Lógica Computacional 2017-1
Práctica 2

Noé Salomón Hernández Sánchez
Albert M. Orozco Camacho
C. Moisés Vázquez Reyes
Diego Murillo Albarrán
José Roberto Piche Limeta

Se deja el: 6 de octubre
Se entrega el: **23 de octubre**
Facultad de Ciencias UNAM

En esta práctica van a implementar formas normales de la lógica de predicados. La nota 8 del curso les será de mucha ayuda para guiarse. Pueden implementar todas las funciones auxiliares que quieran pero deben comentarlas debidamente. Se da por hecho que las fórmulas de entrada no tienen cuantificadores múltiples ni vácuos.

1. Lógica de predicados

Se utilizará el tipo:

```
data Term = X VarIndex | Fn Name [Term] deriving Eq
```

para representar a los términos. Las constantes se representan como funciones que no reciben argumentos, es decir, funciones cuya lista de argumentos es vacía. El tipo *VarIndex* es un alias de *Integer*.

Las fórmulas están representadas con el tipo:

```
data Form = Top | Bot | Pr Name [Term] | Eq Term Term | Neg Form |
Or Form Form | And Form Form | Impl Form Form | Syss Form Form | All TVar Form |
Ex TVar Form deriving Eq
```

El constructor *Eq* se refiere a la igualdad de términos. El tipo *TVar* es un alias de *Term*, pero sólo se usará para el caso de variables.

Finalmente, se representan las sustituciones con el tipo:

```
type Sust = [(TVar,Term)]
```

La idea es que la sustitución $[x_1, x_2..., x_n := t_1, t_2, ..., t_n]$ se representa con la sustitución `[(X 1,t1),(X 2,t2),...,(X n,tn)]`.

2. Ejercicios:

- `varsT::Term->[VarIndex]`

  Devuelve todas las variables de un término dado. Sólo se devuelven los índices de las variables.

  Ejemplo:

  ```
  >varsT $ Fn ‘‘f’’ [X 1,Fn ‘‘g’’ [X 3, X 5], Fn ‘‘a’’ [], X 10]
  ```
  $[1, 3, 5, 10]$

- `fv::Form->[VarIndex]`

  Devuelve todas las variables libres de una fórmula dada. Sólo se devuelven los índices de las variables.

  Ejemplos:

  ```
  >fv $ And (Ex (X 2) $ Pr ‘‘P’’ [Fn ‘‘f’’ [X 1, X 2]]) (Pr ‘‘Q’’ [X 3, X 2])
  ```
  $[1,3,2]$
  ```
  >fv $ Impl (Ex (X 3) $ All (X 4) $ Pr ‘‘R’’ [X 0,X 1]) (Ex (X 5) $ Pr ‘‘P’’ [X 3])
  ```
  $[0,1,3]$

- `bv::Form->[VarIndex]`

  Devuelve todas las variables ligadas de una fórmula dada. Sólo se devuelven los índices de las variables.

  Ejemplos:

  ```
  >bv $ And (Ex (X 2) $ Pr ‘‘P’’ [Fn ‘‘f’’ [X 1, X 2]]) (Pr ‘‘Q’’ [X 3, X 2])
  ```
  $[2]$
  ```
  >bv $ Impl (Ex (X 3) $ All (X 4) $ Pr ‘‘R’’ [X 0,X 1]) (Ex (X 5) $ Pr ‘‘P’’ [X 3])
  ```
  $[3,4,5]$

- `apsubsT::Term->Sust->Term`

  Aplica la sustitución en términos.

  Ejemplo:

  ```
  >apsubsT (Fn ‘‘g’’ $ [X 1,X 2,Fn ‘‘f’’ [Fn ‘‘a’’ []]]) [(X 2, X 3)]
  g(X1,X3,f(a))
  ```

- `varsSust::Sust->[VarIndex]`

  Devuelve todas las variables de una sustitución dada. Sólo se devuelven los índices de las variables.

  Ejemplo:

  ```
  >varsSust [(X 1,Fn ‘‘f’’ [X 2,Fn ‘‘c’’ []]),(X 2,X 4),(X 4,X 6),(X 5, Fn ‘‘g’’ [X 1])]
  ```
  $[1,2,2,4,4,6,5,1]$

- `apsubsF::Form->Sust->Form`

  Aplica la sustitución en fórmulas.

  Ejemplos:

  ```
  >apsubsF (Impl (Pr ‘‘P’’ [X 1, X 2]) (Pr ‘‘Q’’ [Fn ‘‘a’’ []])) [(X 2, X 4)]
  P(X1,X4) → Q(a)
  >apsubsF (All (X 4) $ Impl (Pr ‘‘P’’ [X 1, X 2]) (Pr ‘‘Q’’ [X 3])) [(X 2, X 4)]
  ∀X4.[P(X1,X2) → Q(X3)]
  ```

- `renVL::Form->[VarIndex]->Form`

  Renombra las variables ligadas de tal manera que la lista de variables ligadas y la lista de variables que se pasa como argumento sean ajenas, y no haya cuantificadores de la misma variable con alcances ajenos.

  Ejemplos:

  ```
  >renVL (Impl (All (X 0) $ Pr ‘‘P’’ [X 1,X 0]) (Ex (X 0) $ Pr ‘‘Q’’ [X 0])) [0,1,3]
  (∀X4.[P(X1,X4)]) → (∃X5.[Q(X5)])
  >renVL (And (Pr ‘‘P’’ [X 0]) (All (X 0) $ Pr ‘‘Q’’ [X 0,X 1])) [0,1,3]
  P(X0) ∧ (∀X4.[Q(X4,X1)])
  ```

- `recF::Form->Form`

  Renombra las variables ligadas de tal manera que la lista de variables ligadas y la lista de variables libres sean ajenas, y no haya cuantificadores de la misma variable con alcances ajenos.

  Ejemplos:

  ```
  >recF (Impl (All (X 0) $ Pr ‘‘P’’ [X 1,X 0]) (Ex (X 0) $ Pr ‘‘Q’’ [X 0]))
  (∀X0.[P(X1,X0)]) → (∃X2.[Q(X2)])
  >recF (And (Pr ‘‘P’’ [X 0]) (All (X 0) $ Pr ‘‘Q’’ [X 0,X 1]))
  P(X0) ∧ (∀X2.[Q(X2,X1)])
  ```

- `elimImp::Form->Form`

  Elimina las implicaciones de una fórmula utilzando equivalencias lógicas.

  Ejemplo:

  ```
  >elimImp (Impl (All (X 0) $ Pr ‘‘P’’ [X 1,X 0]) (Ex (X 0) $ Pr ‘‘Q’’ [X 0]))
  ¬(∀X0.[P(X1,X0)]) ∨ (∃X0.[Q(X0)])
  ```

- `fnn::Form->Form`

  Dada una fórmula, se obtiene su forma normal negativa.

  Ejemplo:

  ```
  >fnn (Impl (All (X 0) $ Pr ‘‘P’’ [X 1,X 0]) (Ex (X 0) $ Pr ‘‘Q’’ [X 0]))
  (∃X0.[¬P(X1,X0)]) ∨ (∃X0.[Q(X0)])
  ```

- `fnp::Form->Form`

  Devuelve la forma normal prenex de una fórmula dada. Se da por hecho que antes se aplicó *fnn*.

  Ejemplo:

  ```
  >fnp (Impl (All (X 0) $ Pr ‘‘P’’ [X 1,X 0]) (Ex (X 0) $ Pr ‘‘Q’’ [X 0]))
  ∃X0.[∃X0.[¬P(X1,X0) ∨ Q(X0)]]
  ```

- `elimCuant::Form->(Int,Int,[Term])->Form`

  Esta función es auxiliar para calcular la forma normal Skolem de una fórmula dada. La idea es que el segundo argumento de la función recuerde cuál ha sido la última constante usada, el último símbolo de función usado, y las variables cuantificadas universalmente antes del primer cuantificador existencial, respectivamente. Se da por hecho que antes se aplicó *fnp*.

  Ejemplos:

  ```
  >elimCuant (Ex (X 2) $ Ex (X 0) $ Or (Neg $ Pr ‘‘P’’ [X 0]) (Pr ‘‘Q’’ [X 2])) (1,3,[])
  ¬P(c2) ∨ Q(c1)
  >elimCuant (Ex (X 2) $ Ex (X 0) $ Or (Neg $ Pr ‘‘P’’ [X 0]) (Pr ‘‘Q’’ [X 2])) (1,3,[X
  4])
  ¬P(f4(X4)) ∨ Q(f3(X4))
  ```

- `fns::Form->Form`

  Obtiene la forma normal Skolem de una fórmula.

  Ejemplos:

  ```
  >fns (Ex (X 2) $ Ex (X 0) $ Or (Neg $ Pr ‘‘P’’ [X 0]) (Pr ‘‘Q’’ [X 2]))
  ¬P(c1) ∨ Q(c0)
  >fns (All (X 1) $ Ex (X 2) $ Ex (X 0) $ Or (Neg $ Pr ‘‘P’’ [X 0]) (Pr ‘‘Q’’ [X 2]))
  ¬P(f1(X1)) ∨ Q(f0(X1))
  ```

- `fnc::Form->Form`

  Obtiene la forma normal conjuntiva de una fórmula dada. Se da por hecho que la fórmula está en fns.

  Ejemplo:

  ```
  fnc (Or (Pr ‘‘P’’ [X 1, Fn ‘‘a’’ []]) (And (Pr ‘‘Q’’ [X 1, X 2]) (Pr ‘‘R’’ [X 3])))
  (P(X1,a) ∨ Q(X1,X2)) ∧ (P(X1,a) ∨ R(X3))
  ```