CS 581: Theory of Computation
Final exam

James Hook

December 5, 2005

This is a closed-book, closed-notes exam. All problems have equal weight.

1. Relating classes of languages.
   (a) Define P, NP, and NP complete.
   (b) Define Turing-decidable and Turing-recognizable.
   (c) What is a verification problem?
   (d) How do verification problems relate P and NP?
   (e) How do verification problems relate Turing-decidable and Turing-recognizable?

2. For each of the following indicate if the statement is true, false, or an open problem.
   (a) P = NP
   (b) If $P(x, y)$ is decidable then $\{y|\exists x.P(x, y)\}$ is recognizable
   (c) In a reasonable proof system all true things are provable
   (d) In a reasonable proof system all provable things are true
   (e) All finite sets are regular
   (f) Regular languages are closed under intersection
   (g) Context Free languages are closed under intersection
   (h) Decidable languages are closed under intersection
   (i) Turing-recognizable languages are closed under intersection

3. Use an explicit construction to show that the regular languages are closed under intersection. Argue briefly that your construction is correct.

4. Turing's Y combinator is defined as:

$$(\lambda x.\lambda y.y(xxy))(\lambda x.\lambda y.y(xxy))$$

   Prove that this is a fixedpoint combinator. That is, show that

$$(YF) = F(YF)$$

   Recall the α rule:

$$\lambda x.M = \lambda y.M[y/x] \quad \text{provided } y \text{ does not occur free in } M.$$

   The β rule:

$$(\lambda x.M)N = M[N/x]$$

Substitution:

$$
\begin{array}{rcll}
z[M/x] & = & z \quad \text{if } x \neq z & \\
x[M/x] & = & M \quad \text{if } x = z & \\
(\lambda y.N)[M/x] & = & \lambda z.(N[z/y][M/x]) & \text{where } z \text{ is a new variable that is distinct from } x \text{ and does not occur in } N \text{ and is not free in } M. \\
(N_1 N_2)[M/x] & = & (N_1[M/x])(N_2[M/x]) &
\end{array}
$$

5. Recall Rice's theorem:

> Let P be a language consisting of Turing machine descriptions where P fulfills two conditions. First, P is nontrivial —it contains some, but not all, TM descriptions. Second, P is a property of the TM's language—whenever $L(M_1) = L(M_2)$, we have $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$. Here M_1 and M_2 are any TMs. P is an undecidable language.

Although we proved it in the context of Turing Machines, Rice's theorem can be proven for any acceptable programming system, $\phi_1, \phi_2, \phi_3, \ldots$. When given in the context an acceptable programming system the second property is restated in terms of function equality. It is stated: P is a property of indexes if whenever $\phi_i = \phi_j$ as partial functions $i \in P$ iff $j \in P$. (In lecture I may have described P as an index set.)

(a) Restate Rice's theorem in the vocabulary of the lambda calculus.

(b) A type system is a formal system that identifies sets of terms in a programming language that are "good" or "safe". Curry developed a type system for the lambda calculus in which all typable terms had a unique beta-normal form (that is, were unique up to alpha-conversion). For example, $\lambda x.x$ and $\lambda y.y$ are irreducible terms that are alpha-equivalent. Curry's type system is decidable. What does Rice's theorem say about the set of terms that Curry's type system accepts as type correct?