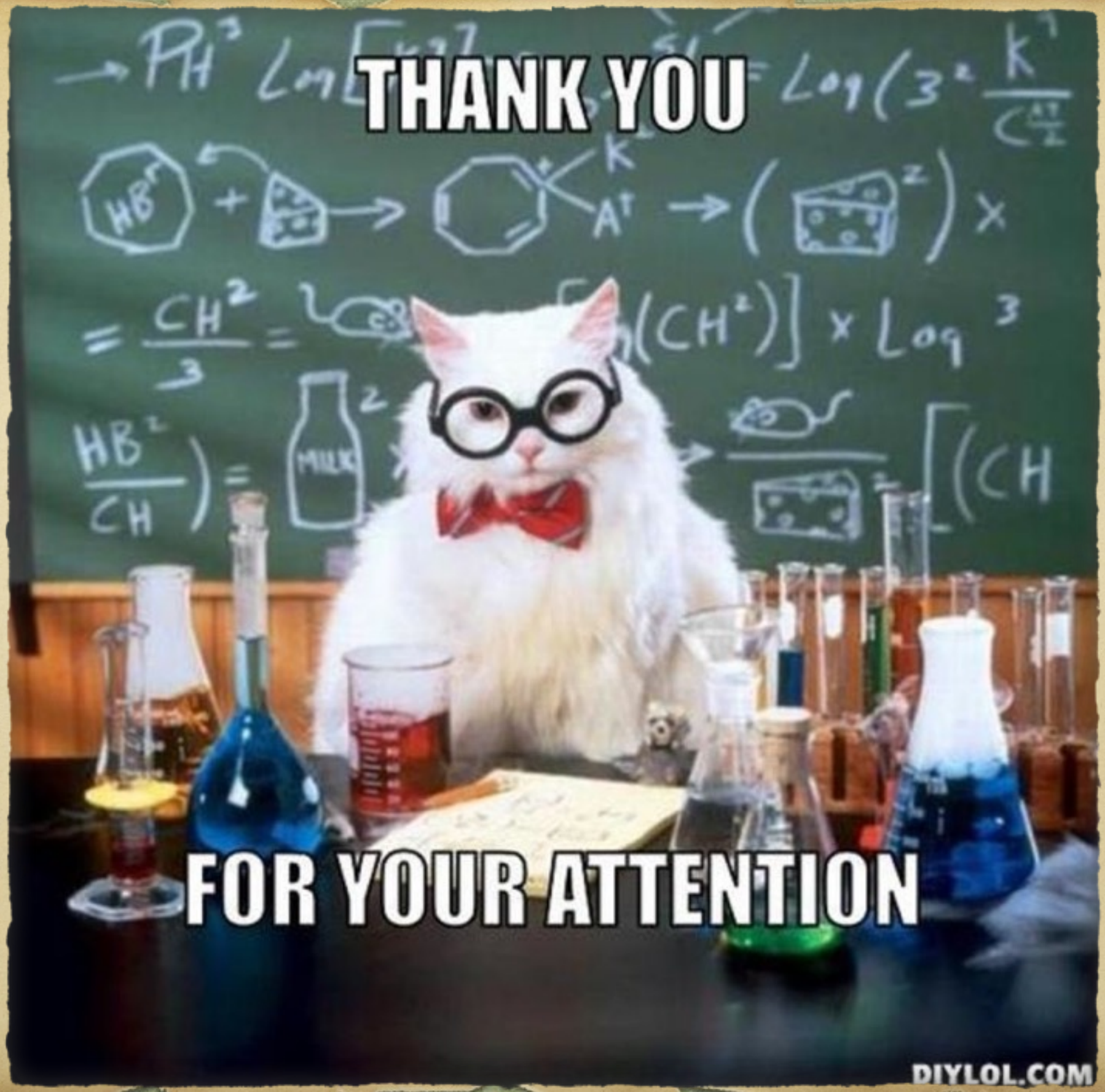
- We want to do that for any possible cell X
- and combine all together by parallel rule

- Obstacle: what if two cells have the same Boolean rule?

- Similar to ordinary case:
  - they will have the same update rule!

Final step

- So P is a union of rules for all cells
- P(X)=∪P(Xi) due to motherhood


THANK YOU
FOR YOUR ATTENTION
DIYLOL.COM