Lecture 41

Encodings (contd.), Computers vs Turing machine, Halting Problem

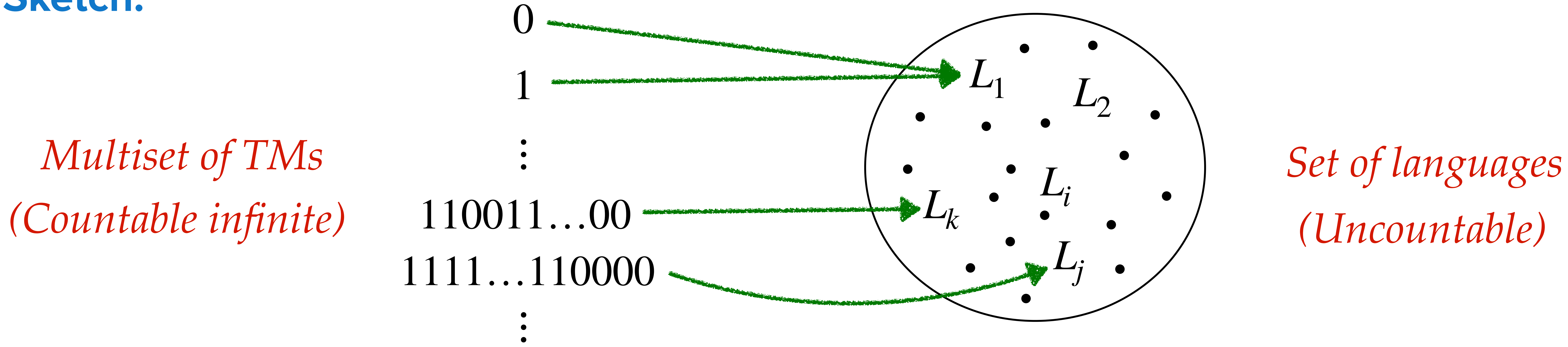
Too Many Problems, Too Few TMs

Notations:

- For any $\alpha \in \{0,1\}^*$, M_α denotes the TM whose encoding is α.
- $\langle M \rangle$ denotes the encoding of a TM M.
- $L(M)$ denotes the language of a TM M, i.e., the set of inputs that are accepted by M.

Theorem: There exist languages that cannot be decided/recognised by any TM.

Proof Sketch:



Computer vs Turing Machine

Simulating a Computer by Turing machine

High-level language programs can be translated into a assembly language program which is a finite sequence of instructions of type:

- Move data from memory into registers or vice-versa.
- Add or multiply the content of two registers into some register.

$\cdots$	$inst_1$	$inst_2$	$\cdots$	reg_1	reg_2	$\cdots$	0	1	1	$\sqcup$	$\sqcup$	$\cdots$

Assembly language program can be simulated in a TM by:

- Allocating portions of tape for registers and memory.
- Storing instructions on tapes.
- Executing instructions using δ.

Computer vs Turing Machine

Simulating a Turing machine by Computer

A C program with infinite memory can be written that simulates a Turing machine where:

- An infinite array can act as the tape of the TM.
- Transition function's entries can be stored in a finite 2-dimensional array.

Equal Power but Different Roles

- High-level languages are used to demonstrate an effective procedure that decides a given language because they are user-friendly.
- Turing machines are used to prove non-existence of an (efficient) effective procedure that decides a given language because of their simple mathematical structure.

Halting Problem

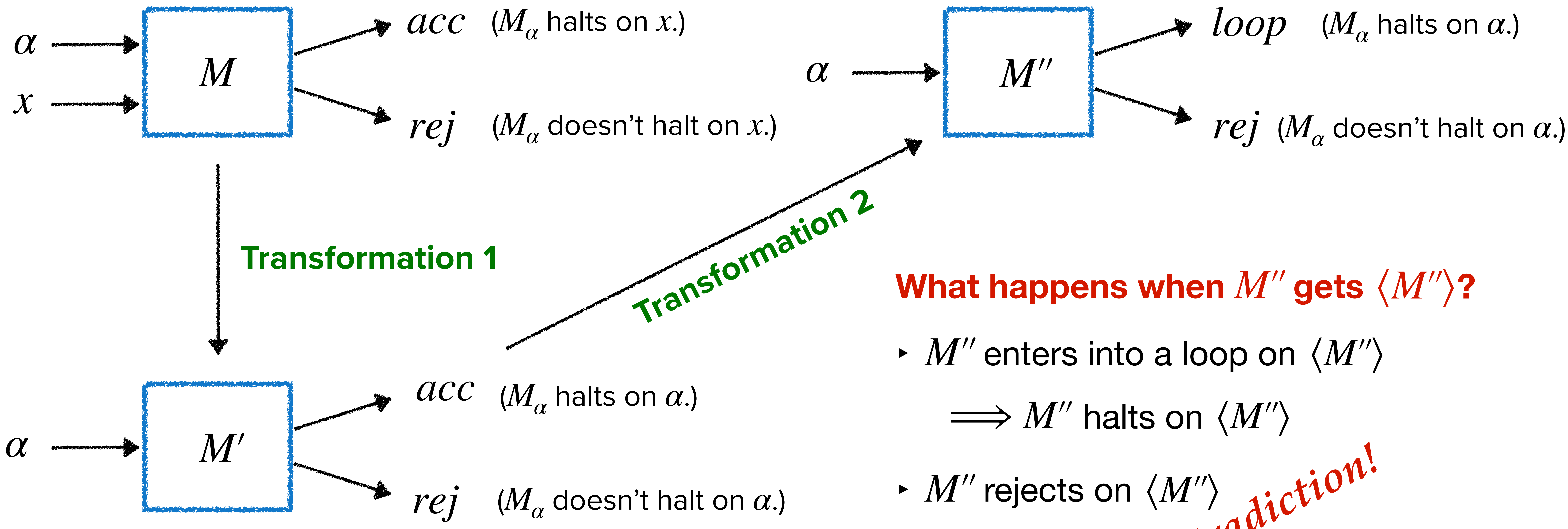
Definition: A TM that halts, i.e., either accepts or rejects, on every input is called a **Halting Turing Machine.**

Halting Problem: $\text{HALT} = \{(\alpha, x) \mid M_\alpha \text{ halts on input } x\}$

Theorem: HALT is undecidable.

Undecidability of HALT

Suppose a TM M decides HALT $= \{(\alpha, x) \mid M_\alpha$ halts on input $x\}$.



What happens when M'' gets $\langle M'' \rangle$?

- M'' enters into a loop on $\langle M'' \rangle$
  - $\implies M''$ halts on $\langle M'' \rangle$
- M'' rejects on $\langle M'' \rangle$
  - $\implies M''$ does not halt on $\langle M'' \rangle$

Contradiction!

Recognising HALT

Theorem: HALT $= \{(\alpha, x) \mid M_\alpha$ halts on input $x\}$ is recognisable.

Proof: Consider a TM U that on input (α, x):

- Starts simulating M_α on x.
- If M_α ever halts on x, U halts and accepts.
- If M_α never halts on x, U does not halt as well.

U uses portion of its tape as the tape of M_α and binary encoding to store symbols of M_α.

Clearly, $L(U) =$ HALT. ∎

The Big Picture