Lógica Computacional 2017-2
Práctica 3: Semántica de la lógica de primer orden

Lourdes del Carmen González Huesca
Roberto Monroy Argumedo
Fernando A. Galicia Mendoza

Facultad de ciencias, UNAM

Fecha de entrega: Martes, 14 de marzo del 2017

Esta práctica puede ser entregada en equipo: máximo dos personas.

Utilizando los módulos `LPO`, `LPOSust` hechos en el laboratorio y realiza lo indicado en las siguientes secciones.

1. Sustitución en fórmulas

Modifica el archivo `LPOSust` de tal forma que se pueda hacer sustituciones en fórmulas, para esto deberás realizar los siguientes ejercicios:

1. Define una función en el módulo `LPO` que devuelva la lista que represente el conjunto de variables libres de una fórmula, nómbrala `fv`.

2. Define una función en el módulo `LPO` que devuelva la lista que represente el conjunto de variables ligadas de una fórmula, nómbrala `bv`.

3. Define una función que dada una fórmula y una sustitución, realice la sustitución, es decir, dada φ elemento de tipo `Form` y $[\vec{x} := \vec{t}]$ elemento de tipo `Sust` se tiene que
`apsubF` $\varphi\ [\vec{x} := \vec{t}] = \varphi[\vec{x} := \vec{t}]$

2. Semántica de la lógica de primer orden

Define un archivo llamado `LPOSem.hs`, considera los siguientes tipos:

```
-- | IntF. Tipo que representa una interpretacion de formulas.
type IntF a = Nombre → [a] → a

-- | IntR. Tipo que representa una interpretacion de relaciones.
type IntR a = Nombre → [a] → Bool

-- | Estado. Tipo que representa el estado de una variable del universo.
type Estado a = Ind → a

-- | Mundo. Tipo que representa un mundo.
type Mundo a = (Estado a,IntF a, IntR a)
```

Y realiza los siguientes ejercicios:

1. Define una función que devuelva la actualización de estados.
   Nómbrala `actEstados`.

2. Define una función que devuelve la interpretacion de un término respecto a un estado.
   Nómbrala `iTerm`.

3. Define una función que devuelve la interpretación de una fórmula respecto a un mundo.
   Nómbrala `iForm`.
   Observación: Cuando el universo es finito, entonces los cuantificadores universal y existencial, pasan a ser una serie de conjunciones y disyunciones, respectivamente. Por lo que se sugiere utilizar la función `and` y `or` provistas en la biblioteca `List` del lenguaje.

3. Sustitución de términos en términos

En lógica no resulta natural la sustitución de términos en términos, es decir, $[\vec{t} := \vec{s}]$ con $\vec{t}, \vec{s}$ dos series de términos.
Sin embargo, este concepto si es natural, y necesario, en lenguajes de programación. Por ejemplo los editores de texto estructurados[1], cuando se requiere sustituir una expresión del código, el mecanismo interno del editor busca la expresión (que es un término) y realiza la sustitución por otra expresión.

En el módulo `LPOSust`, deberás implementar la sustitución de términos en términos, para esto realiza los siguientes ejercicios:

1. Define un tipo de datos que represente una sustitución de un término en un término.

2. Define una función de sustitución para que dado un término t y una sustitución $[s := r]$, con s, r dos términos, devuelva la sustitución $t[s := r]$.

[1] Ejemplos de editores de texto estructurados: Netbeans o Eclipse

4. Teoría

Los siguientes ejercicios pueden ser entregadas en el archivo `README` o bien impreso al inicio de la ayudantía el día de entrega de esta práctica.[2]

1. Define formalmente la aplicación de una sustitución de términos.
2. Utilizando tu definición anterior, realiza los siguientes cálculos (sin saltarte pasos):
   - $suma(1, x)[x := suc(suma(2, x))]$
   - $suc(1)[suc(1) := suc(suc(1))]$
   - $if(iZ(x), 3, 2)[isZ(x) := and(true, false)]$

   Considerar $suma, suc, if, iZ, and$ solo como funciones, es decir, para este ejercicio no nos interesa su significado, lo mismo para las constantes $1, 2, 3, true, false$ y la variable x.

Sugerencia: Se recomienda hacer primero esta sección antes de hacer la implementación en el lenguaje de programación `Haskell`. Por dos razones: Tener claro en teoría para poder implementarlo y se comparará la definición formal respecto a la implementación entregada.

5. Dato

En el último ejercicio se utilizó un lenguaje formal: el lenguaje de expresiones aritmético booleanas (EAB, pa' los cuates).

Este lenguaje resulta ser el lenguaje de programación mas pequeño, su estudio sirve para construir lenguajes mas especializados (funcional,imperativo,orientado a objetos) y permite un análisis agradable sobre la evaluación de expresiones para la obtención de resultados.

El mecanismo de sustitución de variables en términos es esencial para la evaluación de programas, ya que, la expresión `let` es una expresión primitiva en este lenguaje, cuya evaluación a groso modo es: Dada la expresión `let x = e1 in e2` evaluar `e1` hasta que sea una expresión mínima (llamemosle `e1'`), es decir, sea un número o un booleano; acto seguido, se hace la operación $e2[x := e1']$.

Para mayor información: Esperar la asignatura **Lenguajes de programación**.

Reglas:

- Todo archivo debe seguir los estándares establecidos en el laboratorio.
- Todas las funciones deben ser recursivas.
- De las bibliotecas brindadas por el lenguaje, únicamente se puede importar `List`.

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar conocimientos en la práctica. -Aristóteles

[2]En caso de ser impresa, se sugiere utilizar hojas de rehuso.