

What is a computer?

I. Overview

II. Details

I. Overview. What is a computer?

- A computer is an
 - **interpreted** ← Ch 3
 - **automatic**
 - **formal system**
 - manipulates tokens (symbols)
 - digital
 - finitely-playable: no magic, no intuition
- Key properties of formal systems:
 - formal equivalence
 - medium independence

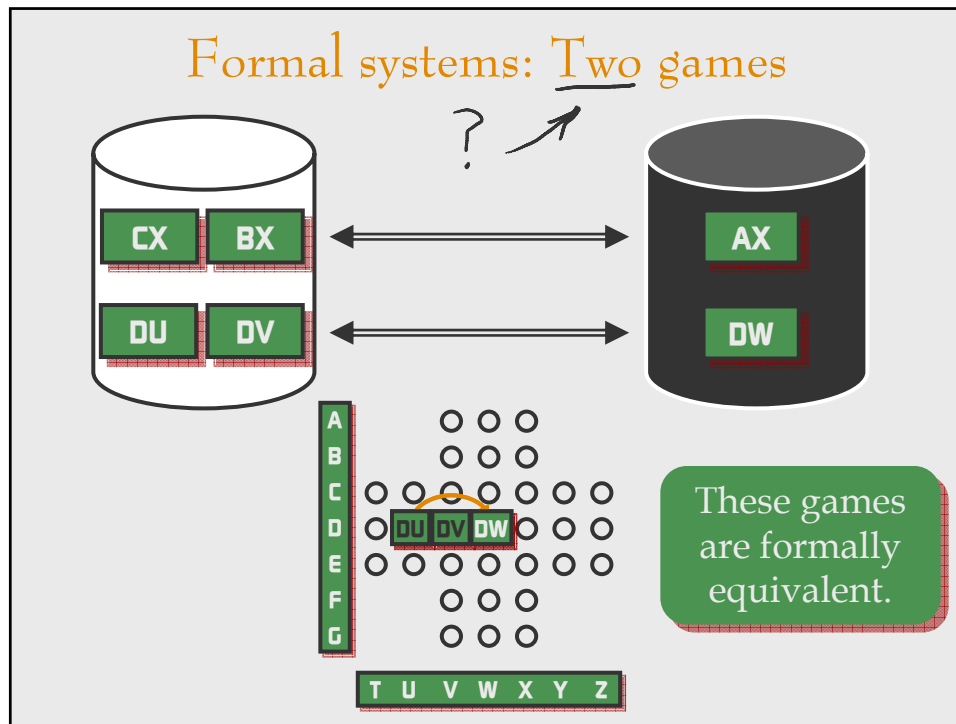
I. Overview. What is a computer?

Central Hypothesis of the Symbolic Theory of Cognition (AI):

- At some abstract level, the *brain* is formally equivalent to a **formal system**.
- That level is the *mind*; that **formal system** is the mental program for cognition.
- Key properties of formal systems:
 - formal equivalence
 - medium independence

Formal systems: mental programs

- Rules for reasoning, arithmetic, solving problems
 - Ⓐ Anderson article
- Rules for generating/interpreting sentences
 - Ⓛ Frank article



Games, brains & computers

Central Hypothesis of the Symbolic Theory of Cognition (AI):

- At some abstract level, the brain is formally equivalent to a formal system.
- A brain is not 'played' by an outside player.
- If the brain is a formal system, it is an *automated* formal system: a game that *plays itself*.
- A computer is an *automatic formal system*.
- [Unlike a game, a computer is an *interpreted* automatic formal system – Ch. 3]

Automating formal systems

- To be automatizable, the formal system defining a computer must be *finitely playable*
- Plan: automate
 - *Players*: make legal moves (changes of the formal system's configuration)
 - *Referee*: tells which player to move, perhaps which tokens to use, start/end configurations
- Trick: the players and referee *are themselves (automated) formal systems* – **algorithms**

II. Details. Formal systems

- A formal system: three properties
 - **Token manipulation**
 - Digital
 - Finite playability
- **Token manipulation**
 - *tokens of different types (two tokens of the same type are subject to exactly the same rules)*
 - *configurations: arrangement of tokens*
 - *initial configuration(s)*
 - *legal sequences of configurations are defined by*
 - *rules: define what moves (configuration changes) are legal given only the current configuration (system is self-contained: tokens' referents, if any, cannot matter)*
 - *final/goal configuration(s)* **Meaning is not a formal property**

II. Details. Formal systems

- A formal system: three properties
 - Token manipulation
 - Digital
 - Finite playability
- Digital system: a set of
 - *positive* and *reliable* techniques for producing ('writing') and reidentifying ('reading') tokens in different configurations

can succeed completely in a read/write cycle, and do so most of the time (tolerance of error)

Examples:
bets/bids using chips vs. sand; chess vs. billiards

II. Details. Formal systems

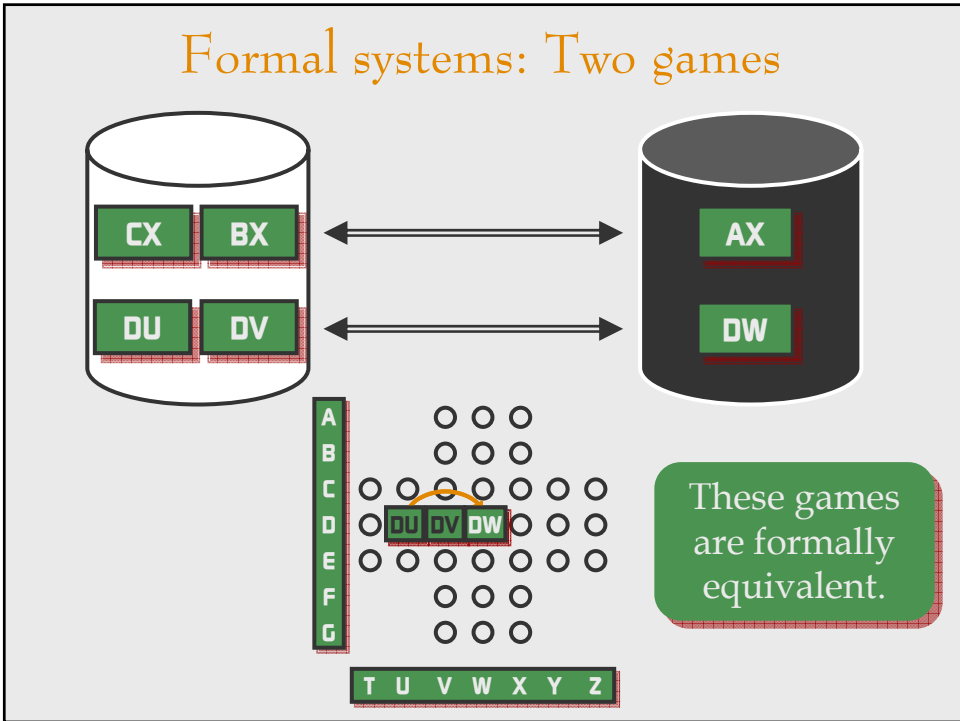
- A formal system: three properties
 - Token manipulation
 - Digital
 - Finite playability
- Finite playability: no infinite or magical powers required to 'play', i.e., to
 - determine if a potential move is legal
 - generate at least one legal move (or determine that no legal move exists)
- Algorithms
- But first: digital token manipulation suffices to define formal equivalence of formal systems (finite playability is required to enable automatization)

Formal equivalence

isomorphism [Galileo? Descartes?]

Two formal systems are *formally equivalent* iff


- distinct configurations in one system are in one-to-one correspondence with distinct configurations in the other system
- a move from one configuration to another is legal in one system if and only if the move between the corresponding configurations in the other system is legal
- a configuration in the first system is a valid starting (or ending) configuration if and only if its corresponding configuration is a valid starting (or ending) configuration in the other



Finite playability & algorithms

- AUTOMATION PRINCIPLE: Whenever the legal moves of a formal system are fully determined by **algorithms**, then that system can be automated.
- An **algorithm** is “an **infallible**, **step-by-step** recipe for obtaining a prespecified result”
 - “**infallible**”: assuming each step is carried out correctly, the procedure succeeds positively in a finite number of steps (or terminates in a finite number of steps, reporting failure, when no result exists)
 - “**step-by-step**”:
 - recipe prescribes one step at a time, in sequence
 - after each step, next step is fully determined (no options or uncertainties)
 - after each step, the next step is “obvious” (no insight or ingenuity required)

‘Finite’ vs. non-‘finite’ playability

- Handwritten spelling-to-picture task
 - A non-formal rule (non-algorithm)
 - Given: a sequence of three shapes *CAT*
 - If the first shape looks like ‘C’ and the next shape looks like ‘A’ and the last shape looks ‘T’ then choose
- 
- A formal algorithm?

Algorithms: Automating formal systems

- Primitive abilities
 - ability to maintain certain types of data
 - ability to perform primitive operations on this data (positively & reliably)
 - ability to follow primitive instructions, i.e., a ...
- Primitive algorithm: recipe for procedure, built from
 - primitive instructions: 'perform operation X ...'
 - straight schedule: '... then go to next instruction'
 - primitive branching instructions: 'next, follow instruction I_0 or I_1 , depending on whether the condition C is false or true'
- Complex algorithm
 - an instruction may be an already-defined algorithm
- Scaffolding of more and more complex algorithms

control

Example algorithm

- Primitive algorithm to build a list l containing n copies of a symbol s : **COPYSYMBOL**
 1. Set $i = 0$
 2. Set $l = \emptyset$
 3. Prepend s to l
 4. Add 1 to i
 5. Is $i = n$?
If yes, stop;
if no, jump to 3
- Example execution: $n = 3, s = \mathbf{X}$

Memory

n :

s :

i :

l :

Example algorithm

- Primitive algorithm to build a list l containing n copies of a symbol s : **COPYSYMBOL**
 1. Set $i = 0$
 2. Set $l = \emptyset$
 3. Prepend s to l
 4. Add 1 to i
 5. Is $i = n$?
 If yes, stop;
 if no, jump to 3
- What are the data types?
 Primitive operations?
 Control mechanisms?
- Complex algorithms

Memory

n :

s :

i :

l :

Players inside of players inside of ...

CHESS MACHINE

Tokens

Chess pieces (outer position)

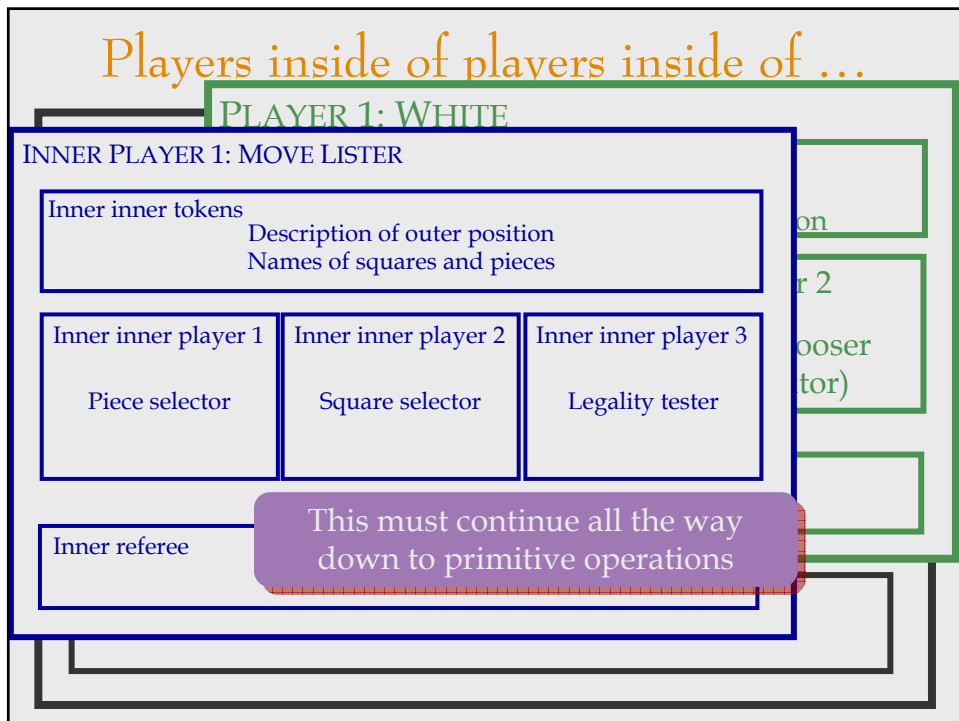
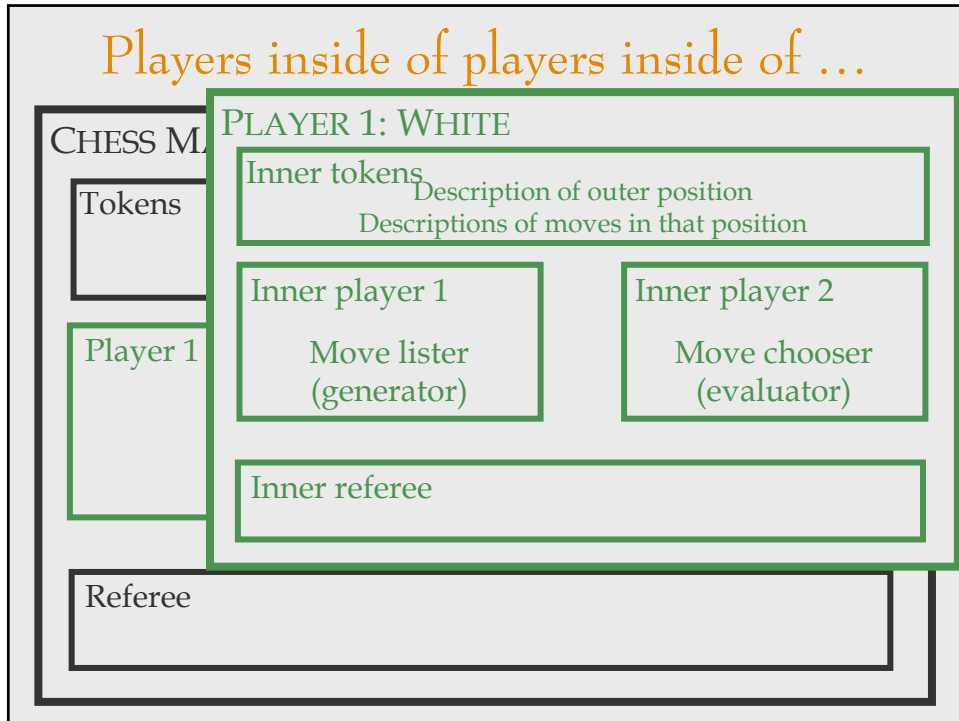
Player 1

White

Player 2

Black

Referee



Finite playability & algorithms

- AUTOMATION PRINCIPLE: If the legal moves of a formal system are fully determined by **algorithms**, then that system can be automated.
- Just need to build a physical system with the:
 - Primitive abilities
 - ability to maintain certain types of data
 - ability to perform primitive operations on this data (positively & reliably)
 - ability to follow primitive instructions
- Then scaffolding of algorithms yields an arbitrarily high degree of complexity

Complex tokens

- Arabic numerals
 - simple tokens: digits
 - a configuration (sequence or 'string') of digits can be considered to be a *complex token*
 - *compositionality* holds: the rules operating on the complex token are completely determined by the configuration of simple tokens that define it
- The formal multiplication game
- Meaning again: Chinese multiplication room?

Review. What is a computer?

- A computer is an
 - **interpreted** ← Ch 3
 - **automatic**
 - achieved by algorithms via Automation Principle
 - **formal system**
 - manipulates tokens (symbols)
 - digital
 - finitely-playable: no magic, no intuition
- Key properties of formal systems:
 - formal equivalence
 - medium independence

Review. What is a computer?

Central Hypothesis of the Symbolic Theory of Cognition (AI):

- At some abstract level, the *brain* is formally equivalent to a **formal system**.
- That level is the *mind*; that **formal system** is the mental program for cognition.
 - finitely-playable: no magic, no intuition
- Key properties of formal systems:
 - formal equivalence
 - medium independence