Procesadores de Lenguaje

Examen final, junio de 2010

1. Las expresiones del λ-cálculo (léase "lambda-cálculo") se generan por medio de la gramática siguiente:

$$E \rightarrow E\ E \quad | \quad L\ id\ P\ E \quad | \quad id \quad | \quad '('\ E\ ')'$$

Los "tokens" son L, que representa la letra griega λ que da nombre al cálculo; P, que representa un punto; id, que corresponde a variables (identificadores), y ambos paréntesis. Un ejemplo es $\lambda x.\lambda y.(x\,y\,z)$, cuyos "tokens" serían $L\,id\,P\,L\,id\,P\,(\,id\,id\,id\,)$. (Quien quede con curiosidad sobre el lambda-cálculo puede satisfacerla en la Wikipedia a la salida del examen.)

a/ (*1,5 puntos*) Calcula FIRST, FOLLOW, ANULABLE, y todos los estados del autómata LR(0).

b/ (*1,5 puntos*) Explica cómo se generarían y propagarían "look-aheads" para obtener el autómata LALR. ¿Qué diferencias existen entre las tablas de análisis ascendente LR(0), SLR y LALR para esta gramática? ¿Qué ambigüedad tiene esta gramática, y en qué se refleja en el autómata?

2. (*3 puntos*) Resolvemos la ambigüedad de esta gramática aplicando reducción, y no desplazamiento, para resolver los conflictos. Intuitivamente, la misión de la λ (la L de nuestra gramática) es "declarar parámetros": en una expresión de la forma $LidPE$, el identificador id queda "declarado" (el término clásico es "ligado") dentro de la expresión E. Por ejemplo, en $\lambda x.\lambda y.(x\,y\,z)$ las variables x e y aparecen declaradas (ligadas) en la expresión $(x\,y\,z)$ interior, mientras que la z no aparece declarada (el término clásico es "libre").

Añade a la gramática atributos y rutinas semánticas que permitan tener, en cada expresión, la lista de variables ligadas (declaradas) que aparecen en ella.

(La puntuación obtenida en los problemas 1 y 2 se sumará a la nota de curso, truncándose esta suma parcial a un máximo de 6; la nota final se obtiene sumando adicionalmente las notas de los problemas 3, 4 y 5, que aparecen a la vuelta de la hoja.)

3. (*1 punto*) Dispones de una máquina M en la que corren un compilador de C (para ella misma) y un intérprete de Python. Tienes un programa Q escrito en C que deseas hacer funcionar pero que depende de una librería de la que careces. Sin embargo, tienes algunos módulos escritos en Python ya disponibles, que cubren la misma funcionalidad que la librería en cuestión. Has encontrado en internet un "wrapper" que traduce código C a Python, y él mismo es capaz de encontrar los módulos sustitutos en las carpetas de Python. El "wrapper" está escrito en C, y lo tienes en fuente. Dibuja diagramas en T que representen todos los pasos necesarios para ejecutar Q.

4. (*1 punto*) Compara las fases de la compilación con el proceso de lenguas humanas. ¿Qué fases adicionales necesitas? ¿Qué dificultades adicionales se presentan?

5. (*2 puntos*) Consideramos un código de tres direcciones del formato usual (MIPS...). Un "bloque básico" es una secuencia de instrucciones de tres direcciones que sólo puede comenzar a ejecutarse por su comienzo, y no otro punto, y sólo puede terminar de ejecutarse en su final, y no en otro punto. Por ejemplo, una secuencia de operaciones constituye un bloque básico, a condición de no tener dentro ninguna etiqueta; pero, si aparece una etiqueta que permite saltar a "dentro" de la secuencia, o en ella hay un salto condicional o incondicional que sale del bloque en un punto distinto del final, no es un bloque básico.

Explica cómo se traduce a código de tres direcciones la instrucción siguiente, indicando cómo se formarían los bloques básicos.

```
while (a > b)
    {
        if (b < c)
            {
                b = c;
            }
        a--;
        b--;
    }
```