CSE 396 Lecture Tue 4/16 Spring 2019

TopHat # 4497

Def^n: A language A is (Turing-)recognizable if there is a deterministic TM M s.t. $A = L(M)$.

Synonyms:
- Turing-acceptable (safest IMHO)
- recursively enumerable (r.e.) Historical term.
- computably enumerable (c.e.) modern alternative
- "partially decidable" (on the "yes" side)

Def^n: A DTM M is total, synonym halts for all inputs, synonym is a decider, if for all $x \in \Sigma^*$, $M(x)\downarrow$ "halts".

Def^n: The language A is decidable if it equals $L(M)$ for some total TM M.
Synonym: recursive. Historical: concept originally defined for formal systems of recursion..

Def^n: The class of decidable languages is denoted by DEC or REC.
The class of Turing-recognizable languages is only called RE.

Def^n: A function $f: \Sigma^* \to \Sigma^*$ is computable if there is a Turing machine Transducer (i.e., a DTM M that produces output) s.t. for all x, $\underbrace{M(x)\downarrow =}_{\text{"halts and outputs"}} f(x)$
Synonym: f is recursive or total recursive

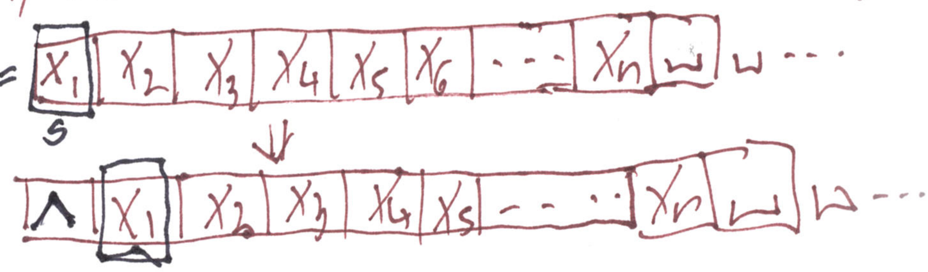
If we allow $Dom(f)$ to not be all of Σ^*, and allow $M(x)$ might not halt ($\uparrow$) for $x \notin Dom(f)$, then f is partial computable syn.: partial recursive.

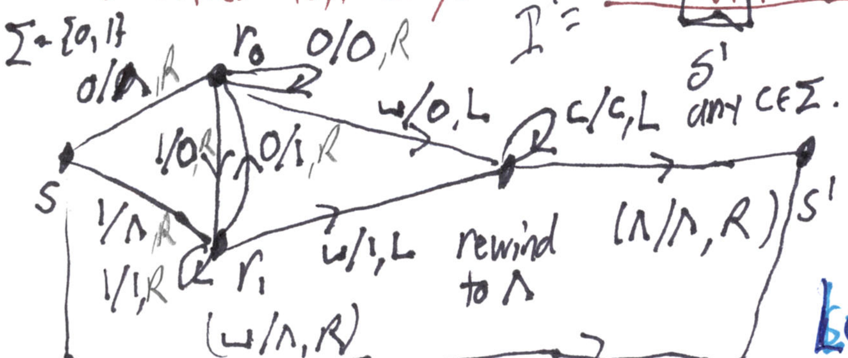
We will now see that these def^{n}s apply not just to TMs, but to every other high-level model or programming language (HLL).

Some basic capabilities of TMs that we've seen (in demos):

- Simple arithmetic, like $3n+1$. With a 3-tape TM we can directly execute standard addition in binary. Can do multiplication either directly ("CISC") or by repeated addition ("RISC-like") Mult by 2 and div. by 2 are done by <u>shifting</u>.
- Compare two strings char by char, forward or backwards. A second tape helps speed this up. Can also keep counts, e.g. of parens. This capability can be used to search for a label on a tape.
- Copy strings from tape to tape.
- If not enough room to copy w between two '[', ']' delimiters, say, then we can automatically enter a "Shift-Over" routine on reading the ']'.

Can use same routine for TMs whose tapes have a fixed left edge.





These capabilities suffice to emulate a "Mini Assembly" language, actually fairly rich — even with

<u>Load Indirect</u> i: Look up the value in register i, treat v as another address and load the value v' to the ALU from there.

<u>Store Indirect</u> i: Look up v at reg i then copy ALU into v.

Show Univ RAM Simulator Handout. What it shows is:

<u>Theorem</u>: If a function f is (partially) computable in any HLL, then it is partially computable on a TM. Likewise for (partially) deciding languages.

This is the largest evidence for the <u>Church-Turing Thesis</u>: Every machine in Nature, any human decision criterion, will be no stronger than the TM-based criteria

Theorem: There is a Universal TM, i.e. a TM U that takes inputs of the form $w = M\#x$ and such that $U(w)$ simulates $M(x)$.

Proof: We can give inputs w already to the Turing Kit. Compile Turing Kit to mini-assembly program P. Use U = the UTM Simulator with P, ~~M, x~~ hard-coded. $w = M\#x$ as its input. ⊠

Note: $L(U) = \{\langle M, x\rangle : M \text{ accepts } x\}$ = the language of the TM Acceptance Problem A_{TM} in Ch 4.

Hence A_{TM} is a c.e. language.
Is it decidable? We'll see... not!

Theorem: If A is decidable then so is its complement $\tilde{A}$.

Fact: Not all DPDAs are total, but every DPDA D can be converted to an equivalent D' that is total. (§2.4)

Thm: Every k-tape TM M can be converted into an equivalent 1-tape TM M', st M' is total iff M is.

(proof skipped).

FOOTNOTE:
We should include as a basic capability on page 2 the idea of testing a cell for 1 or 0 (or
blank). This was exemplified in the "3n+1" machine example by the final test for $n=1$. We can
test a whole string in a register for being all-0s. This implements JMPℓ = jump code to label ℓ if ALU = 0.