COT6410 Topics for Final Exams

Formal Languages

Recognition models
(Finite States, Single Stack (can be stateless, Linear Bounded, TM)

Grammar hierarchy
Regular
DFA == NFA
Exponential explosion when go from DFA to NFA
Arden’s Theorem: \( R = Q + RP \), \( Q \) does not contain lambda, \( R = QP^* \)
Myhill-Nerode (Consequences: unique min state and alternative to PL)

CFLs
DPDA != NPDA (PDA)
Ambiguity (inherent versus just incidental to a grammar)
Reduced Grammars and CNF (implications not constructions)
\( O(N^3) \) CFL parser based on CNF grammar – Dynamic Programming
Incorrect traces (related result on complement of \( ww \))

Pumping Lemmas (What they are; not their proofs or applications)

CSLs
DLBA (LBA) == NLBA
Trace languages
Closure and non-closure: Meta approach with intersection with regular and subst.
Why does it fail on CSLs?

PSLs (RE Languages)
DTM (TM) == NDTM (provided time is irrelevant)
What happens when sets interact: Can we get Regular, CFL non-Reg, CSL non-
CFL, RE non-CSL?

Decidable Problems and why they are decidable
Examples: Membership, Emptiness, \( \Sigma^* \)

Various operations on CSLs, CFLs and Regular Languages
Examples: Union, Intersection, Quotient, Complement, Prefix, Suffix, Substring
Computability Theory
Models of computation and required elements (divergence, ability to branch on absence/presence)
Determinism vs non-determinism; why non-det is not always better
Relationships between rec, re, co-re, re-complete, non-re/non-co-re
Proofs about relations, e.g., re & co-re iff rec; re iff semi-dec.;
inf. rec iff range of monotonically increasing total function
Various operations on non-re/non-co-re, re and recursive sets (Examples Sum, Product)
Use of quantified decidable predicates to get upper bound on complexity
Reduction (many-one); degrees of unsolvability (many-one)
Rice’s Theorem (including its proof)
Applications of Rice’s Theorem; when does it fail?
Proof of re-completeness (re and known re-complete reduces to problem)
Trace languages (CSL) and complement of trace languages (CFL)
  L = \Sigma^* for CFL, L \neq \emptyset for CSL
  For CFL L, L = L^2?  
  For CFL L, does there exist an n such that L^n = L^{n+1}?
Post Correspondence Problem
Semi-Thue word problem to PCP (No details, quick pathway)
  Other rewrite models: Post Canonical, Thue, Post Normal, Tag
PCP and context free grammars
  From any PCP instance, P, can specify CFGs, G1 and G2, such that
  L(G1) \cap L(G2) \neq \emptyset iff P has a solution
  Merging these together to new grammar G with start symbol S and rule
  S \rightarrow S1 | S2 where S1 is start symbol of G1 and S2 is start symbol of G2
  we have that G is ambiguous iff P has a solution
PCP and context sensitive grammars
  From any PCP instance, P, can specify CSG, G, such that
  L(G) \neq \emptyset iff P has a solution; it is also the case that L(G) is infinite if so
  Note that this is second proof of undecidability of emptiness for CSG
PSG
  Given TM, M, can specify PSG, G, such that L(G) = L(M)
  Every PSL is homomorphic image of a CSL
  Closure of CSL’s under \lambda-free homomorphisms
Quotient
  Given TM, M, specify CFGs, G1 and G2, such that L(G1) / L(G2) = L(M)
  Consider terminal traces (even/odd; odd/even correctness)
More Computability Theory

Two-Variable Implication Calculus
- Starts with axioms and rules of inference
- Derivation versus refutation
- MP and Substitution versus Resolution (great for refutation but incomplete for Derivation/Inference)
- Constrained to no associativity
- Reduce HALT to deciding what theorems follow from some set of axioms
  - Representation as two stacks, each of which uses a composition (not simple linear) encoding.
    - One variable is used for left stack (state, scanned symbol, right side of tape)
    - Other variable for left side of tape
    - “Shape” of outer expression form is contents of top of stack
    - Shape of substitution for variable in outer shape determines next item on stack and so on
    - Bottom of stack has its own special shape, so we know when stack is empty
- Constant execution time (uniform halting)
  - Notion of arbitrary starting point
  - Why is this re and not worse?
  - What is notion of an infinite rather than unbounded tape?
  - What is mortality and how does constant time TM relate to mortal TM?

Finite Power of CFLs
- Reducing is $L = \Sigma^*$ to is $L = L^2$
  - Remember start point is to check if $\Sigma \cup \{\lambda\}$
- Reducing traces that have a fixed maximum length to $\exists n \ L^n = L^{n+1}$
  - Remember trick of a language with three parts (bad traces, pairs of configs, $\{\lambda\}$)

Factor Replacement Systems with Residue
- Use residue to check for non-divisibility, thereby avoiding determinism
  - $2x + 1 \rightarrow 6x + 4$
  - $2x \rightarrow x$

Collatz Conjecture is that starting at any positive integer this eventually reaches 1 and cycles there on $1 \rightarrow 4 \rightarrow 2 \rightarrow 1$

Collatz Function
- $T(x) = x/2$ if $x$ even; $T(x) = 3x+1$ if $x$ is odd

- Reaches 1 for starting numbers up to $2^{68}$
- If a counterexample exists, it is greater than 300 quintillion!!!!!!
Complexity Theory

P, NP (verification vs non-det. solution), co-NP, NP-Complete
Polynomial many-one versus polynomial Turing reductions
Problems I will focus on
Polynomial-time bounded NDTM to SAT (basic idea)
SAT to 3-SAT; 3SAT to Independent Set problem (IS) for undirected graph
3SAT to SubsetSum; SubsetSum to Partition
Integer Linear Programming Feasibility
  Is there an assignment that satisfies the constraints?
  SAT (not necessarily 3SAT) and 0-1 case.
  k-vertex cover, k-coloring (3-coloring),
    Optimization versions: min vertex cover; min coloring
Knapsack is limited to one bin and asks for best fit (values & weights)
  SubsetSum optimization problem for \( \leq G \) when weight and value are same
BinPacking allows multiple bins and optimizes number of bins of some fixed size
Scheduling with fixed number (p) of processors and no deadlines
  Goal is to finish all tasks as soon as possible
  This is an optimization version of a p-partition problem
Deadline scheduling
  BinPacking uses all items in list so list could be times of tasks leading to an
    Optimization problem to minimize the number of processors while
    obeying a deadline
Scheduling heuristics and anomalies
Unit execution scheduling of tree/forest and of anti-tree/anti-forest
Hamiltonian circuit (cycle)
  Traveling Salesman adds distances (weights); seeks circuit of distance \( \leq K \)
    Reduce HC to TSP set K to |V| and distances to 1 where links
    and to K+1 otherwise
  Optimization version looks for minimum distance circuit
Knapsack 0-1 Problem
  Dynamic Programming (differing dimensions – n,W/O(n*W) vs n/O(2^n))
Tiling the plane (basic concepts)
  Halting problem to Tiling (really complement of Halting)
Polynomial step bounded NDTM to Bounded Tiling
Bounded PCP based on Semi-Thue simulation of NDTM (NP-Complete)
Weighted MaxCut and Partition
Parallels and non-parallels to Recursive, RE, RE-Complete,
  Co-RE, Co-RE-Complete, RE-Hard (Turing versus many-one reductions)
NP-Easy, NP-Hard, NP-Equivalent
NP-Equivalent Optimization Problems associated with
  SubsetSum (max SubsetSum less than a Goal value) –
    reduction using power of two values
K-Coloring (min coloring) – binary search
2SAT
- Use of Implication Graph and SCC (Strongly Connected Components)
- Positive Min-Ones 2SAT and relation to VC (Vertex Cover)
- NP-Equivalence based on VC

Finding Triangle Strips is NP-Complete (NP-Hard by reducing Hamiltonian Path to Triangle Strips)
Weakly versus Strongly NP-Hard/Complete
- $P \subseteq NP \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}$
- $\not\subseteq 2\text{-EXPTIME} \not\subseteq 3\text{-EXPTIME} \not\subseteq \ldots \not\subseteq \text{ELEMENTARY} \not\subseteq \text{PRF} \not\subseteq \text{REC}$

$P \neq \text{EXPTIME}$; At least one of these is true
- $P \not\subseteq NP$; $NP \not\subseteq \text{PSPACE}$; $\text{PSPACE} \not\subseteq \text{EXPTIME}$

$NP \neq \text{NEXPTIME}$; at least one of these is true
- $NP \not\subseteq \text{PSPACE}$
- $\text{PSPACE} \not\subseteq \text{EXPTIME}$
- $\text{EXPTIME} \not\subseteq \text{NEXPTIME}$

- Note that $\text{EXPTIME} = \text{NEXPTIME}$ iff $P=NP$
- Note that $k\text{-EXPTIME} \not\subseteq (k+1)\text{-EXPTIME}$, $k>0$

$\text{PSPACE} \neq \text{EXPSPACE}$; At least one of these is true
- $\text{PSPACE} \not\subseteq \text{EXPTIME}; \text{EXPTIME} \not\subseteq \text{EXPSPACE}$

ATM (Alternating Turing Machine) – This is just concept stuff with no details
- $\text{AP} = \text{PSPACE}$, where AP is solvable in polynomial time on an ATM
- QSAT is solvable by an alternating TM in polynomial time and polynomial space (Why?)
- QSAT is PSPACE-Complete
- Petri net reachability is EXPSPACE-hard and requires 2-EXPTIME
- Presburger arithmetic is at least in 2-EXPTIME, at most in 3-EXPTIME, and can be solved by an ATM with $n$ alternating quantifiers in doubly exponential time

Savitch’s Theorem: $\text{NPSpace}(f(n)) \subseteq \text{DSpace}(f(n)^2)$
- Uses extreme time-space tradeoff – we don’t care about time, only space
- Limit depth of recursion in search for path from start to ending configuration
- Do this by a recursive binary search using all possible intermediaries
- Bad for time but good for max level of recursion
- Assume space is $\log N$ (valid for retaining node number or SAT assignments)
- Time for DFS (non-det or det) is $O(N)$.
- Space for non-det is $\log N$; for det is $N \log N$ (why?)
- With ignoring time can get $(\log N)^2$ space. Shows poly growth.
- Time is $O(N^{\log N})$ – big time tradeoff

Functional Equivalent of P/NP
- FP, FNP, TFNP
Constraints

Promise Problems
Example is 4-coloring (planar is Promise Set; rest are maybes)
Promise set of VALUE(f,x,t) when STP(f,x,t) is true

CLP(R)

Khot’s Conjecture
Graph Coloring with pairwise constraints is NP-Hard even when we know there is a coloring that satisfies almost all constraints, and we just need a coloring that satisfies a small percentage
if Khot’s conjecture is true and P ≠ NP, then NP-Hard problems not only require exponential time but also getting good, generally applicable, polynomial-time approximations is hard