INF2080 2018 Exam

Sensorveiledning

1. Regular Expression (5 points)

One correct expression is:

$$(I'm \cup ((NA)^{16})^{+})Batman!$$

Grading notes: no point subtractions for missing spaces, 2 points off for writing NA^{16} instead of $(NA)^{16}$.

2. Context-Free Grammar (5 points)

A possible grammar is

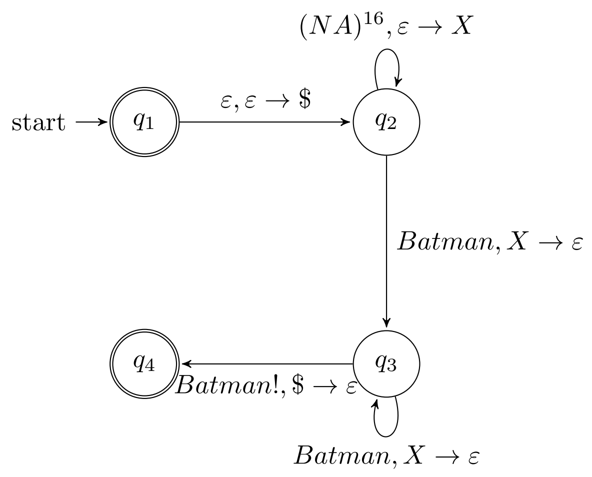
$$S \to XBatman! \mid I'mBatman!$$
$$X \to (NA)^{16} \mid XX$$

3 points for correct (NA) part, 2 points for correct "I'm" part.

Grading notes: no point subtractions for missing spaces, 1 point off for writing NA^{16} instead of $(NA)^{16}$. 2 points off if missing "I'm" part.

3. Pushdown Automaton (5 points)

Standard PDA for $a^n b^n$ slightly modified (e.g., p. 115 in the book):


$(NA)^{16}, \varepsilon \to X$
start
q_1
$\varepsilon, \varepsilon \to \$$
q_2
$Batman, X \to \varepsilon$
q_4
q_3
$Batman!, \$ \to \varepsilon$
$Batman, X \to \varepsilon$

Grading notes: 3 points for correct notation and pushing/popping an "empty stack" symbol on to stack in the first and last transitions. 1 point off for wrong placement of exclamation point. 1 point off if it accepts "Batman!", i.e., n=0. No points off for no mention of implicit sink states.

4. Nonregular Languages, I (5 points)

L_1 is the complement of the language $L = \{w | w \in \{a, b\}^*$ is a palindrome$\}$. This language is known to be context free. Since the class of regular languages is closed under complementation, if L_1 were regular, then so is L, a contradiction.

Alternatively, one can use the pumping lemma directly for L_1 (very hard). For instance, the string $a^p b a^{p!+p}$ cannot be pumped: By conditions 2 and 3, y must consist of only a's. Thus $y = a^k$ for $1 \leq k \leq p$. Since $p!$ is a multiple of all integers between 1 and p, there exists i such that $ik = p!$. By pumping y i times, the resulting string is $a^{p!+p} b a^{p!+p}$, which is a palindrome.

Grading notes: If they use the first (and intended) argument: If they say L is context-free hence L_1 must be context free too, 0 points total (CFLs are not closed under complementation... and this is not an argument that solves the exercise,even if this was the case). 1 point off if they fail to mention that the class of regular languages is closed under complementation. It is not necessary to show that palindromes are nonregular.

If they attempt a pumping lemma proof: 1 point off for any lemma condition they disregard. Incorrect if they argue that a string cannot be pumped for a specific partition of the string (the statement must hold for any partition). In particular: the string $a^p b^p$ does NOT work! By choosing $x = \varepsilon, y = a, z = a^{p-1} b^p$, the string can be pumped (you can only "pump down" one step, i.e., lose a single a in this circumstance)!

5. Nonregular languages, II (5 points)

Standard pumping lemma proof, analogous to $a^n b^n$. For example, one can choose $a^p b^{p+1}$. Then by condition 3, y must consist of only a's. Pumping up (i.e., for $i \geq 2$) leads to a word of the form $a^k b^{p+1}$ for $k \geq p+1$. This is not in L_1.

Grading notes: 1 point off if they disregard any of the conditions. Incorrect if they argue that a string cannot be pumped for a given partition of the string (the statement must hold for any partition).

6. Addressable Memory TM (10 points)

A sufficient, high-level solution need not go into specific tape read/write operations. It is sufficient to describe the following tasks: On a GOTO transition, the tape 1 head does the current cell modifications according to the transition, is moved all the way to the left, and then moved to the right a number of times equal to the address on tape 2. After this, tape 2 does the modifications as described in the GOTO transition.

Grading notes: 2 points off if they forget that the second tape must be able to be changed as well, i.e., they don't mark the current position on tape 2, don't perform alterations to tape 2 after moving. If they go into specifics about tape operations, these need to be correct (i.e., simply decrementing tape 2's address without storing a backup is not OK). A more low-level description of the GOTO simulation:

There are a few things the 2 tape Turing machine must do in order to simulate the GOTO transition:

STEP 1: modify current cell on tape 1 according to current transition

STEP 2: jump to cell with address given by tape 2

STEP 3: alter address on tape 2 according to current transition, move head 2

The first step is standard 2 tape behavior. Thus, after modifying the current tape 1 cell, the following steps simulate step 2:

1. mark current position on tape 2.
2. copy tape 2 contents, such that if 0101 was the tape contents, then 0101#0101 is the new contents
3. move head 1 to the left side of tape.
4. Decrease number to the right of # on tape 2 by 1, move head 1 one step to the right.
5. Repeat 4. until only 0s to the right of # on tape 2.

6. Remove # and zeros to its right, return to head 2 to position marked in step 1.

Then head 1 is in the correct cell, the second step is complete. step 3 is again standard 2-tape behavior.

7. Undecidability, I (3 points)

This can be solved either directly, or by using known theorems.

Direct proof: Assume for undecidable L that $\overline{L}$ has a decider D. Construct a new decider that on input w runs D and flips its result. This is a decider for L, contradiction.

Using known results: We have seen that the class of decidable languages is closed under complementation. Thus, the complement of an undecidable language cannot be decidable.

8. Undecidability, II (5 points)

LOOP is essentially the complement of HALT, i.e., $HALT = \overline{LOOP} \cap L$, where L is just the language containing all strings $\langle M, w \rangle$ where M is a valid Turing machine. L is decidable (discussed during the lecture), hence if LOOP is decidable then so is HALT, contradiction.

Alternatively, showing directly: Assume we have a decider D for LOOP. Create a new decider that runs D on input $\langle M, w \rangle$: If D accepts, reject. If D rejects, accept. This is a decider for HALT, contradiction.

Grading notes: If they ignore the fact that the complement of LOOP also contains strings that are not representations of TM's, deduct 1 point.

9. Busy Beavers (7 points)

If BB is a computable function, then there exists a decider D that on input $\langle n, k \rangle$ halts with $BB(n, k)$ on its tape. Then we can create a decider D' for HALT (or alternatively LOOP, by flipping accept/reject):

On input $\langle M, w \rangle$:

1. Determine n the number of states and k the number of tape symbols in M.
2. calculate $BB(n, k)$.

3. Simulate M on w for $BB(n, k)$ steps. If M does not halt, reject. Otherwise, output what M does.

Grading notes: 1 point off if step 1 of the machine is left out, 2 points off if they leave out step 2. Must explain (somehow) that step 2 terminates (use that it is computable).

10. P and NP (5 points)

Show that P = NP implies that NP = coNP.

If P = NP, then they are the same class. P is known to be closed under complement (as are all deterministic classes), that is, for any language in P, the complement of this language is also in P. coNP is the class of languages whose complements are in NP, hence NP = coNP under our assumption.

Partial credit guide: 1-2 points lost for missing definitions of coNP/complementation. No loss for missing argument for why P is closed under complement.

11. Subgraph isomorphism (10 points)

Need to prove both membership and hardness. For membership in NP, sufficient to give a polynomial-time verifiable certificate. The subgraph claimed to be isomorphic is NOT sufficient, as isomorphism is not known to be checkable in polynomial time. However, we can use the a subgraph and a the bijection as the certificate. We can check that the graph in the certificate is a subgraph of G in linear time (look up edges and vertices in the input), and we can check that the bijection f (a list of vertex pairs $\langle v_g, v_h \rangle$) is a bijection (one to one and onto) likewise in linear time.

For hardness, easiest reduction is from clique (does a graph A contain a clique of size k?), an NP-complete problem known from the course. Given a graph A and a size k, output A and a clique of size k. This is obviously doable in polynomial time, and A has a clique of size k if and only if A has a subgraph isomorphic to a clique of size k.

Partial credit: 3 points lost for subgraph without bijection in membership proof. 1-2 points loss for missing time complexity analysis/lack of precision.

Correct membership worth 4, correct hardness 6. 2 points given if only definition of NP-completeness is correct.

12. Powerful reductions (10 points)

Need to show membership and hardness. Every finite language is in P, including the one given, since we can store all the yes-strings in our turing machine and check if a given string is one of them. This can be done in linear time in the size of the input.

Hardness: Need to show that any language in P reduces to the one given in polynomial time. Let A be an arbitrary language in P. There is a DTM that decides A in polynomial time. Given a string w, run this DTM on w. If it accepts, output ”aaa”, if it rejects, output ”abb”. Since the DTM runs in polynomial time, this reduction also runs in polynomial time. Furthermore, since A was arbitrary, such a reduction exists for any language A in P.

Partial credit: Membership is worth 3 points, hardness 7. 1-3 points loss for missing precision.

13. polyL, part 1 (10 points)

Assume that there exists a complete problem A for polyL. Since polyL is a union of classes, this problem must be a member of SPACE($(\log n)^k$) for some k. Since A is complete under logspace reductions, every other problem B in polyL reduces to A in SPACE($\log n$). By the given lemma, SPACE($(\log n)^k$) is closed under logspace reductions, so B is also in SPACE($(\log n)^k$). Hence, all problems in polyL are in this class, and polyL = SPACE($(\log n)^k$).

Partial credit: 1-3 points loss for missing precision. Placing the complete problem at a fixed level of polyL alone is worth 5 points.

14. polyL, part 2 (10 points)

Assume for contradiction that polyL has a complete problem. By the result in part 1, polyL = SPACE($(\log n)^k$) for some k. This implies that SPACE($(\log n)^{k+1}$) = SPACE($(\log n)^k$), since polyL is also the infinite union of SPACE($(\log n)^c$) for all c. However, $(\log n)^k = o((\log n)^{k+1})$, and therefore by the space hierarchy theorem, SPACE($(\log n)^k$) is a strict subset of SPACE($(\log n)^{k+1}$). This contradicts our earlier conclusion that SPACE($(\log n)^{k+1}$) = SPACE($(\log n)^k$).

Partial credit: 4 points for correctly deriving an equality between two distinct space classes. 4 more for correctly applying the space hierarchy theorem. 2 left for missing precision.