Languages, Machines and Computation - A Summary

Vikram Rao S

August 17, 2011

Contents

1 Grammars

- Grammar G=(N,T,P,S) contains sets of non-terminals, terminals, productions and a start symbol.
- Type 0 - Phase structured, Type 1 - context sensitive (αA$\beta- >\alpha\gamma\beta$), Type 2 - context free (A$- >\alpha$), Type 3 - regular (A$- >\alpha$B$|\alpha$).
- Defining grammars for a^n, $a^n cb^n$ etc.
- Derivation trees - leftmost and rightmost derivation trees.
- A CFG is ambiguous if a word has >= 2 leftmost derivations.
- A CFL is inherently ambiguous if every grammar generating it is ambiguous.
- Simplifying CFGs - removing useless productions and unit-rules.
- Normal forms of CFGs - Chomsky, weak and strong Chomsky and Greibach normal forms.

2 Finite State Automata

- An FSA M=(K, Σ, δ, q_0, F) contains a set of states, an input alphabet, a mapping function ($\delta : Kx\Sigma- > K$ for deterministic FSAs and $\delta : Kx\Sigma- > K^*$ for non-deterministic FSAs), a start state and a set of final states.
- Regular expressions and conversion between regexes and FSAs.
- Pumping lemma for regular sets.
- Regular languages are closed under union, intersection, complementation, concatenation, star and reversal.
- Myhill-Nerode Theorem and the minimum-state FSA.
- FSAs with output.

3 Pushdown Automata

- A PDA M=(K, Σ, Γ, δ, q_0, Z_0, F) contains a set of states, input alphabet, pushdown alphabet, mapping function, initial state and stack symbol and a set of final states.
- A language L is accepted by a PDA M_1 by final state <=> it is accepted by a PDA M_2 by empty store.

- A word w is accepted by a PDA M by final state if it reaches a final state on reading w, irrespective of the stack and it is accepted by empty store if the stack is emptied on reading w irrespective of the state.
- A language L is generated by a CFG G <=> it is accepted by a PDA M.
- CFLs are closed under union, catenation, * and homomorphism.
- It is decidable whether a CFL is empty, finite or infinite.
- The membership problem in CFLs is decidable.

4 Turing Machines

- Turing machine M = (K, Σ, Γ, δ, q_0, F), where all symbols are as usual. Γ is a set of tape symbols. $\Sigma \subseteq \Gamma$ and b is the blank symbol. δ is the mapping from from KxΓ to KxΓX{L, r}.
- A TM's instantaneous description (ID) is of the form αqβ, meaning that the TM is in state q on the first symbol of β.
- A TM can be thought of as both an acceptance device and a computational device.
- Techniques for TM construction - considering state as a tuple, considering state as a tuple, having subroutines etc.
- Turing Machine variations - two-way infinite tape TM, multi-tape TM, multi-head TM, non-deterministic TM and two-dimensional TM.
- Restricted versions of TMs - 4-counter TM, 3-counter TM, 2-counter TM, a TM with tape alphabet {0, 1, b}.
- A turing machine can enumerate all the strings of a Type-0 language. Hence, a TM can be considered as an enumerator.
- L is generated by a Type-0 grammar <=> L is accepted by a TM M.
- Godel number of a sequence i_1,..., i_n is $2^{i_1} * 3^{i_2} * \ldots * (n^{th} prime)^{i_n}$.

5 Turing Machines - 2

- A universal turing machine U takes an encoding of a TM M and an input w as an input and simulates it.
- L_u is the language accepted by the universal TM ie., it contains such strings Mw such that M represents a valid TM and M accepts w.

- L_d, the language containing strings w_i which are not accepted by T_i, where strings and turing machines are ordered lexicographically, is not recursively enumerable.

$$L_d = \{w_i | w_i \textit{ is not accepted by } T_i\}$$

- L_d is not recursively enumerable, because if if it was, then a certain T_i would accept it. Therefore, w is in L_d means, w is accepted by T_i, contradicting its definition.

- The complement of L_d is recursively enumerable but not recursive (Recursive lenguages can be accepted by a TM that halts on all inputs).

- L_u is recursively enumerable but not recursive.

6 Problems - undecidablity etc.

- The halting problem of turing machines is undecidable recursively.

- A set $\mathcal{F}$ of languages is called a property.

- Rice's theorem states that any non-trivial (meaning, non empty and not containing all RE languages) property of recursively enumerable languages is undecidable.

- Hence, the following properties of RE sets are undecidable - emptiness, finiteness, regularity, context-freedom, nonemptiness, recursiveness etc.

- Post's Correspondence Problem is undecidable.

- To show that a problem is undecidable, reduce it to a known undecidable problem.

7 Complexity

- A Turing machine that, given an input of length n, always halts within T(n) moves is said to be T(n)-time bounded.

- If a DTM M is T(n)-time bounded for some polynomial T(n), then we say M is polynomial-time bounded. And L(M) is said to be in the class $\mathcal{P}$.

- A multitape TM can simulate a computer that runs for time O(T(n)) in at most O(T^2(n)) of its own steps.

- Input size has a specific meaning: the length of the representation of the problem instance as it is input to a TM.

- The running time of a nondeterministic TM is the maximum number of steps taken along any branch.

- If that time bound is polynomial, the non-deterministic TM is said to be polynomial-time bounded. And its language/problem is said to be in the class $\mathcal{NP}$.

- Originally a curiosity of Computer Science, mathematicians now recognize as one of the most important open problems the question P = NP?

- There are thousands of problems that are in NP but appear not to be in P.

- But no proof that they arent really in P.

- We say that a language L is polynomial time reducible to a language M is there exists a deterministic polynomial time bounded TM that for each input x produces an output y that is in M if and only if x is in L.

- Let C be a class of languages. We say that a language L is complete for C wrt polynomial time reductions if L is in C and every language in C is polynomial time reducible in L.

  Also, L is said to be hard for C wrt polynomial time reductions if every language in C is polynomial time reducible to L but L is not necessarily in C.

- The Boolean Satisfiability Problem is $\mathcal{NP}$ complete.

- To show that a problem is $\mathcal{NP}$ complete, show that it is polynomial time reducible to a known $\mathcal{NP}$ complete problem.