Especificación formal con lógica de predicados. Lógica Computacional 2016-2, Nota de clase 5

Favio Ezequiel Miranda Perea Araceli Liliana Reyes Cabello
Lourdes Del Carmen González Huesca

3 de marzo de 2016

El proceso de especificación o traducción del español a la lógica formal no es siempre sencillo. Algunas frases del español no se pueden traducir o especificar de una manera completamente fiel a la lógica de predicados, como veremos en algunos ejemplos. Sin embargo el proceso es de gran importancia pues es la base de muchos métodos de especificación formal utilizados en inteligencia artificial o ingenieria de software, como son el lenguaje de especificación Z que es un lenguaje formal basado en la teoría de conjuntos de Zermelo-Fraenkel y en la lógica de predicados y que se usa en la industria, especialmente en sistema de alta integridad, como parte del proceso de desarrollo de software o hardware tanto en Europa como en Estados Unidos.

1. Algunos consejos

A continuación presentamos algunos consejos y observaciones que pretenden facilitar el proceso de especificación del español por medio de la lógica.

- Únicamente podemos especificar afirmaciones o proposiciones, no es posible traducir preguntas, exclamaciones, órdenes, invitaciones etcétera.
- La idea básica es extraer predicados a partir de los enunciados dados en español de manera que el enunciado completo se construya al combinar dichos predicados mediante conectivos y cuantificadores. Por ejemplo la frase *me gustan los tacos y las pizzas* debe entenderse como *me gustan los tacos y me gustan las pizzas*. Análogamente *iré de vacaciones a la playa o a la montaña* significa *iré de vacaciones a la playa ó iré de vacaciones a la montaña*.
- La conjunción "y" se traduce como $\wedge$. La palabra "pero" también corresponde a una conjunción aunque el sentido original del español se pierde. Por ejemplo *te doy dulces pero haces la tarea* sólo puede especificarse como *te doy dulces y haces la tarea*. Lo cual es diferente en el español.
- La disyunción es incluyente *comeremos pollo o vegetales* incluye el caso en que se coman ambos.
- Con la implicación hay que ser cautelosos, sobre todo en el caso de frases de la forma *A sólo si B* lo cual es equivalente con *Si no B entonces no A*, que a su vez es equivalente con *Si A entonces B*. Es un error común intentar especificar dicha frase inicial mediante $B \to A$.

- Si en el español aparecen frases como *para todos, para cualquier, todos, cualquiera,los, las etc.* debe usarse el cuantificador universal $\forall$

- Si en el español frases como *para algún, existe un, alguno,alguna, uno, una etc.* debe usarse el cuantificador existencial $\exists$.
  Importante: En ciertas ocasiones, frases en español que involucran *alguien, algo* deben especificarse con un cuantificador universal y no un existencial. Por ejemplo, el enunciado *si hay alguien demasiado alto entonces se pegará con el marco de la puerta* se puede reescribir en español como *cualquiera demasiado alto se pegará con el marco de la puerta*, lo cual nos lleva a $\forall x(A(x) \rightarrow C(x))$. El lector debe convencerse de que no es posible especificar esta oración usando un cuantificador existencial.

- Cualquier especificación compuesta que involucre cuantificadores puede formarse identificando en ella alguno de los cuatro juicios aristotélicos:

  - Universal afirmativo *Todo S es P* traducido mediante $\forall x(S(x) \rightarrow P(x))$.
  - Existencial afirmativo. *Algún S es P* traducido mediante $\exists x(S(x) \wedge P(x))$.
  - Universal negativo: *Ningún S es P* traducido mediante $\forall x(S(x) \rightarrow \neg P(x))$
  - Existencial negativo: *Algún S no es P* traducido mediante $\exists x(S(x) \wedge \neg P(x))$

  Por supuesto que en especificaciones complicadas S y P no serán predicados sino fórmulas compuestas.

- Pronombres como *él, ella, eso* no se refieren a un individuo particular sino que se usan como referencia a algo o alguien mencionado previamente, por lo que obtienen significado del contexto particular. Cuando un pronombre aparezca en un enunciado debe uno averiguar primero a quién o qué se refiere. Por ejemplo, en el enunciado *Martha es amiga de Lupita pero ella no es amiga de Karla* debe traducirse como *Martha es amiga de Lupita y Lupita no es amiga de Karla*. Similarmente, cuando necesitamos de variables, como en *hay un perro con manchas y él ladra en las mañanas* es un error tratar de traducir por separado *hay un perro con manchas* y *él ladra en las mañanas* puesto que lo que existe está ligado con lo que ladra por la conjunción, de manera que debe utilizarse una variable que modele esta conexión.

- Las variables no se mencionan en español, son sólo un formalismo para representar individuos, por ejemplo la fórmula $\forall x(M(x) \rightarrow T(x))$ debe escribirse en español como *Cualquier minotauro es troyano* y no como *para cualquier x, si x es minotauro entonces x es troyano*. Enunciados de esta forma sólo sirven como pasos intermedios en el proceso de traducción pues no forman parte del español correcto ni de la lógica formal. A este mismo respecto es prácticamente imposible que en una traducción del español figuren variables libres.

- Los esquemas $\forall x(A \rightarrow B)$ y $\exists x(A \wedge B)$ son de gran utilidad y bastante comunes. Menos comunes, aunque también adecuados, son los esquemas $\forall x(A \wedge B)$, $\forall x(A \vee B)$ y $\exists x(A \vee B)$.

- El esquema $\exists x(A \rightarrow B)$, si bien es una fórmula sintácticamente correcta, es extremadamente raro que figure en una traducción del español.

- El hecho de que se usen dos o más variables distintas no implica que éstas representen a elementos distintos del universo, de manera que para especificar dos individuos distintos no

es suficiente contar simplemente con variables distintas. Las fórmulas $\exists x P(x)$ y $\exists x \exists y (P(x) \wedge P(y))$ expresan ambas lo mismo, a saber que algo cumple P. Se debe agregar explícitamente que x e y tienen la propiedad de ser distintos, es decir $x \neq y$.

En las siguientes secciones damos diversos ejemplos de especificación formal con lógica de predicados.

2. Especificación de listas

En lenguajes de programación dado un tipo de datos A se define el tipo de datos de las listas con elementos de tipo A, denotado L_A recursivamente como sigue:

- La lista vacía, denotada *nil* es un elemento de L_A
- Si a es un elemento de A y ℓ es una lista entonces $cons(a, \ell)$ es un elemento de L_A.
- Son todas.

A continuación presentamos la especificación formal en lógica de predicados para este tipo de datos y para algunas de sus operaciones

- Sean *nil* un símbolo de constante para la lista vacía, $A^{(1)}, L_A^{(1)}$ símbolos de predicado para el tipo A y listas de A respectivamente y $cons^{(2)}$ un símbolo de función para la construcción de listas. La definición del tipo de datos de listas con elementos en A se formaliza como sigue:
  - $L_A(nil)$
  - $\forall x \forall y \big(A(x) \wedge L_A(y) \rightarrow L_A(cons(x, y))\big)$
  - $\forall x \Big(L_A(x) \rightarrow x = nil \vee \exists y \exists z \big(A(y) \wedge L_A(z) \wedge x = cons(y, z)\big)\Big)$

  Otras especificaciones acerca de listas son:
- La lista vacía no es construible con la función *cons*.

$$\forall x \forall y (cons(x, y) \neq nil)$$

- La concatenación de dos listas es una lista.
  - Funcionalmente: sea $app^{(2)}$ un símbolo de función que representa a la concatenación de dos listas. La especificación es

$$\forall x \forall y \big(L_A(x) \wedge L_A(y) \rightarrow L_A(app(x, y))\big)$$

  - Relacionalmente: sea $App^{(3)}$ un predicado ternario que simbolice a la relación "z es la concatenación de x con y". La especificación es

$$\forall x \forall y \forall z \big(L_A(x) \wedge L_A(y) \wedge App(x, y, z) \rightarrow L_A(z)\big)$$

De este ejemplo se observa que una especificación formal puede darse usando funciones o bien predicados o relaciones. Cúal opción elegir dependerá de los objetivos particulares del problema en cuestión.

- La concatenación de la lista vacía con cualquier otra lista ℓ es ℓ.
  - Funcionalmente:
    $$\forall x(app(nil, x) = x)$$
  - Relacionalmente:
    $$\forall x App(nil, x, x)$$
- La concatenación de una lista no vacía ℓ_1 con una lista cualquiera ℓ_2 es la lista cuya cabeza es la cabeza de ℓ_1 y cuya cola es la concatenación de la cola de ℓ_1 con ℓ_2.
  - Funcionalmente utilizando las funciones *cons* y *app*:
    $$\forall x \forall y \forall z \big(app(cons(x, y), z) = cons(x, app(y, z))\big)$$
  - Funcionalmente utilizando funciones $hd^{(1)}$ y $tl^{(1)}$ para cabeza y cola de una lista.
    $$\forall x \forall y \big(x \neq nil \rightarrow app(x, y) = cons(hd(x), app(tl(x), y))\big)$$
  - Relacionalmente usando el constructor de listas *cons*
    $$\forall x \forall y \forall w \forall z \big(App(x, y, w) \rightarrow App(cons(z, x), y, cons(z, w))\big)$$
  - Relacionalmente usando cabeza y cola:
    $$\forall x \forall y \forall w \forall z \big(App(tl(x), y, w) \rightarrow App(x, y, cons(hd(x), w))\big)$$
- La reversa de una lista es una lista.
  - Funcionalmente: sea rev un símbolo de función de índice uno que representa a la reversa de una lista. La especificación es:
    $$\forall x \big(L_A(x) \rightarrow L_A(rev(x))\big)$$
  - Relacionalmente: sea $Rev(x, y)$ un predicado binario que represente a la relación "y es la reversa de x". La especificación es:
    $$\forall x \forall y \big(L_A(x) \wedge Rev(x, y) \rightarrow L_A(y)\big)$$
- La reversa de la lista vacía es la lista vacía.
  - Funcionalmente:
    $$rev\, nil = nil$$
  - Relacionalmente:
    $$Rev(nil, nil)$$

- La reversa de una lista no vacía es la concatenación de la reversa de su cola con la lista que contiene a su cabeza.
  - Funcionalmente

$$\forall x \forall y \big(rev(cons(x, y)) = app(rev(y), cons(x, nil)) \big)$$

  - Funcionalmente usando cabeza y cola

$$\forall x \big(x \neq nil \rightarrow rev(x) = app(rev(tl(x)), cons(hd(x), nil)) \big)$$

  - Relacionalmente usando *cons*

$$\forall x Rev \big(cons(x, y), app(rev(y), cons(x, nil)) \big)$$

  - Relacionalmente usando cabeza y cola

$$\forall x \big(x \neq nil \rightarrow Rev(x, app(rev(tl(x)), cons(hd(x), nil))) \big)$$

3. El juego de Risk

Risk es un juego de mesa que modela estrategias de guerra en el cual pueden participar de 2 a 6 jugadores. Aquí vamos a representar algunos aspectos de la versión Risk II para computadora. En el mundo hay jugadores, paises, tarjetas y numeros.

- Los jugadores pueden ser humanos o virtuales manejados por la computadora. Aquí no haremos distinción. Algunos jugadores son *Barbacena, Bonaparte, Wellington, Solignac, Maransin, Marmont, Spencer, Mackenzie, Pach, Menelao, Mega, Charal, Polain, Ardilio, Taupin, Vauban*
- Los paises son algunos de los paises actuales y otros que no necesariamente corresponden a la división política actual.
- Las tarjetas se otorgan si un jugador hizo al menos una conquista en su turno. Al tener un jugador 5 tarjetas está obligado a cambiarlas. De acuerdo a la figura en la tarjeta (tropa,cañon, caballo) se reciben refuerzos por cada 3 tarjetas.
- Los números se usarán como auxiliar para contar.

Introducimos un lenguaje de predicados para representar el juego mediante ejemplos, usaremos nombres completos para constantes y nombres cortos para predicados para facilitar la correspondencia con el español.

- Solignac ha ocupado China `Oc(solignac,china)`
- Barbacena ha defendido con éxito Peru `DefEn(barbacena,peru)`
- Bonaparte ha ganado una tarjeta `Gt(bonaparte,1)`
- Vauban vence a Mackenzie en Quebec `Ven(vauban,mackenzie,quebec)`

- Taupin se retira de Japón `Ret(taupin,japon)`
- Spencer ha intercambiado tarjetas `Int(spencer)`
- Mackenzie ataca a D'erlon en Argentina desde Brasil `At(mackenzie,d'erlon,argentina,brasil)`
- Menelao recibe 10 refuerzos `Ref(menelao,10)`
- Vauban hace un movimiento táctico de Alaska a Ucrania `Tac(vauban,alaska,ucrania)`
- Spencer ha sido aniquilado por Bonaparte `Ani(bonaparte,spencer)`
- 3 tropas se colocan en Francia `Tro(3,francia)`
- 5 tropas se mueven de Quebec a Groenlandia `Mov(5,quebec,groenlandia)`
- Maransin tiene 8 refuerzos `Tref(maransin,8)`
- Marmont tiene 5 tarjetas `Ttar(marmont,5)`
- Wellington tiene 23 paises `Tpais(wellington,23)`

A partir de estos predicados básicos podemos representar más propiedades del juego o partida en particular.
Obsérvese que algunos predicados requieren más información de la que a veces se tiene, analicemos por ejemplo el caso de un ataque:

- *Wellington ataca a Menelao.*
  En este caso no se tiene información acerca de los lugares de ataque. Con el predicado definido anteriormente tendriamos

  $$\exists x \exists y At(wellington, menelao, x, y)$$

  Si se prefiere evitar el uso de cuantificadores se pueden definir predicados mas simples, por ejemplo:

  - x ataca. $At1(x)$.
  - x ataca a y. $At2(x, y)$
  - x ataca a y en z $At2e(x, y, z)$
  - Atacar desde el pais x hacia el pais y. $AtD(x, y)$

- *Hay un ataque.* Usando distintos predicados de ataque:

  $$\exists x At1(x)$$
  $$\exists x \exists y At2(x, y)$$
  $$\exists x \exists y \exists z At2e(x, y, z)$$
  $$\exists x \exists y \exists z \exists w At(x, y, z, w)$$

- *Alguien ataca a Freire (Freire es atacado.)*

$$\exists x At2(x, freire)$$
$$\exists y \exists y At2e(x, freire, y)$$
$$\exists x \exists z \exists w At(x, Freire, z, w)$$

- *Aubert ataca*

$$\exists x At1(aubert)$$
$$\exists x At2(aubert, x)$$
$$\exists x \exists y At2e(aubert, x, y)$$
$$\exists y \exists z \exists w At(Aubert, y, z, w)$$

- *Hay un ataque en Argentina*

$$\exists x AtD(x, argentina)$$
$$\exists x \exists y \exists z At2e(x, y, argentina)$$
$$\exists x \exists y \exists w At(x, y, argentina, w)$$

- *Hay un ataque desde Madagascar*

$$\exists x AtD(madagascar, x)$$
$$\exists x, y, z At(x, y, z, madagascar)$$

- *Todos los paises están ocupados.*

$$\forall x \exists y Oc(y, x)$$

- *Si un jugador tiene 3 tarjetas y las intercambia entonces recibe refuerzos.*

$$\forall x(Ttar(x, 3) \land Int(x) \rightarrow \exists y Ref(x, y))$$

- *Si un jugador tiene 5 o mas tarjetas debe intercambiarlas.*

$$\forall x, y(Ttar(x, y) \land y > 5 \rightarrow Int(x))$$

- *Si un jugador tiene 8 paises o menos entonces recibe* 3 *refuerzos.*

$$\forall x \forall n(Tpais(x, n) \land n \leq 8 \rightarrow Ref(x, 3))$$

- Si un jugador ocupa n paises con $n \geq 9$ entonces recibe $(n \ mod \ 3)$ refuerzos

$$\forall x \forall n(Tpais(x, n) \land n \geq 9 \rightarrow Ref(x, n \ mod \ 3))$$

  Obsérvese en este ejemplo el uso de una función (mod) como argumento de un predicado.

- *Un pais puede ser ocupado por un único jugador.*

$$\forall x \forall y \forall z(Oc(x, z) \land Oc(y, z) \rightarrow x = y)$$

  Cuando hablamos de cosas únicas es necesario usar la igualdad.

- *Un jugador puede hacer movimiento táctico de un pais a otro sólo si posee ambos paises y estos están contiguos.*

$$\forall x \forall y \forall z \Big(Tac(x,y,z) \rightarrow Oc(x,y) \wedge Oc(x,z) \wedge Cont(y,z) \Big)$$

  Aquí introducimos el nuevo predicado binario $Cont(x,y)$.

- *Solignac fue aniquilado*

$$\exists y Ani(y, solignac)$$

- *Si Pach ataca a Mega en Siam y lo vence entonces lo aniquilará*

$$\exists y (At(pach, mega, siam, y)) \wedge Ven(pach, mega, siam) \rightarrow Ani(pach, mega)$$

- *Polain ataca desde Ontario a Menelao quien se defiende con éxito.*

$$\exists x \big(At(polain, menelao, x, Ontario) \wedge DefEn(menelao, x))$$

- *Si todos atacan a Ardilio en Rusia entonces él intercambiará tarjetas y colocará 50 tropas ahí.*

$$\forall x \exists y At(x, ardilio, rusia, y) \rightarrow Int(ardilio) \wedge Trop(50, rusia)$$

- *Si Mega ataca a Polain en Venezuela y gana entonces Polain es aniquilada, Mega moverá 7 tropas ahí y recibirá 4 tarjetas.*

$$\begin{gathered} At2e(mega, polain, venezuela) \wedge Ven(mega, polain, venezuela) \rightarrow \\ Ani(mega, polain) \wedge MovA(mega, 7, venezuela) \wedge Gt(mega, 4) \Big) \end{gathered}$$

  Aquí introducimos un nuevo predicado $MovA(x,y,z)$ para la relación "x mueve y tropas a z".

- Menelao recibe 10 refuerzos y los coloca en Gibraltar mientras que Mega se retira de Egipto moviendo 7 tropas hacia el Congo.

$$Ref(menelao, 10) \wedge Tro(10, gibraltar) \wedge Ret(mega, egipto) \wedge Mov(7, egipto, congo)$$

  Obsérvese que en la formalización no se captura el hecho de que las 10 tropas de refuerzo son las mismas 10 que se colocan en Gibraltar. Una mejor solución sería utilizar un nuevo predicado ternario $RefEn(x,y,z)$ para la relación "x recibe y refuerzos y los coloca en z".

- Todos los que ataquen a menelao serán derrotados y aniquilados por él.

$$\forall x \forall y \forall z \Big(At(x, menelao, y, z) \rightarrow Ven(menelao, x, y) \wedge Ani(menelao, x) \Big)$$

- Si Mega recibe 15 refuerzos y los coloca en Filipinas entonces Charal no lo atacará en Japón y se retirará de Kamchatka a Alaska.

$$\begin{gathered} Ref(mega, 15) \wedge Tro(15, filipinas) \rightarrow \\ \neg At2e(charal, mega, japon) \wedge RetH(charal, kamchatka, alaska) \end{gathered}$$

  Aquí introducimos el nuevo predicado $RetH(x,y,z)$ para la relación "x se retira de y hacia z"

3.1. Predicados para tipos

Obsérvese que aún cuando el mundo es heterogeneo en las especificaciones acerca del juego de Risk no hemos tenido el cuidado de distinguir entre los distintos tipos de objetos, con lo cual podemos construir fórmulas con argumentos incorrectos desde el punto de vista semántico. Por ejemplo

$$Tro(filipinas, mega), Ret(venezuela, peru), At2e(15, polain, mega)$$

Si bien éstas son fórmulas bien formadas, no tienen sentido en la situación que hemos fijado. Hemos preferido no especificar los tipos para simplificar las traducciones, sin embargo éstos pueden recuperarse introduciendo predicados unitarios por ejemplo $J(x), N(x), P(x)$ para las propiedades de ser jugador, número y predicado respectivamente. En tal caso por ejemplo la frase *alguién ataca a mega* traducida antes como $\exists x At2(x, mega)$ debe traducirse ahora como $\exists x(J(x) \wedge At2(x, mega))$ de esta manera se evitan fórmulas sin sentido como las de arriba. Por ejemplo *Cualquier entero es más pequeño que un natural* se puede traducir como $\forall x \exists y(x < y)$ sin embargo se pierde mucha información por lo que debe preferirse la traducción con tipos $\forall x\big(Int(x) \to \exists y(Nat(y) \wedge x < y)\big)$.

Para ahorrarnos algunos conectivos podemos introducir abreviaturas al estilo de los lenguajes de programación:

$$\forall x : A.\, P(x) \text{ significa } \forall x(A(x) \to P(x))$$

$$\exists x : A.\, P(x) \text{ significa } \exists x(A(x) \wedge P(x))$$

Por ejemplo la traducción anterior se expresa como $\forall x : Int.\, \exists y : Nat.(x < y)$. Veamos un último ejemplo considerando un tipo $Prog(x)$ para programas así como los predicados $F(x)$ para funcionar y $T(x)$ para terminar.

Todos los programas que funcionan terminan.

- Sin tipos: $\forall x(F(x) \to T(x))$
- Con tipos: $\forall x(P(x) \to F(x) \to T(x))$
- Con tipos abreviados: $\forall x : P.(F(x) \to T(x))$

4. Libros y Autores

Considérense las siguientes declaraciones de predicados y constantes:

$L(x, y)$	"x es más largo que y"
$E(x, y)$	"x es escrito por y"
$A(x)$	"x está escrito en alemán"
$M(x)$	"x es matemático"
$F(x)$	"x es filósofo"
$B(x)$	"x es libro"
g	"Grundlehren der Mathematik"
q	"El Quijote de la Mancha"
c	"La construcción lógica del mundo"
a	"Alicia en el país de las maravillas"
h	David Hilbert
d	Charles Dogdson.

Usaremos especificaciones con tipos para libros, matemáticos y filósofos en algunos casos:

- *"Grundlehren der Mathematik" es un libro escrito en alemán.*

$$B(g) \wedge A(g).$$

- *Hay un libro más largo que "El Quijote de la Mancha".*

$$\exists x : B.\, L(x, q)$$

- *"La construcción lógica del mundo" es un libro escrito por un filósofo y no es mas largo que "Alicia en el país de las maravillas"*

$$(\exists x : F.\, E(c, x)) \wedge \neg L(c, a)$$

- *"Alicia en el país de las maravillas" es un libro escrito por un matemático.*

$$\exists x : M.(B(a) \wedge E(a, x)) \text{ equivalente a } \; B(a) \wedge \exists x : M.E(a, x)$$

- *Hay un libro escrito en alemán por un matemático que es más largo que "Grundlehren der Mathematik".*

$$\exists y : B.\exists x : M.\, (A(y) \wedge E(y, x) \wedge L(y, g))$$

- *Si "Alicia en el país de las maravillas" es un libro escrito por un matemático entonces es un libro escrito por un filósofo.*

$$\exists x : M.\, (B(a) \wedge E(a, x)) \rightarrow \exists y : F\, (B(a) \wedge E(a, y))$$

- *No hay un libro escrito en alemán por un filósofo.*

$$\neg \exists x : B.\, (A(x) \wedge \exists y : F.\, E(x, y))$$

- *No es el caso que cualquier libro es escrito por un filósofo.*

$$\neg\forall x\exists y(F(y) \land E(x,y))$$

- *Para cualquier libro escrito por Dogdson, "Grundlehren der Mathematik" es un libro más largo escrito por Hilbert.*

$$\forall x : B.\,(E(x,d) \to E(g,h) \land L(g,x))$$

- *Hay alguien que es filósofo y matemático y ha escrito un libro en alemán.*

$$\exists x(F(x) \land M(x) \land \exists y : B(E(y,x) \land A(y)))$$

5. El Mundo de Cubos

Tenemos seis cubos de color amarillo, azul o verde. Un cubo puede estar sobre otro o en el piso. Considerense los predicados $S(x,y)$ representando qu x está sobre y, $P(x)$ para x está en el piso, $A(x), Az(x), V(x)$ representan los colores, $L(x)$ significa que x está libre, es decir que ningún cubo está sobre x. Obsérvese que en el mundo hay cubos únicamente por lo que un predicado para "ser cubo" es redundante.
Simbolizar los siguiente:

- Hay un cubo azul en el piso con un cubo amarillo sobre él y un cubo verde sobre el amarillo.

$$\exists x\exists y\exists w\Big(Az(x) \land A(y) \land V(w) \land P(x) \land S(y,x) \land S(w,y)\Big)$$

- Ningún cubo amarillo está libre.

$$\forall x(A(x) \to \neg L(x))$$

- Hay un cubo azul libre y un cubo verde libre.

$$\exists x\exists y\big(Az(x) \land L(x) \land V(y) \land L(y)\big)$$

- Cualquier cubo amarillo tiene un cubo sobre él.

$$\forall x\big(A(x) \to \exists y S(y,x)\big)$$

- No todos los cubos azules están libres.

$$\exists x(Az(x) \land \neg L(x))$$

- Cualquier cubo verde está libre.

$$\forall x\big(V(x) \to L(x)\big)$$

- Todos los cubos en el piso son azules.

$$\forall x\big(P(x) \to Az(x)\big)$$

- Cualquier cubo que esté sobre un cubo amarillo es verde o azul.

$$\forall x\Big(\exists y(A(y) \wedge S(x,y)) \rightarrow V(x) \vee Az(x)\Big)$$

- Hay un cubo verde sobre un cubo verde.

$$\exists x \exists y\big(V(x) \wedge V(y) \wedge S(x,y)\big)$$

- Hay un cubo amarillo libre en el piso.

$$\exists x\big(A(x) \wedge L(x) \wedge P(x)\big)$$

- Ningún cubo está en el piso.

$$\forall x\ \neg P(x)$$

- Hay un cubo amarillo que está sobre uno azul y hay un cubo azul sobre él.

$$\exists x \exists y\Big(A(x) \wedge Az(y) \wedge S(x,y) \wedge \exists w(Az(w) \wedge S(w,x))\Big)$$

- Todos los cubos están sobre algo.

$$\forall x \exists y S(x,y)$$

6. Triangulos, Círculos, Cuadrados

El mundo consta de una cuadricula de cualquier tamaño. Los objetos son circulos, cuadrados o triangulos y pueden ser pequeños, medianos o grandes. También se tienen las relaciones dadas por la posición, sur, norte, este, oeste y las relaciones dadas por la misma columna y el mismo renglón. Los predicados para las figuras son: $T(x), C(x), S(x)$ para triangulo, círculo y cuadrado respectivamente; para tamaño tenemos $P(x), M(x), G(x)$ para pequeño, mediano y grande. Para la posición tenemos $Su(x,y), N(x,y), E(x,y), O(x,y)$ por ejemplo $N(x,y)$ significa x está al norte de y. Finalmente tenemos $Co(x,y), R(x,y)$ para x está en la misma columna o renglón que y.

- Hay circulos medianos y cuadrados grandes

$$\exists x(C(x) \wedge M(x)) \wedge \exists y(S(y) \wedge G(y))$$

- No hay cuadrados pequeños.

$$\forall x(S(x) \rightarrow \neg P(x))$$

- Hay un triangulo al sur de todos los circulos

$$\exists x\Big(T(x) \wedge \forall y(C(y) \rightarrow Su(x,y))\Big)$$

- No hay dos triangulos en el mismo renglón

$$\neg \exists y \exists x\big(T(x) \wedge T(y) \wedge R(x,y)\big)$$

- Hay un círculo tal que todos los círculos al oeste de él son grandes.

$$\exists x\Big(C(x) \land \forall y(C(y) \land O(y,x) \to G(y))\Big)$$

- Ningún cuadrado está al norte de un círculo grande.

$$\neg\exists x(S(x) \land \exists y\Big(C(y) \land G(y) \land N(x,y))\Big)$$

- Todos los circulos medianos están al oeste de un mismo triangulo grande

$$\exists x(T(x) \land G(x) \land \forall y(C(y) \land M(y) \to O(y,x)))$$

- Todos los cuadrados pequeños están al sur de cualquier tríangulo.

$$\forall x\Big(S(x) \land P(x) \to \forall y(T(y) \to Su(x,y))\Big)$$

- Si dos cuadrados están en el mismo renglón entonces cualquier triangulo al sur de ambos es mediano.

$$\forall x\forall y\Big(S(x) \land S(y) \land R(x,y) \to \forall z\big(T(z) \land Su(z,x) \land Su(z,y) \to M(z)\big)\Big)$$

- No hay dos triangulos medianos en la misma columna y si un triangulo es grande entonces hay un circulo pequeño al este de él.

$$\neg\exists x\exists y\big(T(x) \land T(y) \land Co(x,y)\big) \land \forall z\Big(T(z) \land G(z) \to \exists w(C(w) \land P(w) \land E(w,z))\Big)$$

7. Ejercicios

1. Realizar la especificación formal acerca de las siguientes propiedades de la función que verifica la pertenencia de un elemento a una lista dada. En cada caso dar dos especificaciones, una funcional y otra relacional. Definir primero el lenguaje a utilizar.

   a) Ningún elemento pertenece a la lista vacía.

   b) Si cierto elemento es la cabeza de una lista, entonces dicho elemento pertenece a esa lista.

   c) La cabeza de una lista pertenece a dicha lista.

   d) Si un elemento pertenece a una lista entonces o es la cabeza de esa lista o pertenece a la cola.

   e) Si un elemento pertenece a una lista entonces pertenece a cualquier otra lista cuya cola es la lista anterior.

   f) Si un elemento pertenece a la concatenación de dos listas entonces pertenece a alguna de estas dos listas.

   g) Si un elemento pertenece a una lista entonces dicha lista es la concatenación de otras dos donde la segunda tiene como cabeza a dicho elemento.

   h) Que un elemento pertenezca a la reversa de una lista equivale a que pertenezca a la lista.

2. Realizar la especificación formal de las siguientes propiedades de las funciones reversa de una lista y concatenación de dos listas. En cada caso dar dos especificaciones, una funcional y otra relacional. Definir primero el lenguaje a utilizar.

   a) La operación de concatenación es asociativa.

   b) Si la concatenación de dos listas es vacía entonces ambas listas son vacías.

   c) Si la concatenación de dos listas es una lista unitaria entonces una de dichas listas es vacía y la otra es la misma lista unitaria.

   d) La reversa de la concatenación de dos listas es la concatenación de la reversa de la segunda lista con la reversa de la primera lista.

   e) La reversa de la reversa de una lista es la misma lista.

   f) Si la reversa de una lista es la misma lista entonces dicha lista es vacía o unitaria.

3. En el lenguaje de programación LISP una expresión simbólica o S-expresión es alguna de las siguientes

   - Un símbolo.
   - Una S-lista, donde una S-lista es una sucesión, tal vez vacía, de S-expresiones, encerrada entre paréntesis.

   Por ejemplo, las siguientes son cinco S-expresiones:

   ```
   e,  (), (e z), (b (a m)), (f  (a  ()  )  v  (i  o))
   ```

   Dé una especificación formal de las S-expresiones.

4. Realizar la especificación formal acerca del tipo abstracto de datos pila considerando homogeneidad, es decir, suponiendo que todos los elementos en una pila son del mismo tipo digamos A. En cada caso dar dos especificaciones, una funcional y otra relacional. Definir primero el lenguaje a utilizar.

   a) Definición del tipo de datos pila:
      1) La vacía es una pila (de elementos de A).
      2) El resultado de agregar un elemento de A en el tope de una pila es una pila.
      3) Son todos.

   b) Si una pila no es vacía entonces el elemento en el tope es un elemento de A

   c) Si una pila no es vacía entonces la pila que resulta al eliminar el elemento en el tope es una pila de elementos de A.

   d) Si una pila no es vacía entonces la pila que resulta al agregar el tope de la pila dada a la pila obtenida al eliminar el tope de la pila dada es la pila dada.

   e) La pila vacía no tiene elementos.

   f) La pila que resulta de agregar un elemento a una pila dada tiene un elemento más que la pila dada.

   g) Si la pila resultante de eliminar el tope de una pila dada es vacía entonces la pila dada tiene sólo un elemento.

5. Realizar una especificación formal de alguna estructura o tema de interés particular. Definir primero el lenguaje a utilizar. Algunas sugerencias son:

   a) Sistemas numéricos y sus propiedades:

      1) Enteros, racionales, etc..
      2) Binarios, ternarios, etc..
      3) Propiedades de conversión entre sistemas.

   b) Estructuras de datos y sus propiedades:

      1) Listas con el constructor `snoc`
      2) Listas heterogéneas con elementos de dos tipos dados.
      3) Árboles binarios
      4) Colas, tablas, arreglos, etc.
      5) Árboles rojinegros, árboles AVL, etc..
      6) Conjuntos.

   c) Micromundos:

      1) Extensión del micromundo de figuras a tres dimensiones.
      2) Mundo de bloques agregando un brazo de robot (reglas para que el brazo mueva los bloques (planeación en inteligencia artificial))
      3) Sistema de correo electrónico.
      4) Sistema de funcionamiento de un elevador.

   d) Reglas de algún juego de su interés (juegos de mesa, juegos de rol, ajedrez, sudoku, etc)