Spring, 2014 CIS 511

Introduction to the Theory of Computation
Jean Gallier

Homework 6

April 15, 2014; Due May 1, 2014

"A problems" are for practice only, and should not be turned in.

Problem A1. Prove that every context-free language is a recursive set.

Problem A2. Consider the definition of the Kleene T-predicate given in the notes in Definition 5.4.1.

(i) Verify that $T(x, y, z)$ holds iff x codes a RAM program, y is an input, and z codes a halting computation of P_x on input y.

(ii) Verify that the Kleene normal form holds:

$$\varphi_x(y) = \mathrm{Res}[\min z(T(x, y, z))].$$

"B problems" must be turned in.

Problem B1 (80 pts). In this problem, the fundamental property of LR-parsing (due to Knuth) is established.

For simplicity, let us consider context-free grammars without ϵ-rules. Given a reduced context-free grammar $G = (V, \Sigma, P, S')$ augmented with start production $S' \to S$, where S' does not appear in any other productions, the set C_G of *characteristic strings of* G is the following subset of V^* (watch out, not Σ^*):

$$C_G = \{\alpha\beta \in V^* \mid S' \underset{rm}{\Longrightarrow}^* \alpha B v \underset{rm}{\Longrightarrow} \alpha\beta v, \\ \alpha, \beta \in V^*, v \in \Sigma^*, B \to \beta \in P\}.$$

In words, C_G is a certain set of prefixes of sentential forms obtained in rightmost derivations: those obtained by truncating the part of the sentential form immediately following the rightmost symbol in the righthand side of the production applied at the last step.

The fundamental property of LR-parsing is that C_G is a *regular language*. A nondeterministic automaton N_{C_G} accepting C_G can be constructed according to the method described in Section 1 of the handout *A Survey of LR-Parsing Methods, etc.*. Please, review this construction.

(i) Let G be the following grammar:

$$\begin{aligned} S' &\to E \\ E &\to E + T \mid T \\ T &\to T * F \mid F \\ F &\to (E) \mid a \end{aligned}$$

with $\Sigma = \{+, *, (,), a\}$.

Give the automaton N_{C_G} for the grammar G.

(ii) Using the standard algorithms, give a deterministic finite automaton equivalent to N_{C_G}. Do not include the "dead state".

(iii) You shall now prove that $L(N_{C_G}) = C_G$!

(1) Prove the following claim by induction on the length of rightmost derivations:

Claim 1: For any nonterminal A, for every rightmost derivation

$$A \underset{rm}{\Longrightarrow}^* \alpha B v \underset{rm}{\Longrightarrow} \alpha\beta v,$$

where $v \in \Sigma^*$, $B \in N$, and $\alpha, \beta \in V^*$, if we denote the first production in the above rightmost derivation as $A \to \delta$, then there is a computation on input $\alpha\beta$ from state $A \to$ "."δ to the final state $B \to \beta$"."$.$

To prove this claim, you will have to show the following (think about it in terms of parse trees). For any nonterminal A, every rightmost derivation from A is either of the form

(i) $A \underset{rm}{\Longrightarrow} \delta$, for some production $A \to \delta$, in which case $A = B$ and $\delta = \beta$, or of the form

(ii) $A \underset{rm}{\Longrightarrow} \lambda B_i \rho \underset{rm}{\Longrightarrow}^* \lambda B_i w \underset{rm}{\Longrightarrow}^* \lambda\alpha_i B w_i w \underset{rm}{\Longrightarrow} \lambda\alpha_i\beta w_i w$, with $w, w_i \in \Sigma^*$, $A, B, B_i \in N$, $\lambda, \rho, \alpha_i, \beta \in V^*$, and where

$$B_i \underset{rm}{\Longrightarrow}^* \alpha_i B w_i \underset{rm}{\Longrightarrow} \alpha_i\beta w_i \quad \text{and} \quad \rho \underset{rm}{\Longrightarrow}^* w.$$

Let $B_i \to \delta_i$ be the first production applied in the rightmost derivation from B_i. In the first case, there is a computation in N_{C_G} from state $A \to$ "."δ to the final state $A \to \delta$"." (where again, $A \to \delta = B \to \beta$), and in the second case, there is a computation in N_{C_G} from state $A \to$ "."$\lambda B_i \rho$ to $B_i \to$ "."δ_i on input λ, and a computation from state $B_i \to$ "."δ_i to the final state $B \to \beta$"." on input $\alpha_i\beta$.

Conclude that C_G is a subset of $L(N_{C_G})$.

(2) Prove the following claim by induction on the number of ϵ-transitions in a computation in N_{C_G}:

Claim 2: For any state $A \to$ "."δ, if there is a computation on input γ to some final state $B \to \beta$"."$,$ then there is some rightmost derivation $A \underset{rm}{\Longrightarrow}^* \alpha B v \underset{rm}{\Longrightarrow} \alpha\beta v$, such that, the production applied in the first rightmost derivation step is $A \to \delta$, and $\gamma = \alpha\beta$.

For this, prove the following:

(i) Either $\gamma = \delta$ and the computation is from state $A \to \text{“.”}\delta$ to state $A \to \delta\text{“.”}$, or

(ii) δ is of the form $\lambda B_i \rho$, γ is of the form $\lambda\alpha_i\beta$, and there is a computation on input $\alpha_i\beta$ from some state of the form $B_i \to \text{“.”}\delta_i$ to the final state $B \to \beta\text{“.”}$, and a rightmost derivation as in Claim 1.

Conclude that $L(N_{C_G})$ is a subset of C_G, thus establishing that $C_G = L(N_{C_G})$.

Problem B2 (30 pts). Let $\Sigma = \{a_1, \ldots, a_k\}$ be some alphabet and suppose $g, h_1, \ldots, h_k$ are some total functions, with $g\colon (\Sigma^*)^{n-1} \to \Sigma^*$, and $h_i\colon (\Sigma^*)^{n+1} \to \Sigma^*$, for $i = 1, \ldots, k$. If we write $\overline{x}$ for $(x_2, \ldots, x_n)$, for any $y \in \Sigma^*$, where $y = a_{i_1} \cdots a_{i_m}$ (with $a_{i_j} \in \Sigma$), define the following sequences, u_j and v_j, for $j = 0, \ldots, m+1$:

$$\begin{aligned} u_0 &= \epsilon \\ u_1 &= u_0 a_{i_1} \\ &\vdots \\ u_j &= u_{j-1} a_{i_j} \\ &\vdots \\ u_m &= u_{m-1} a_{i_m} \\ u_{m+1} &= u_m a_i \end{aligned}$$

and

$$\begin{aligned} v_0 &= g(\overline{x}) \\ v_1 &= h_{i_1}(u_0, v_0, \overline{x}) \\ &\vdots \\ v_j &= h_{i_j}(u_{j-1}, v_{j-1}, \overline{x}) \\ &\vdots \\ v_m &= h_{i_m}(u_{m-1}, v_{m-1}, \overline{x}) \\ v_{m+1} &= h_i(y, v_m, \overline{x}). \end{aligned}$$

(i) Prove that

$$v_j = f(u_j, \overline{x}),$$

for $j = 0, \ldots, m+1$, where f is defined by primitive recursion from g and the h_i's, that is

$$\begin{aligned} f(\epsilon, \overline{x}) &= g(\overline{x}) \\ f(ya_1, \overline{x}) &= h_1(y, f(y, \overline{x}), \overline{x}) \\ &\vdots \\ f(ya_i, \overline{x}) &= h_i(y, f(y, \overline{x}), \overline{x}) \\ &\vdots \\ f(ya_k, \overline{x}) &= h_k(y, f(y, \overline{x}), \overline{x}), \end{aligned}$$

for all $y \in \Sigma^*$ and all $\overline{x} \in (\Sigma^*)^{n-1}$. Conclude that f is a total function.

(ii) Use (i) to prove that if g and the h_i's are RAM computable, then the function, f, defined by primitive recursion from g and the h_i's is also RAM computable.

Problem B3 (10 pts). Prove that the function, $f\colon \Sigma^* \to \Sigma^*$, given by

$$f(w) = a_1^{|w|}$$

is primitive recursive ($\Sigma = \{a_1, \ldots, a_N\}$).

Problem B4 (30 pts). *Ackermann's function* A is defined recursively as follows:

$$\begin{aligned} A(0,\, y) &= y + 1, \\ A(x+1,\, 0) &= A(x,\, 1), \\ A(x+1,\, y+1) &= A(x,\, A(x+1, y)). \end{aligned}$$

Prove that

$$\begin{aligned} A(0,\, x) &= x + 1, \\ A(1,\, x) &= x + 2, \\ A(2,\, x) &= 2x + 3, \\ A(3,\, x) &= 2^{x+3} - 3, \end{aligned}$$

and

$$A(4, x) = 2^{2^{\cdot^{\cdot^{\cdot^{2^{16}}}}}\Big\}x} - 3,$$

with $A(4, 0) = 16 - 3 = 13$. Equivalently (and perhaps less confusing)

$$A(4, x) = 2^{2^{\cdot^{\cdot^{\cdot^{2^{2}}}}}\Big\}x+3} - 3.$$

Problem B5 (20 pts). Prove that the following properties of partial recursive functions are undecidable:

(a) A partial recursive function is a constant function.

(b) Two partial recursive functions φ_x and φ_y are identical.

(c) A partial recursive function φ_x is equal to a given partial recursive function φ_a.

(d) A partial recursive function diverges for all input.

Problem B6 (30 pts). Prove that it is undecidable whether a context-free grammar generates a regular language.

Problem B7 (50 pts). Let $\mathcal{NEXP}$ be the class of languages accepted in time bounded by $2^{p(n)}$ by a nondeterministic Turing machine, where $p(n)$ is a polynomial. Consider the

problem of tiling a $2s \times s$ rectangle, described in Section 7.5 of the notes (slides on the web), but only with a single initial tile σ_0, and assume that s is given in binary.

(i) Prove that the above tiling problem is $\mathcal{NEXP}$-complete.

(ii) Now, consider the problem of tiling the *entire upper half-plane*, starting with a single tile σ_0 (of course, the set of tile patterns, $\mathcal{T}$, is finite). More precisely, this problem is said to have a solution if for every $s > 1$, there a function σ_s tiling the $2s \times s$-rectangle.

Problem B8 (60 pts). Let A be any $p \times q$ matrix with integer coefficients and let $b \in \mathbb{Z}^p$ be any vector with integer coefficients. The 0-1 *integer programming problem* is to find whether the system

$$Ax = b$$

has any solution, $x \in \{0, 1\}^q$.

(i) Prove that the 0-1 integer programming problem is in $\mathcal{NP}$.

(ii) Prove that the 0-1 integer programming problem is $\mathcal{NP}$-complete by providing a polynomial-time reduction from the bounded-tiling problem. **Do not try to reduce any other problem to the** 0-1 **integer programming problem**.

Hint. Given a tiling problem, $((\mathcal{T}, V, H), \widehat{s}, \sigma_0)$, create a 0-1-valued variable, x_{mnt}, such that $x_{mnt} = 1$ iff tile t occurs in position (m, n) in some tiling. Write equations or inequalities expressing that a tiling exists and then use "slack variables" to convert inequalities to equations. For example, to express the fact that every position is tiled by a single tile, use the equation

$$\sum_{t \in \mathcal{T}} x_{mnt} = 1,$$

for all m, n with $1 \leq m \leq 2s$ and $1 \leq n \leq s$.

(iii) Prove that the restricted 0-1 integer programming problem in which the coefficients of A are 0 or 1 and all entries in b are equal to 1 is also $\mathcal{NP}$-complete.

TOTAL: 310 points.