COT6410 Topics for Final Exam

Computability Theory

Some Formal Language Material

- Pumping Lemmas (What they are; not their proofs or applications)
- Myhill-Nerode (What it says and its implications, not its proof or applications)
- Arden’s Theorem: \( R = Q + RP \), \( Q \) does not contain lambda, \( R = QP^* \)
- Reduced Grammars and CNF (implications not proofs)
- \( O(N^3) \) CFL parser based on CNF grammar – Dynamic Programming
- Decidable Problems and why they are decidable (Examples: Membership in Regular Languages and CFLs; Emptiness of Regular Languages and CFLs)
- Various operations on CSLs, CFLs and Regular Languages (Examples Union, Intersection)
- Relations between rec, re, co-re, re-complete, non-re/non-co-re
- Proofs about relations, e.g., re & co-re => decidable;
  union of re and rec is re but can be rec
- Various operations on non-re/non-co-re, re and recursive sets (Examples Sum, Product)
- Use of quantified decidable predicates to categorize complexity
- Reduction (many-one); degrees of unsolvability (many-one)
- Rice’s Theorem (including its proof)
- Applications of Rice’s Theorem
- Proof of re-completeness (re and known re-complete reduces to problem)
- Basic decidability results in formal grammars
- Trace languages (CSL) and complement of trace languages (CFL)
  \( L = \Sigma^* \) for CFL, \( L \neq \emptyset \) for CSL
  For CFL \( L \), \( L = L^2 \) ?

Post Correspondence Problem

Semi-Thue word problem to PCP (No details, just that it’s so and is a quick pathway)

PCP and context free grammars

- From any PCP instance, \( P \), can specify CFGs, \( G_1 \) and \( G_2 \), such that
  \( L(G_1) \cap L(G_2) \neq \emptyset \) if \( P \) has a solution
- Merging these together to new grammar \( G \) with start symbol \( S \) and rule
  \( S \rightarrow S_1 | S_2 \) where \( S_1 \) is start symbol of \( G_1 \) and \( S_2 \) is start symbol of \( G_2 \) 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, \( M \), can specify CFGs, \( G_1 \) and \( G_2 \), such that \( L(G_1) / L(G_2) = L(M) \)
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?
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 (usually with 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)
Travelling Salesman adds distances (weights) and 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 (looking at different dimensions – \(n, W/O(n*W)\) versus just \(n/O(2^n)\))

Tiling the plane (basic concepts)
Halting problem to Tiling
Polynomial step bounded NDTM to Bounded Tiling
Bounded PCP based on Semi-Thue simulation of NDTM (NP-Complete)
MaxCut (weighted) 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 \( \subseteq \) PSPACE = NPSpace \( \subseteq \) EXPTIME \( \subseteq \) NEXPTime \( \subseteq \) EXPSPACE
\( \not\subseteq \) 2-EXPTIME \( \not\subseteq \) 3-EXPTIME \( \not\subseteq \) … \( \not\subseteq \) ELEMENTARY \( \not\subseteq \) PRF \( \not\subseteq \) REC

P \( \not= \) EXPTime; At least one of these is true
P \( \not\subseteq \) NP
NP \( \not\subseteq \) PSPACE
PSPACE \( \not\subseteq \) EXPTime

NP \( \not= \) NEXPTime
Note that EXPTime = NEXPTime iff P=NP
Note that k-EXPTIME \( \not\subseteq \) (k+1)-EXPTIME, \(k>0\)
PSPACE \( \not= \) EXPSpace; At least one of these is true
PSPACE \not\subseteq \text{EXPTIME}
\text{EXPTIME} \not\subseteq \text{EXPSPACE}

ATM (Alternating Turing Machine) – This is just concept stuff with no details
AP = PSPACE, where AP is solvable in polynomial time on an ATM
Q SAT is solvable by an alternating TM in polynomial time and polynomial space (Why?)
Q SAT 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: \( \text{NPSPACE}(f(n)) \subseteq \text{DPSPACE}(f(n)^2) \)
Uses extreme time-space tradeoff – we don’t care about time, only space
Limit depth of recursion in search for path from starting to ending configuration
Do this by a recursive binary search using all possible intermediaries
Bad for time but good for max level of recursion

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 \( P \neq \text{NP} \), then NP-Hard problems not only require exponential time
but also getting good, generally applicable, polynomial-time approximations is hard

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 TM to determining what Theorems can be proved from an arbitrary set of axioms
We use 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)
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 relate to notion of a mortal TM?

Finite Power of CFLs
Reducing is \( L = \Sigma^* \text{ to is } L = L^2 \)
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, \( \{\lambda\} \))