14. Context-Sensitive Languages

Goal:
- Introduce context-sensitive languages (CSLs) and Turing machines.
- Prove decidability of membership in CSLs.
- Show correspondence of CSLs and languages accepted by linear-bounded Turing machines.
- This also establishes equivalence of the recursively enumerable languages and the languages accepted by Turing machines.

Definition:
- A type-0 grammar is a grammar $G = (N, \Sigma, P, S)$ with $P \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^*$.
- The grammar is context-sensitive (or type-1), if for all productions $\alpha_1 \to \alpha_2 \in P$ we have $|\alpha_1| \leq |\alpha_2|$.

  Moreover, we may have $S \to \varepsilon$, but then there is no rule that has S in its right-hand side.
- A language $L \subseteq \Sigma^*$ is context-sensitive, if there is a context-sensitive grammar G with $L = L(G)$.
- A language $L \subseteq \Sigma^*$ is recursively enumerable, if $L = L(G)$ for G of type-0.

Example:

$L = \{ w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w) \}$.

We have $L = L(G)$ with

$G = S \to \varepsilon \mid R$

$R \to TABC \mid ABCR$

$T \to a \quad B \to b \quad C \to c$

$TB \to BT \quad CT \to TC$

$TC \to CT \quad CB \to BC$

$BT \to TB$

$RT \to CB$

Let $L(G)$ be a context-sensitive language (over Σ).
The membership problem is

MEMBERSHIP $L(G)$:

Given: $w \in \Sigma^*$.

Question: $w \in L(G)$?

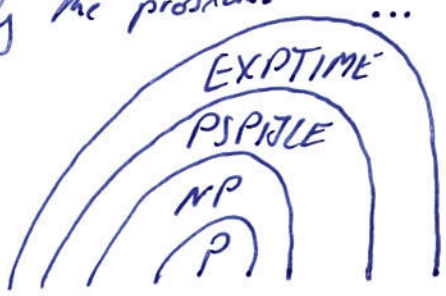
Theorem: For every context-sensitive grammar G,
MEMBERSHIP $L(G)$ can be solved in $2^{O(|w|)}$.

Proof:
- Since the productions are length preserving,
once we derived a sentential form of length $> |w|$,
there is no chance to derive w from it.
- We thus enumerate all sentential forms of length $\leq |w|$
and see whether we find w.
- There are
$$(|N| + |\Sigma| + 1)^{|w|} = 2^{O(|w|)} \text{ many,}$$
which yields the time bound.
Note that N and Σ are not part of the input
and therefore $|N| + |\Sigma| + 1 = 2^c$
for c a constant that is independent of the input (word w). □

The problem is hard:
- The correspondence with linear-bounded Turing machines
that we establish below shows that

CSL = PSPACE,

the context-sensitive languages are precisely the problems solvable with polynomial space.



- Since there are PSPACE-hard problems,
  there are context-sensitive languages $L(G)$
  for which membership is PSPACE-hard.

→ PSPACE is an important complexity class
  for verification problems:
  ↳ Reachability in Boolean shift-programs
  ↳ Reachability in multi-threaded programs.

→ Context-sensitive aspects show up in compilers
  ↳ Parameters in procedures
  ↳ Scopes of objects.

14.1 Turing Machines

Goal: Define an automaton model, the Turing machine (TM),
for context-sensitive languages and
for recursively enumerable languages.

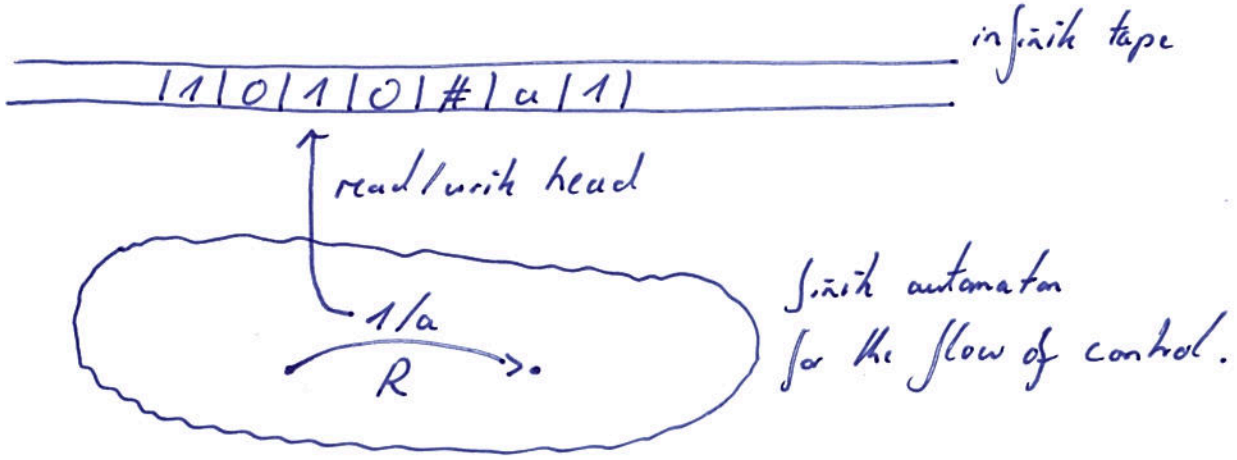
Idea:
- Drop the last-in-first-out restriction for pushdowns
  and access arbitrary information
  stored about the computation so far.
- For context sensitivity, we just bound the amount of information
  we can store.

Church's thesis ('Alonzo Church 1936):
Turing's idea was more than capturing a class of languages.
He wanted to formulate the idea of computability itself,
wanted to capture what is an algorithm.
Up to today, we believe he succeeded.
EVERY problem for which we found an algorithm (C, C++, Java)
we were able to solve with a Turing machine
(even with low overhead).

Church's thesis states that it will stay this way:

Everything that is computable is computable by a TM.

Technically:



Definition (Syntax of a TM):

A Turing machine (TM) is a tuple $M = (Q, \Sigma, \Gamma, q_0, \sqcup, \delta, Q_F)$ with

- Q a finite set of states, $q_0 \in Q$ the initial state, $Q_F \subseteq Q$ the set of final states,
- Σ the input alphabet,
- Γ the tape alphabet with $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma \setminus \Sigma$ the blank symbol,
- $\delta : Q \times \Gamma \longrightarrow \Gamma \times \{L(eft), R(ight), N(eutral)\} \times Q$, in which case the TM is deterministic (DTM), or $\delta \subseteq Q \times \Gamma \times \Gamma \times \{L, R, N\} \times Q$ for a non-deterministic TM (NTM)

To define the semantics of a TM, we need the notion of a configuration, the state of the TM at runtime.

Definition (Semantics of a TM):

Let $M = (Q, \Sigma, \Gamma, q_0, \sqcup, \delta, Q_F)$ be a TM.

- A configuration of M is a word from $\Gamma^* Q \Gamma^+$.

If $uqav$, then uav are the cells of the tape already visited,
q is the current state and
$a \in \Gamma$ is the symbol currently under the read-write head.
Given input $x \in \Sigma^+$, the corresponding <u>initial configuration</u> is $q_0 x$.
Given input ε, the <u>initial configuration</u> is $q_0 \sqcup$.

- A configuration is <u>accepting</u> if it is from $\Gamma^* Q_F \Gamma^+$.
- The <u>transition relation</u> among <u>configurations</u>

$$\longrightarrow \subseteq (\Gamma^* Q \Gamma^+) \times (\Gamma^* Q \Gamma^+)$$

is defined by

$uaqbv \longrightarrow uq'acv$, if $q \xrightarrow[L]{b/c} q' \in \delta$

$uaqbv \longrightarrow uaq'cv$, if $q \xrightarrow[N]{b/c} q' \in \delta$

$uaqbv \longrightarrow uacq'v$, if $v \neq \varepsilon$ and $q \xrightarrow[R]{b/c} q' \in \delta$

$uqb \longrightarrow ucq'\sqcup$, if $q \xrightarrow[R]{b/c} q' \in \delta$

$qbu \longrightarrow q'\sqcup cu$, if $q \xrightarrow[L]{b/c} q' \in \delta$.

We use $\longrightarrow^*$ for the reflexive and transitive closure of $\longrightarrow$.

- The <u>language of M</u> is

$$L(M) := \{ w \in \Sigma^* \mid q_0 w \longrightarrow^* u q_f v \in \Gamma^* Q_F \Gamma^+ \}.$$

($q_0 \sqcup$ for ε)

For the characterization of the context-sensitive languages,
we define Turing machines that only use the part of the tape
given by the input (plus a left and a right end marker).

<u>Definition (Linear-bounded automaton)</u>:

- A <u>linear-bounded automaton</u> (LBA) is an NTM

$$M = (Q, \Gamma, \Sigma \cup \{\$_l, \$_r\}, q_0, \delta, Q_F).$$

Instead of a blank $\sqcup$, we have a left end marker $\$_l$
and a right end marker $\$_r$.

Moreover, there are two restrictions
1) There are no moves left from $\$_l$ and right from $\$_r$.
2) The symbols $\$_l$ and $\$_r$ are never overwritten.

The Language of M is
$$L(M) := \{ w \in \Sigma^* \mid q_0 \$_l w \$_r \to^* u q_f v \in \Gamma^* Q_F \Gamma^* \}.$$

A tape compression result in complexity theory shows
that an amount of tape linear in the size of the input
leads to the same computational power (as restricting to the input itself).
Hence the name linear-bounded automaton.

14.2 Equivalence with Context-Sensitive Languages

Goal: Show that the languages accepted by LBAs
are precisely the context-sensitive languages.

Theorem (Kuroda '64):
A language $L \subseteq \Sigma^*$ is
accepted by an LBA iff it is context-sensitive.

Proof:
"$\Leftarrow$" Let $L = L(G)$ with $G = (N, \Sigma, P, S)$.
We give an NTM M that accepts L.
The idea is to apply the productions backwards
until the start symbol is found.
↳ Consider an input $x \in \Sigma^*$.
↳ M non-deterministically selects a production
$$\alpha \to \beta \in P$$
and finds an arbitrary occurrence of β in x.

↳ Now β is replaced by α,
potentially moving letters to the left if $|\alpha| < |\beta|$.

↳ If we reach S, we stop and accept.
Otherwise, we repeat the above non-deterministic procedure.

Since the productions are length preserving ($|\beta| \geq |\alpha|$),
we never exceed the part of the tape used by the input.

$\Rightarrow$ Let $L = L(M)$ for an LBA $M = (Q, T, \Sigma \dot\cup \{\$_\ell, \$_r\}, q_0, \delta, Q_E)$.

Idea:
- We give a grammar G that operates on sentential forms representing the configurations of M.
- We will have to generate the initially given word from these sentential forms.
  Therefore, they cannot be longer than the input (we cannot delete).
- Note that this bound works for the intermediary configurations due to the linear boundedness assumption.

Technically:
- We will store pairs
  $(act_1, a_1) \dots (act_n, a_n)$
  where
  $act_1 \dots act_n$ forms the current tape content and
  $a_1 \dots a_n$ is the initial input.
- The letters act_i are taken from the alphabet Δ with
  $\Delta' := T \cup (\{\$_\ell, \$_r\} \times T)$ and $\Delta := \Delta' \cup (Q \times \Delta') \cup (\{\$_\ell, \$_r\} \times Q \times T)$.
  So an intermediary configuration
  $\$_\ell \times q y a z \$_r$

derived from input

$\$_\ell\ a\ b\ a\ b\ \$_r$

has the shape

$((\$_\ell, x), a)((q, y), b)(a, a)((\$_r, \varepsilon), b)$.

- It is immediate to simulate the moves of the LBA by considering only the current configuration (one has to track a bit at the border (*)).

So if $q \xrightarrow[R]{a/b} q' \in \delta$, then we obtain the productions

$((q, a), x).(c, y) \longrightarrow (b, x).((q', c), y)$ for all $x, y \in \Sigma$

in the grammar.

(*) For the border, we use
$(q, \$_\ell, a)$ to indicate that the head is on the left end marker
and $(\$_\ell, q, a)$ to indicate that the head is on the first letter.

Construction:

We define
$G := (N, \Sigma, P, S)$

with $N := \{S, A\} \cup (\Delta \times \Sigma)$

$P := \{ S \to A.((\$_r, a), a) \mid a \in \Sigma \}$
$\cup \{ A \to A.(a, a) \mid a \in \Sigma \}$
$\cup \{ A \to ((q_0, \$_\ell, a), a) \mid a \in \Sigma \}$
$\cup$ Productions that simulate the behavior of M
$\cup \{ ((q_f, a), b) \to b \mid q_f \in Q_f,\ a \in \Delta',\ b \in \Sigma \}$
$\cup \{ (a, b) \to b \mid a \in \Delta',\ b \in \Sigma \}$.

□

Note: The construction works the same way
if we drop
↳ the length-preserving condition for grammars and
↳ the linear boundedness assumption for TMs.

Corollary:
The languages accepted by an NTM are precisely
the recursively enumerable languages.

14.3 Determinism

Recall: · For finite automata, we know that
NFAs and DFAs accept the same languages.
· For pushdown automata, we know that
DPDAs accept a strict subclass of the context-free languages.

Open: Kuroda, in his 1964 paper, posed two questions:

(1) The first question is indeed on
the role of determinism for LBAs:
Are the languages accepted by NLBAs
precisely the languages accepted by DLBAs?

It is still open!

(2) The second question was on complementation:
Are the languages accepted by NLBAs
(the context-sensitive languages)
closed under complement?

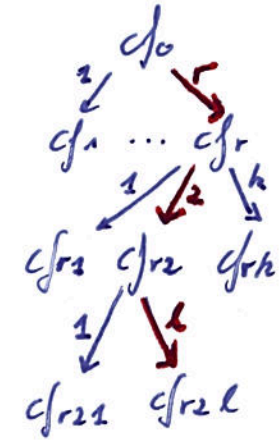
Kuroda showed that ¬(2) ⇒ ¬(1).
This did not help much:
Immerman & Szelepcsényi proved (2) to hold!
We will see the proof in the next lecture.

Goal: Show that for TM, NTMs and DTMs accept the same languages.

Theorem:
L is accepted by an NTM iff L is accepted by a DTM.

Proof:
„<=" ✓ Every DTM is an NTM.
„=>" The idea is to traverse the computation tree of the NTM in breadth-first manner.

- Let $L = L(M_1)$ with M_1 an NTM.
For any control state and any tape symbol, there is a finite number of choices for the next move of M_1.
Let $r \in \mathbb{N}$ be the maximal number of choices for any pair of control state and tape symbol.
Then any finite sequence of choices can be represented by a sequence of digits from 1 to r.
Not all such sequences may represent choices of moves, there may be fewer than r choices in some situations.



The path is given by the sequence r.2.l

- We simulate M_1 by a DTM M_2 that has three tapes.
The three tapes can be compressed into one tape by taking letters to be triples $\binom{a}{b}{c}$ or by writing the tapes one after the other.
// We will discuss such techniques when we come to computability.
The first tape of M_2 holds the input.
On the second tape, M_2 generates sequences of the digits 1 to r. in a systematic manner that implements a breadth-first search:
↳ The sequences are generated in increasing length, starting from the shortest.
↳ Sequences of equal length are generated in numerical order.

- For each sequence generated on tape 2,
  M_2 copies the input onto tape 3
  and then simulates M_1 on tape 3 —
  using the sequence on tape 2 to <u>resolve non-deterministic choices</u>.
  If there is a sequence of choices leading to acceptance of M_1,
  it will eventually be generated on tape 2.
  Then M_2 then simulates M_1, it will accept.
  If no sequence of choices of M_1 leads to acceptance,
  M_2 will not accept. □

<u>Note</u>: The construction does not carry over to LBAs
as it is not clear that it preserves linear boundedness
(the sequence of choices may be long).