Midterm Exam for Course 1-18
"Tree Automata and Applications"
Solution

January 26, 2010

Exercise 1.

1. (a) We first show that **MI** is NP-easy for deterministic automata.
   The non-deterministic polynomial time works as follows:
   i. guess for each variable x appearing in t an accessible state q_x of the automaton.
   ii. verify if $t[\{x \mapsto q_x\}_{x \in vars(t)}] \rightarrow^* q_f$ for some final state q_f.

   The above non-deterministic algorithm obviously terminates in polynomial time. It is therefore sufficient to prove its correctness:
   i. if the algorithm answers 'yes', then we can construct a ground instance t' of t which is accepted by the automaton by considering witnesses t_x of the accessibility of q_x (i.e. ground terms such that $t_x \rightarrow^* q_x$) and letting $t' = t[\{x \mapsto t_x\}_{x \in vars(t)}]$.
   ii. if there is a ground instance t' of t which is accepted by the automaton, then we consider the substitution σ of minimal domain such that $t' = t\sigma$. As the automaton is deterministic, then there exists at most one accessible state q_x such that $\sigma(x) \rightarrow^* q_x$. Furthermore, as t' is accepted by the automaton, there must exist exactly one such state q_x for each x (if there were no such state, we couldn't build any run on $\sigma(x)$ (and therefore neither on t')). Therefore the guess in the first step of the algorithm will find these states q_x such that the second step (the verification step) works.

   (b) We now show that **MI** is NP-hard by giving a Karp reduction to 3SAT.
   We consider the signature $\{true/0, false/0, +/1, \text{-}/1, \vee/2, \wedge/2\}$. Terms over this signature intuitively represent boolean expressions: *true* and *false* are the usual boolean constants, $+$ is the identity function, - is the negation function, $\vee$ is the logical or and $\wedge$ is the logical and.
   We consider as usual a tree automata (let us call it A) which evaluates boolean expressions as follows:

$$\begin{array}{ll} true \rightarrow q_t & false \rightarrow q_f \\ +(q_t) \rightarrow q_t & +(q_f) \rightarrow q_f \\ \text{-}(q_t) \rightarrow q_f & \text{-}(q_f) \rightarrow q_t \\ \wedge(q_f, q_t) \rightarrow q_f & \wedge(q_t, q_f) \rightarrow q_f \\ \wedge(q_t, q_t) \rightarrow q_t & \wedge(q_f, q_f) \rightarrow q_f \\ \vee(q_f, q_t) \rightarrow q_t & \vee(q_t, q_f) \rightarrow q_t \\ \vee(q_t, q_t) \rightarrow q_t & \vee(q_f, q_f) \rightarrow q_f \end{array}$$

   We will define q_t to be the final state of the automaton.

For each 3SAT problem

$$\bigwedge_{i\in\{1,\ldots,n\}} (\alpha_i x_{\alpha_i} \vee \beta_i x_{\beta_i} \vee \gamma_i x_{\gamma_i})$$

we let the term t be:

$$\wedge(\ldots \wedge (\alpha_1 x_{\alpha_1} \vee \beta_1 x_{\beta_1} \vee \gamma_1 x_{\gamma_1}, \alpha_2 x_{\alpha_2} \vee \beta_2 x_{\beta_2} \vee \gamma_2 x_{\gamma_2}) \ldots, \alpha_n x_{\alpha_n} \vee \beta_n x_{\beta_n} \vee \gamma_n x_{\gamma_n})$$

It is immediate to see that there is a ground instance of t accepted by A iff the answer to the 3SAT problem is yes.

2. A deterministic automaton rewrites every term to at most one state. Therefore there is no need to test the intersection of the languages of some states; it is sufficient to guess one state for each variable.

Exercise 2. (An application)

1. Show that if $L = \{\mathsf{enc}(\mathsf{pair}(a,b),\mathsf{pub}(b)), \mathsf{enc}(\mathsf{priv}(b),\mathsf{pub}(c)), \mathsf{priv}(c)\}$ then $\mathsf{pair}(b,b)$ is deducible ($\mathsf{pair}(b,b) \in \mathit{ded}(L)$).

   We have that

$$\begin{array}{lcl} \mathsf{enc}(\mathsf{priv}(b),\mathsf{pub}(c)), \mathsf{priv}(c) & \vdash & \mathsf{priv}(b) \\ \mathsf{enc}(\mathsf{pair}(a,b),\mathsf{pub}(b)), \mathsf{priv}(b) & \vdash & \mathsf{pair}(a,b) \\ \mathsf{pair}(a,b) & \vdash & b \\ b, b & \vdash & \mathsf{pair}(b,b) \end{array}$$

2. As L is regular, there exists a tree automaton $A = (\Sigma, Q, Q_f, \Delta)$ accepting L. We assume w.l.o.g. that all states in Q are accessible. Starting from A, we will construct an ϵ-automaton accepting $\mathit{ded}(A)$ as follows.

   Initially, we let $A' = (\Sigma, Q \cup \{q_f\}, \{q_f\}, \Delta')$ (where q_f is a new state (not appearing in Q)) and where

$$\begin{array}{lcll} \Delta' = \Delta & \cup & \{q \to q_f\}_{q\in Q_f} & \\ & \cup & \{\mathsf{pair}(q_f, q_f) \to q_f\} & \textit{corresponding to } x_1, x_2 \vdash \mathsf{pair}(x_1,x_2) \\ & \cup & \{\mathsf{enc}(q_f, q_f) \to q_f\} & \textit{corresponding to } x_1, x_2 \vdash \mathsf{enc}(x_1,x_2) \end{array}$$

   and we apply the following saturation process until a fixpoint is reached:

   (a) for every pair of states (q_1, q_2) such that $\mathsf{pair}(q_1,q_2) \to^* q_f$, we add the transitions $q_1 \to q_f$ and $q_2 \to q_f$, corresponding to the deduction rules $\mathsf{pair}(x_1,x_2) \vdash x_1$ and $\mathsf{pair}(x_1,x_2) \vdash x_2$.

   (b) for every constant c in Σ_0 and for every state q, if $\mathsf{enc}(q,\mathsf{pub}(c)) \to^* q_f$ and if $\mathsf{priv}(c) \to^* q_f$, we add the transition $q \to q_f$, corresponding to the deduction rule $\mathsf{enc}(x,\mathsf{pub}(a)), \mathsf{priv}(a) \vdash x$.

   (c) similarly, for every constant c in Σ_0 and for every state q, if $\mathsf{enc}(q,\mathsf{priv}(c)) \to^* q_f$ and if $\mathsf{pub}(c) \to^* q_f$, we add the transition $q \to q_f$, corresponding to the deduction rule $\mathsf{enc}(x,\mathsf{priv}(a)), \mathsf{pub}(a) \vdash x$.

   As $Q \cup \{q_f\}$ is finite, the set of transitions is also finite and therefore the above saturation process terminates in finite time. Let A'' be the automaton obtained from A' by applying the above saturation rules.

   We now prove that $L(A'') = \mathit{ded}(L(A))$.

We first show that $ded(L(A)) \subseteq L(A'')$. Indeed, assume $t \in ded(A)$. We show that $t \in L(A'')$ by induction on the size of the proof that $t \in ded(A)$. If $t \in L(A)$, then obviously $t \in L(A'')$ (base case). Otherwise (inductive case), there exist $k \in \{1, 2\}$, $t_1, \ldots, t_k \in ded(L(A))$ such that $t_1, \ldots, t_k \vdash t$ and such that the proofs that $t_1, \ldots, t_k$ are in $ded(L(A))$ are shorter than the proof that $t \in ded(L(A))$. By the induction hypothesis, we have that $t_1, \ldots, t_k$ are in $L(A'')$. By case analysis on the deduction rule $t_1, \ldots, t_k \vdash t$, we conclude that t is also in $L(A')$.

Reciprocally, we prove that $L(A'') \subseteq ded(L(A))$. It is easy to see that $L(A') \subseteq ded(L(A))$. It is sufficient to notice that the saturation rules (a), (b) and (c) are *sound*, i.e. if only deducible terms were accepted by the automaton before application of the rule, then only deducible terms are accepted afterwards as well.

Exercise 3.

1. We consider $L = \{f(\Box, \Box)\}$. Then $L^\star$ is the set of trees over Σ where every (complete) branch of the tree has the same length. This set is obviously not regular (there is no way to remember the unbounded height of a branch in a finite number of states).

2. Let $A = (\Sigma, Q, Q_f, \Delta)$ be a tree automaton for L. We construct A', an ϵ-automaton for L^*, as follows:
$$A' = (\Sigma, Q \cup \{q_f\}, \{q_f\}, \Delta')$$
where q_f is a new state (not appearing in Q) and where
$$\begin{array}{rcl} \Delta' = \Delta & \cup & \{q \to q_f\}_{q \in Q_f} \\ & \cup & \{\Box \to q_f\} \\ & \cup & \{q_f \to q\}_{\Box \to_\Delta^* q} \end{array}$$
We prove that $L^* = L(A')$ by showing each of the inclusions.

We first show that $star_2(L, i) \subseteq L(A')$ for every i, by induction on i. As $\Box \to q_f \in \Delta'$, we have that $star_2(L, 0) \subseteq L(A')$ (base case). Otherwise (inductive case), we have to show that $L \cdot star_2(L, i) \subseteq L(A')$, knowing that $star_2(L, i) \subseteq L(A')$. Let $t \in L$ be arbitrary. It is sufficient to show that $t \cdot star_2(L, i) \subseteq L(A')$. We consider a run r of A on t and runs $r_1, \ldots, r_n$ of A' on some terms $t_1, \ldots, t_n \in star_2(L, i)$, where n is the number of $\Box$ in t. Then it is easy to use $r, r_1, \ldots, r_n$ to construct a run of A' on $t[t_1, \ldots, t_n]$ ($t[t_1, \ldots, t_n]$ denotes the term obtained by replacing every $\Box$ in t by the corresponding t_i).

We prove the other direction, namely that $L(A') \subseteq L^\star$ or equivalently, that for every $t \in L(A')$, there exists an i such that $t \in star_2(L, i)$. We show this by induction on the size of t. Let r be a run of A' on t (we assume each node is labeled with a sequence of states representing the ϵ-transitions that took place in that node) and let $p_1, \ldots, p_n$ be the positions in the run labeled with a sequence containing q_f. Let $t' = ((t[\Box]|_{p_1})[\Box]|_{p_2} \ldots)[\Box]|_{p_n}$ (we assume $|_p$ has not effect on the term if the position is outside of the legal range of positions). By the induction hypothesis for each $t|_{p_j}$ there exists a i_j such that $t|_{p_j} \in star_2(L, i_j)$. Furthermore, it is easy to see that t' is accepted by A. Therefore, it is sufficient to choose $i = max\{i_1, \ldots, i_n\} + 1$ to conclude.

3. First we note that the $\cdot$ operation is associative on unary trees: $(t_1 \cdot t_2) \cdot t_3 = t_1 \cdot (t_2 \cdot t_3)$, so we do not need parenthesis to write $t_1 \cdot t_2 \cdot \ldots \cdot t_n$. Also, it is easy to see that $\Box$ is a neutral element for $\cdot$: $t \cdot \Box = \Box \cdot t = t$.

We let $L' = \{w_1(w_2(\ldots w_n(\Box))) \mid w_1(w_2(\ldots w_n(\Box))) \in L\}$ (i.e. the subset of unary trees in L ending with $\Box$). We by induction prove that $star_1(L, i) = star_2(L, i) = \{\Box \cdot t_1 \cdot \ldots \cdot t_{n-1} \mid n \leq i, t_1, \ldots, t_{n-2} \in L', t_{n-1} \in L\}$.

For $i = 0$ (base case), $star_1(L, 0) = star_2(L, 0) = \{\Box\}$. For $i > 0$ (inductive case), we have that $star_1(L, i-1) = star_2(L, i-1) = L_{i-1}$ where

$$L_{i-1} = \{\Box \cdot t_1 \cdot \ldots \cdot t_{n-1} \mid n \leq i-1, t_1, \ldots, t_{n-2} \in L', t_{n-1} \in L\}.$$

We show that $L_{i-1} \cup L \cdot L_{i-1} = L_i = L_{i-1} \cup L_{i-1} \cdot L$. We show each equality by showing the double inclusion. Indeed, it is trivial to see that $L_{i-1} \subseteq L_i$. It remains to show that:

(a) $L \cdot L_{i-1} \subseteq L_i$
(b) $L_{i-1} \cdot L \subseteq L_i$
(c) $L_i \subseteq L_{i-1} \cup L \cdot L_{i-1}$
(d) $L_i \subseteq L_{i-1} \cup L_{i-1} \cdot L$

Each of these inclusions follows easily from the definition of L_i.

Exercise 4.

1. Let $n = |\mathcal{D}|$. We consider the ES2S formula

$$\exists X_1, \ldots, X_n : \bigwedge_{\substack{i \in \{1,\ldots,n\} \\ j \in \{1,\ldots,n\}}} (\forall x \in X_i, \forall y \in X_j : H(x,y) \wedge V(x,y))$$

where

$$H(x,y) = S_1(x,y) \implies \bigwedge_{a,b,c,d \in \{1,\ldots,k\}} (L_{a,b,c,d}(x) \implies \bigvee_{a',b',c' \in \{1,\ldots,k\}} L_{a',b',c',b})$$

and where

$$V(x,y) = S_2(x,y) \implies \bigwedge_{a,b,c,d \in \{1,\ldots,k\}} (L_{a,b,c,d}(x) \implies \bigvee_{a',b',d' \in \{1,\ldots,k\}} L_{a',b',a,d'}).$$

The formula $H(x,y)$ checks that if x and y are horizontally adjacent, then the quarters of x and y that "touch" have the same color. Similarly, $V(x,y)$ checks that if x and y are vertically adjacent, then the quarters of x and y that "touch" have the same color. $L_{a,b,c,d}(x)$ is a predicate that states that x is colored with a on the top, b on the right, c on the bottom and d on the left. $\bigvee_{\ldots \in \{1,\ldots,k\}}$ and $\bigwedge_{\ldots \in \{1,\ldots,k\}}$ are syntactic sugar for large disjunctions and respectively conjunctions.

2. By (1) and (ii), ES2S and domino languages are equivalent. By (i), ES2S is not closed under complement. But S2S is obviously closed under complement. Therefore ES2S and S2S cannot possibly be equally expressive.

Exercise 5. (Generalization of a theorem of Greibach to tree languages)

Let us consider a class $\mathcal{C}$ of tree languages over a signature Σ containing all regular tree languages, closed under union and under the application of symbols: for all $L_1, L_2 \in \mathcal{C}$, all $f \in \Sigma$, $f(L_1, L_2) := \{f(t_1, t_2) \mid t_1 \in L_1, t_2 \in L_2\}$ is in $\mathcal{C}$.

1. Let $A = (\Sigma, Q, Q_f, \Delta)$ be an automaton for L. We consider

$$A' = (\Sigma, Q \cup \{q_f\}, \{q_f\}, \Delta')$$

an ϵ-automaton (where q_f is a state not already appearing in Q) with

$$\Delta' = \Delta \cup \{q \to q_f \mid f(q'', q) \to q' \in \Delta \text{ with } q' \in Q_f \text{ and } t \to_\Delta^* q''\}$$

Then A' recognizes $L_{/t} = \{t' \mid f(t, t') \in L\}$.

2. Assume that we can solve the regularity problem for languages in $\mathcal{C}$. To verify if L is empty, it is sufficient to check regularity of $f(L, L')$ (where L' is a non-regular set in $\mathcal{C}$).

   Proof. If $f(L, L')$ is regular, then necessarily L is empty: otherwise L', the projection on the second argument of f would be regular, yielding a contradiction. Conversely, if L is empty, then $f(L, L')$ is empty and therefore regular.

3. Assume that we can solve the regularity problem for languages in $\mathcal{C}$. To verify if L is universal, it is sufficient to check regularity of $f(\mathcal{T}(\Sigma), L') \cup f(L, \mathcal{T}(\Sigma))$, where $\mathcal{T}(\Sigma)$ is the universal language (it is regular, therefore in $\mathcal{C}$).

   Proof. If L is universal, then $L_1 = f(\mathcal{T}(\Sigma), L') \cup f(L, \mathcal{T}(\Sigma))$ is also universal and therefore regular. Conversely, if L is not universal, there exists a term $t \in \mathcal{T}(\Sigma) \setminus L$. Then L_1 cannot be regular since $L_{1/t} = L'$ would then also be regular by Item 1, yielding a contradiction.

4. The class $\mathcal{C}'$ of closures of tree automata by the axiom of associativity for a binary symbol f is obviously closed under union and under the application of the function symbol f. Therefore we can apply Item 3.

   Assume regularity is decidable for $\mathcal{C}'$.

   We consider a context-free word language L_0 over an alphabet Σ_0 and we construct a tree automaton A over $\{f : 2\} \cup \Sigma_0$ such that $\mathit{Yield}(L(A)) = L_0$.

   Then L_0 is universal if $\mathcal{A}_f(L(A))$ is universal, where $\mathcal{A}_f$ denotes the associative closure of a language for the associative symbol f_f. But $\mathcal{A}_f(L(A)) \in \mathcal{C}'$. As we assumed regularity was decidable for $\mathcal{C}'$, by Item 3, we also know universality to be decidable for $\mathcal{C}'$.

   Therefore, to test if L_0 is universal, it is sufficient to check if $\mathcal{A}_f(L(A))$ is universal. As we obtained a decision procedure for an undecidable problem (the problem of universality of an arbitrary context-free word language L_0), we conclude that our supposition was false and that therefore regularity is not decidable for $\mathcal{C}'$.