CSCE 355
4/23/2025

ALL_{CFG} is undecidable (cont.)

Given a TM M and input string w, we build a PDA P that accepts all strings iff M does not accept w.

A *rigid accepting trace* of M on w is a string "$ID_0 \vdash ID_1 \vdash \cdots \vdash ID_n$" such that,

- ID_0 is the initial ID $\underline{q_0 w}$ on input w,
- ID_n is an accepting ID,
- $ID_i \vdash ID_{i+1}$ (ID_{i+1} is the one-step successor of ID_i) for all $0 \le i < n$,
- IDs are padded with $\sqcup$'s (blanks) so that they all cover the same portion of tape. [in particular, all IDs have the same length.]

M accepts w iff a rigid accepting trace exists for M on w.

The PDA ~~P~~ will accept all strings <u>except</u> rigid accepting traces of M on input w.

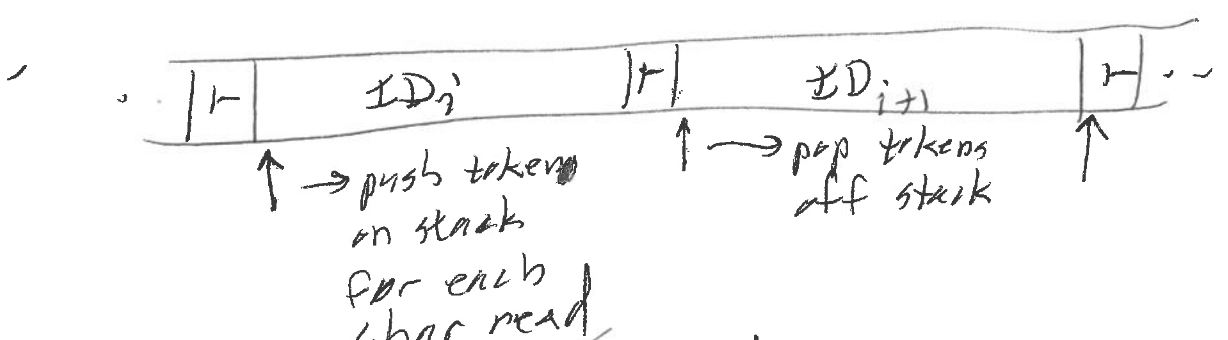
P nondeterministically looks for an "error" in its input, showing that it is not a rigid accepting trace, accepting if it finds one.

Description of P: [terminal alphabet is $Q \cup \Gamma \cup \{\vdash\}$] (Q: state set of M; Γ: tape alphabet of M)

P nondet. branches into a number of "sub" PDAs $P_1, P_2, \ldots$, each looking for one kind of error:

P_1: looks for a syntax error; say a malformed ID, i.e., the ID does not contain exactly one state symbol. Accepts if found. (A DFA can do this, in fact).

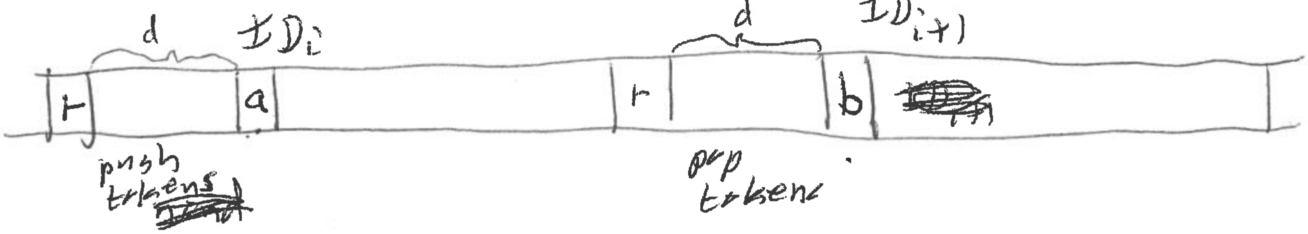
P_2: checks if two IDs have different lengths; nondet. chooses a pair of adjacent IDs:



accepts if all tokens popped before finishing ID_{i+1} or some token left after finishing.

P_3: boundary conditions: accepts iff the first ID is not initial ID $q_0 w$ (hard-coded into P) or the last ID is not accepting ID.

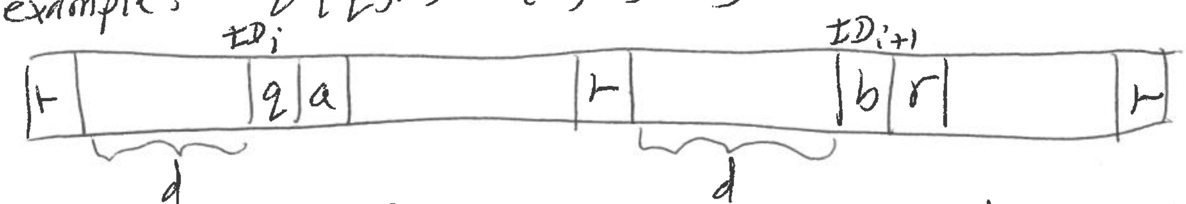
P_4: Checks that unscanned symbols are the same from one ID to the next; accepts if it finds a discrepancy.



nondet. choose i
nondet choose an unscanned symbol in ID_i
record it in the state of P_4 then check that the corresponding symbol in ID_{i+1} is the same, accepting if not.

P_5: Check that the neighborhood of the scanned symbol in ID_i is altered in ID_{i+1} according to M's transition function, accepting if not.

For example: $\delta(q, a) = (r, b, \rightarrow)$



If the input string is not a rigid accepting trace, then one or more of these branches will accept, and conversely.

$\therefore$ Any decision procedure for ALL_{CFG} can be used to decide whether M accepts w, for any TM M and string w:

1. Construct P as above (algorithmically)
2. Convert P into an equivalent CFG G (algorithmically)
3. ~~Use~~ Run any decision procedure for ALL_{CFG} on input G. If accepts, then reject; else accept.

Contradiction, because A_{TM} is undecidable. //

<u>Def</u>: The <u>intersection problem</u> for CFGs, INT_{CFG}, is as follows:

Given a pair of CFGs G_1, G_2 with the same token alphabet, is $L(G_1) \cap L(G_2)$ nonempty?

<u>Theorem</u>: INT_{CFG} is undecidable.

Proof outline: Given a TM M and input string w, we construct (algorithmically) two grammars G_1, G_2 such that $L(G_1) \cap L(G_2) \neq \emptyset$ iff M accepts w

$\therefore$ Any decision procedure for INT_{CFG} can be used to decide A_{TM} ~~$\bar{\Sigma}$~~, so no such decision procedure exists.

An <u>alternating</u> <u>accepting</u> <u>trace</u> of M on w looks like

$$\overbrace{\underbrace{ID_0}\vdash ID_1^R}^{L(G_1)} \vdash \overbrace{ID_2 \vdash ID_3^R}^{L(G_1)} \vdash \cdots \vdash \underbrace{ID_n} \vdash \quad (n \text{ ~~odd~~ even})$$

$L(G_2)$ $L(G_2)$

reverse each odd-numbered ID.

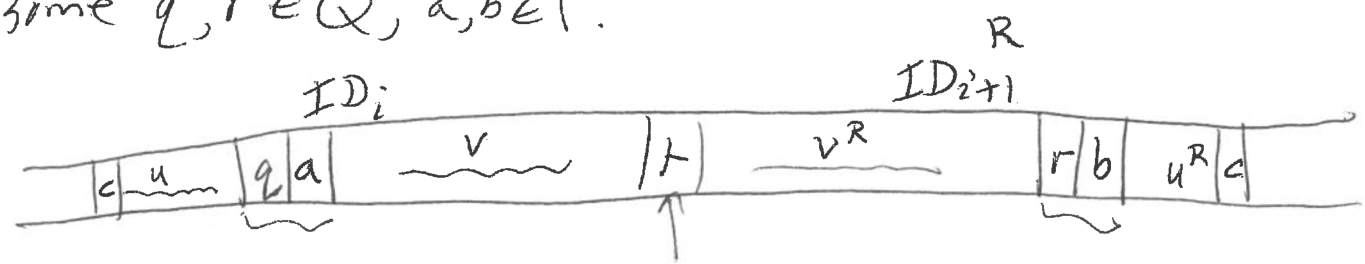
G_1 generates a sequence of pairs of IDs $ID_i \vdash ID_{i+1}^R$ separated by $\vdash$'s

G_2 will generate a sequence of pairs of IDs $ID_j^R \vdash ID_{j+1}$ separated by $\vdash$'s,

ID_{i+1} is the successor of ID_i, and same for ID_{j+1}

prefixed by the initial ID and with an accepting ID appended.

<u>Example</u> of generating $ID_i \vdash ID_{i+1}^R$ where the transition from ID_i to ID_{i+1} uses $\delta(q,a) = (r,b,\rightarrow)$ some $q, r \in Q$, $a, b \in \Gamma$.



$S \to aSa \mid bSb \mid cSc \mid \ldots$

$\Gamma = \{a, b, c, \ldots\}$

each symbol in Γ

$S \to qaTrb$

$T \to aTa \mid bTb \mid cTc \mid \ldots$

$T \to$ "$\vdash$"

$G_1 : S' \to \varepsilon \mid S \vdash S'$

G_2 : similar