Formal Languages

Regular Languages

A regular language is either a regular expression, a DFA or a NFA.

Pumping Lemma:

If *A* is a regular language, then there is a number *p* (the pumping length) where, if *s* is any string in *A* of at least *p*, then *s* may be divided into three pieces, $s = xyz$ satisfying the following conditions:

1. For each
2. $y| > 0$
3. $xy| \le p$

The regular languages are closed under union, complement, concatenation, and star.
Non-regular languages:

- $L_1 = \{0^n 1^n \mid n \ge 0\}$
- Proof:is not regular and ()

Context-Free Languages

A CFL is either expressed as a push down automata or using a context free grammar, both are equivalent.

Chomsky Normal Form:

A context free grammar is in chomsky normal form if every rule is in the form:

- $A \rightarrow BC$
- $A \rightarrow a$

Where *a* is any terminal and *A,B,* and *C* are any variables – except that *B* and *C* may not be the start variable. In addition we permit the rule $S \rightarrow \varepsilon$, where *S* is the start variable.

Pumping Lemma:

If *A* is a context-free language, then there is a number *p* (the pumping length) where, if *s* is any string in *A* of at least *p*, then *s* may be divided into three pieces, $s = uvxyz$ satisfying the following conditions:

4. For each

5. $|vy| > 0$
6. $|vxy| \leq p$

The context free languages are closed under union, complement, concatenation, and star.
If *C* is a context free language and *R* is a regular language then
$C \cap R$
is context free

Non-context free languages:

- $L_1 = \{a^n b^n c^n \mid n \geq 0\}$
- $L_2 = \{a^i b^j c^k \mid k = \max(i, j)\}$

A language *L* over a singleton alphabet {0} is context-free if and only if its is regular

Turing Machines

Recursively Enumerable:
A language is r.e. if there is some *TM* that recognizes it. (Notice that this *TM* can still loop when trying to decide whether or not a string *w* is in the language!)

Decidable:
A language is decidable if some Turing machine decides. (it will either accept or reject!)

Theorem:
A language is decidable if and only if it is both r.e and co-r.e.

Some decidable languages:

- | *B* is a DFA that accepts input string *w*}
- | *B* is a DFA that accepts input string *w*}
- | *R* is a regular expression that generates *w*}
- | *A* is a DFA and }
- | *A* and *B* are DFA's and }
- | *G* is CFG that generates *w* }
- | *A* is a CFG and }
- | *M* is a LBA and *M* accepts *w*}
- Every context free language is decidable

Here we have some undecidable languages:

- | *M* is a TM and *M* accepts *w*}
- | *M* is a TM and *M* halts on input *w*}
- | *M* is a TM and }
- | *M* is a LBA and }

- | *M* is a TM and is a regular language}
- | *M, M'* are TMs and }
- | *G* is a CFG and }

Reducibility

Mapping reducible:
Language *A* is mapping reducible to language *B* written
$A \leq_m B$
, if there is a computable function
$f : \Sigma^* \rightarrow \Sigma^*$
, where for every *w,*
$w \in A \Leftrightarrow f(w) \in B$
the function *f* is called the reduction of *A* to *B*.

- and *B* is decidable, then *A* is decidable .
- and *A* is undecidable, then *B* is undecidable.
- and *B* is r.e., then *A* is r.e .
- and *A* is not r.e., then *B* is not r.e.

Notice that
$A \leq_m B$
is the same as
$\overline{A} \leq_m \overline{B}$
, to show that *B* is not recognizable we may show that
$A_{TM} \leq_m \overline{B}$

Time Complexity, the class P

Let
$t : \mathbb{N} \rightarrow \mathbb{N}$
be a function, we define TIME(*t(n)*) , to be:
$TIME(t(n)) = \{L$
| *L* is a language decided by a *O(t(n))* time TM}

Let *t(n)* be a function, where
$t(n) \geq n$
. Then every *t(n)* multi-tape TM has an equivalent
$O(t^2(n))$

time single-tape TM.

$$P = \bigcup_k TIME(n^k)$$

Some examples of languages in this class:

- | *G* is a directed graph that has a directed path from *s* to *t*}
- | *x* and *y* are relatively prime}
- Every context free language is a member of *P*

The Class NP

Verifier:
A verifier for a language *A* is an algorithm *V*, where:
$A = \{w$

| *V* accepts
$<w,c>$

for some string *c*}
Time of a verifier is measured in terms of the length of *w*, so a polynomial time verifier runs in polynomial time in the length of *w*. A language *A* is polynomially verifiable if it has a polynomial time verifier

NP is the class of languages that have polynomial time verifiers.

Let
$t: \mathbb{N} \rightarrow \mathbb{N}$

be a function, we define NTIME(*t(n)*) , to be:
$NTIME(t(n)) = \{L$

| *L* is a language decided by a *O(t(n))* time NTM}

$$NP = \bigcup_k NTIME(n^k)$$

Cook-Levin theorem:
$SAT \in P <=> P = NP$

Poly-time computable function:
A function

$f : \Sigma^* \to \Sigma^*$

is a poly-time computable function if a polynomial time TM *M* exists that halts with just *f(w)* on its tape, when started on any input *w*.

Poly-time mapping reducible:
Language *A* is poly-time mapping reducible to language *B* written
$A \le_p B$
, if there is a poly-time computable function
$f : \Sigma^* \to \Sigma^*$
, where for every *w*,
$w \in A \Leftrightarrow f(w) \in B$
the function *f* is called the polynomial time reduction of *A* to *B*.

If
$A \le_p B$
and
$B \in P$
then
$A \in P$

NP-Complete:

A language *B* is NP-Complete if it satisfies two conditions

1. *B* is in NP
2. Every *A* in NP is poly-time reducible to *B*

If only the last condition holds, the problem is **NP-Hard**.

Some examples of well-known NPC problems:

- | *G* is an undirected graph with a *k*-clique}
- | ϕ is a satisfiable Boolean formula}
- | ϕ is a satisfiable Boolean formula}
- | *G* is an undirected graph that has a *k*-node vertex cover}
- | *G* has cut of size *k* or more}
- | *G* is a directed graph with a Hamilton path from *s* to *t*}
- | and for some , we have }

Space Complexity

Let
$f : \mathbb{N} \to \mathbb{N}$
be a function. The space complexity classes SPACE(*f(n)*) and NSPACE(*f(n)*) are defined as follows:

$SPACE(f(n)) = \{L$

| *L* is a language decided by a *O(f(n))* space TM}

$NSPACE(f(n)) = \{L$

| *L* is a language decided by a *O(f(n))* space NTM}

Savitch's Theorem:

For any function

$f : \mathbb{N} \to \mathbb{N}$

, where

$f(n) \geq \log n$

,

$NSPACE(f(n)) \subseteq SPACE(f^2(n))$

The class PSPACE is the class of languages that are decidable in polynomial space on a deterministic TM, in other words:

$$PSPACE = \bigcup_k SPACE(n^k)$$

We have the following series of containments:

$$P \subseteq NP \subseteq PSPACE = NSPACE \subseteq EXPTIME = \bigcup_k TIME(2^{n^k})$$

PSPACE-Complete:

A language *B* is PSPACE-Complete if it satisfies two conditions

1. *B* is in PSPACE
2. Every *A* in PSPACE is poly-time reducible to *B*

Some examples of well-known PSPACE hard problems:

- | ϕ is a true fully quantified Boolean formula}
- | Player E has a winning strategy in the formula game associated with ϕ}
- | Player I has a winning strategy for the generalized geography game played on graph *G* starting at node *b*}

The Classes L and NL

L is the class of languages that are decidable in logarithmic space on a deterministic TM. The class NL is its non-deterministic counterpart:

L = SPACE(log *n*)

NL=NSPACE(log *n*)

Configuration of *M* on *w*:
If *M* is a TM that has a separate read-only input tape and *w* is an input a configuration of *M* on *w* is a setting of the state, the work tape and the positions of the two tape heads. The input *w* is not a part of the configuration of *M* on *w*

Log space transducer:
A log space transducer is a TM with a read-only input tape, a write only output tape, and a read/write work tape. The work tape may contain O(log *n*) symbols. A log space transducer *M* computes a function
$f : \Sigma^* \rightarrow \Sigma^*$
, where *f(w)* is the string remaining on the output tape after *M* halts with *w* on its input tape. We call *f* a log *space computable function*.

Log space reducible:
Language *A* is log space reducible to language *B*, written
$A \leq_L B$
, if *A* is mapping reducible to *B* using a log space computable function *f*

NL-Complete:

A language *B* is NL-Complete if it satisfies two conditions
1. *B* is in NL
2. Every *A* in NL is log space reducible to *B*

An example of a NL complete problem:
$PATH = \{<G, s, t>$
| *G* is a directed graph that has a directed path from *s* to *t*}

We have the following relations for the class L and NL:
$L = NL, NL = coNL, NL \subseteq P$

Intractability

Space constructible:
A function
$f : \mathbb{N} \rightarrow \mathbb{N}$
, where
$f(n) \geq \log n$
is called space constructible if the function that maps
1^n
(a string of *n* 1's) to the binary representation of *f(n)* is computable in space O(*f(n)*).

Space Hierarchy Theorem:
For any space constructible function
$f : \mathbb{N} \to \mathbb{N}$
there exists a language *A* that is decidable in space O(*f(n)*) but not in space o(*f(n)*)

For any two function
$f_1, f_2 : \mathbb{N} \to \mathbb{N}$
, where
$f_1(n)$
is
$o(f_2(n))$
and
f_2
is space constructible,
$SPACE(f_1(n)) \subset SPACE(f_2(n))$

For any two real numbers
$0 \le \varepsilon_1 < \varepsilon_2$
, we have
$SPACE(n^{\varepsilon_1}) \subset SPACE(n^{\varepsilon_2})$

As an extra result we have
$NL \subset PSPACE$
, and
$PSPACE \subset EXPSPACE$

Time Constructible:
A function
$t : \mathbb{N} \to \mathbb{N}$
where
$t(n) \ge \log n$
is called time constructible if the function that maps maps
1^n
(a string of *n* 1's) to the binary representation of *t(n)* is computable in time O(*t(n)*).

Time Hierarchy Theorem:
For any space constructible function
$t : \mathbb{N} \to \mathbb{N}$
there exists a language *A* that is decidable in time O(*t(n)*) but not in time o(*t(n)*/log *t(n)*)

For any two function

$t_1, t_2 : \mathbb{N} \to \mathbb{N}$

, where

$t_1(n)$

is

$o(t_2(n)/\log t_2(n))$

and

t_2

is time constructible,

$TIME(t_1) \subsetneq TIME(t_2(n))$

For any two real numbers

$0 \le \varepsilon_1 < \varepsilon_2$

, we have

$TIME(n^{\varepsilon_1}) \subsetneq TIME(n^{\varepsilon_2})$

From this it is easy to see that

$P \subsetneq EXPTIME$

EXPSPACE-Complete:

A language *B* is EXPSPACE-Complete if it satisfies two conditions

1. *B* is in EXPSPACE
2. Every *A* in EXPSPACE is polynomial time reducible to *B*

An example of a language that is EXPSPACE complete:

$EQ_{REX\uparrow} = \{ \langle Q, R \rangle$

| *Q* and *R* are equivalent regular expressions with exponentiation}