Hierarchies of Languages and Accepters

5DV037 — Fundamentals of Computer Science
Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

Language Hierarchies

- The classes of languages which have been studied in this course fit into a natural hierarchy.

Example: The regular languages are a subset of the context-free languages.

- In this presentation, such hierarchies will be formulated more carefully.
- Two main hierarchies will be considered:
- The *Chomsky hierarchy*:
  - formulated by the eminent linguist Noam Chomsky during the 1950's;
  - somewhat incomplete but still important to know because of the widespread use of the associated terminology.
- A full hierarchy summarizing all of the classes which have been studied in the course, together with an additional class present in the Chomsky hierarchy.

The Classical Chomsky Hierarchy

- This hierarchy was forwarded by Noam Chomsky during the 1950s.
- It is summarized in the following table.

Chomsky Name	Modern Name	Accepter	Grammar
type-3	regular	DFA/NFA	regular
type-2	context free	NPDA	context free
type-1	context sensitive	LBA	context sensitive
type-0	Turing enumerable	DTM/NDTM	phrase structure

- Each line in the table identifies a class which is a proper subset of the line below it.
- The third line introduces unfamiliar notions.
- LBA stands for *linear-bounded automaton*.
- LBAs and context-sensitive grammars and languages will be introduced on the following slide.

Context-Sensitive Grammars and Languages and LBAs

- The grammar $G = (V, \Sigma, S, P)$ is *context sensitive* (a *CSG*) if every production $\alpha \to \beta$ has the the property that Length$(\alpha) \leq$ Length(β).
- Thus, a CSG cannot have any null rules $(A \to \lambda)$.
- Note that a context-free grammar is context sensitive provided it has no null rules.
- A language L is *context sensitive* (a *CSL*) if $L \setminus \{\lambda\}$ is generated by a CSG.
- Thus, the empty string is treated as a special case.
- A *linear-bounded automaton* (or *LBA*) is a DTM M which has the restriction that for any input of the form $\mathcal{I}\langle M, \alpha\rangle$, the tape head is only allowed to scan and rewrite the tape squares containing α.
- Thus, the memory allowed an LBA is bounded by the length of the input string.
- This model is largely of historical interest and will not be studied further in this course.

The Hierarchy Studied in the Course

- Each row in the table is a proper subset of the row below it.

Name	Accepter	Grammar
regular language	DFA/NFA	regular grammar
deterministic CFL	DPDA	LR(k)
unambiguous CFL	-	unambiguous CFG
CFL	NPDA	CFG
CSL	LBA	CSG
(Turing) decidable / recursive	DTM/NDTM decider	-
Turing acceptable / recursively enumerable / semidecidable	DTM/NDTM	PSG

Some Closure Results

- L, L_1, L_2 = language in the class; R = regular language.

Name	$L_1 \cup L_2$	$L_1 \cap L_2$	$\overline{L}$	$L \cap R$
regular language	Y	Y	Y	Y
deterministic CFL	N	N	Y	Y
unambiguous CFL	N	N	N	Y
CFL	Y	N	N	Y
CSL	Y	Y	O	U
(Turing) decidable / recursive	Y	Y	Y	Y
Turing acceptable / recursively enumerable / semidecidable	Y	Y	N	Y

- O = Open problem, to the best of my knowledge.

Some Decidability Results

- D = decidable; N= undecidable.

Name	$L = \emptyset$	$L = \Sigma^*$	$L_1 = L_2$	$L_1 \cap L_2 = \emptyset$
regular language	D	D	D	D
deterministic CFL	D	D	D	U
unambiguous CFL	D	?	?	U
CFL	D	U	U	U
CSL	D	U	U	U
(Turing) decidable / recursive	D	U	U	U
Turing acceptable / recursively enumerable / semidecidable	U	U	U	U

- ? = I am not sure, but I think that the answer is U.

The Utility of a Formal Language Hierarchy

Question: What is the practical use of such a hierarchy and all of the theoretical results?

- All real computers have finite memory and thus are modelled using finite automata.

Answer: Although undecidable problems become “decidable” with finite memory,

  - the amount of time required to determine that a program which has not halted is really looping would in general be enormous;
  - enormous as in “the sun will burn out first and all life as we know it will cease”.

- The theory thus provides a way to distinguish between problems which are solvable only by exhaustive simulation and problems which can be solved by a “smart” program.