INTRODUCCIÓN AL CURSO DE LÓGICA

JACINTO DÁVILA

Abstract. El objetivo de este material es definir las reglas y conocer el lenguaje de programación que usaremos en el curso introductorio a la lógica matemática. Como parte de esta preparación tratamos de responder la pregunta ¿Para qué estudiamos lógica?.

1. ¿Para qué estudiamos lógica?

Para aprender a comunicarnos con claridad y precisión. Lógica es un tema de estudio muy antiguo. Sus orígenes se confunden con los de nuestra tradición cultural occidental, con hechos acaecidos hace unos 24 siglos. Desde entonces muchas personas han estudiado lógica. Considerando los variado que ha sido el panorama intelectual occidental durante esos siglos es fácil entender que han habido muchas razones para estudiar lógica. Nuestro apego a la corta respuesta en la primera oración de este párrafo se debe a la creencia de que más que una herramienta para alcanzar la verdad, más que una plataforma para gimnasia mental, más que un método para argumentar y convencer a otros, lógica es un lenguaje. Un lenguaje que, bien usado, puede servir a una persona (un agente) para definir sus objetivos y el cómo alcanzarlos y, al hablar o escribir, decir exáctamente lo que tiene que decir para que su interlocutor entienda el mensaje.

Algunos se preguntarán qué tiene esto que ver con este curso de lógica y matemática para computación o simulación. La razón para seguir este curso es conocer un lenguaje que puede usarse para especificar y programar sistemas computacionales. Hay, sin embargo, mucho en común entre los proyectos para programar computadores y los procesos de comunicación entre personas. Al computador debemos decirle exáctamente que debe hacer. Hay que ser preciso. Pero precisión es algunas veces contrario a claridad. Si somos demasiado precisos (como hay que ser, por ejemplo, al programar un computador en lenguaje de máquina) podemos terminar "diciendo cosas" que serán sumamente difíciles de entender para otros programadores. Si, por el contrario, somos demasiado generales (quizás porque estamos procurando ser claros) y obviamos detalles importantes, el computador no podrá ejecutar el programa.

El compromiso correcto entre claridad y precisión es fundamental en la programación de computadores. Ha habido un incremento enorme en la productividad de los programadores con la invención de lenguajes de alto nivel, que permiten programar con más claridad. Sin embargo, la proliferación de lenguajes de programación de alto nivel parece obedecer a que muchos de esos lenguajes no son suficientemente precisos para decir lo que los programadores desean decir en cierto momento.

Algunos creemos que claridad, precisión y la tensión entre ellas son también de suma importancia en la comunicación entre humanos. Mucho más significativo de

Key words and phrases. lógica, PROLOG.

lo que alcanzamos a distinguir en el quehacer cotidiano de nuestro entorno nacional. ¿Cuantas veces nos hemos desesperado porque no entendemos (o ni siquiera sabemos) las instrucciones para realizar cierto trámite administrativo?. En esas situaciones, es típico que ni siquiera las personas que atienden el servicio conozcan las *reglas*. Esto sin considerar que elementales *reglas* de cortesía son también ignoradas (por ejemplo, en ciertas instituciones la regla que dice "**si** Ud. ve a una persona desesperada **entonces** ofrézcale su ayuda" probablemente nunca ha sido utilizada).

Observe que hablamos en el párrafo anterior de **reglas** y normas **para guiar la acción de la gente**. En este curso estaremos aprendiendo a escribir **reglas para guiar a los computadores**. Si las personan se atreven a "jugar por las reglas" sin el temor usual de que se les convierta en robots, la programación de personas y la programación de computadores se convierten actividades muy parecidas. Programar personas es mucho más fácil que programar computadores porque las primeras son (mucho más) inteligentes. La clave de esa inteligencia está en saber cuando usar una regla y cuando no. La intención no es tener procedimientos con reglas rígidas que dicen todo lo que hay que hacer. No. La intención es tener un conjunto de patrones o guías de acción que el agente (humano ó máquina) aplica inteligentemente en la situación en la que se encuentra.

La programación de personas, por muy odiosa que pueda resultar la idea a primera vista, tiene efectos dramáticos en la organización del entorno social. La idea no es nueva. Los sistemas legales son enormes programas para guiar las acciones de los humanos que interactúan con otros humanos. Lamentablemente, a pesar del esfuerzo de muchos y muy buenos juristas, nuestros códigos legales no son siempre fáciles de leer "y ejecutar" (sin mencionar que muchos no llegan a ser conocidos debido a su tamaño).

En ese contexto, los desarrollos de la lógica matemática y de la programación lógica tienen mucho que ofrecer. En particular, pueden ofrecer un lenguaje con una enorme tradición de cuidado por lo que se dice en él. Un lenguaje que, a cambio de cierta disciplina al escribirlo, ofrece tanta claridad y precisión como puedan ser necesarias incluso para programar computadores. Y además, un lenguaje que puede ser entendido y usado por humanos.

En la sección siguiente comenzamos a usar las ideas que acabamos exponer. Presentamos un conjunto de reglas que pretender guiar nuestra interacción a lo largo de este (y otros cursos). Pretendemos **programarnos** para ser más eficientes, efectivos y sobretodo cooperativos mientras interactuamos para los efectos del curso. Debemos insistir que estos son guías flexibles de acción. No pretenden decir TODO lo que hay que hacer. Más aún, se puede ser eficiente, efectivo y cooperativo sin usar esas reglas. Es decir, no llevan ninguna intención de ser dogmas irrefutables del cómo actuar. En lugar de eso, las reglas deben ser juzgadas por su utilidad. Si en algún momento, Uds. creen que las reglas NO SIRVEN, simplemente deséchenlas y déjense llevar por su **sentido común**. (Por cierto, sería muy útil si Uds. reportaran cuando las reglas no sirven. Así nos ayudarían a *depurar* el programa).

2. Las reglas de este juego

2.1. **Los retos de este curso.** Este curso es uno de los primeros que se ofrecen como parte de proyecto de educación interactiva a distancia de la Universidad de Los Andes. Fue cursado a distancia, con interacción semanal con el instructor

vía Internet, por una veintena de personas en su primera edición. Es además un curso obligatorio para los programas de maestría y especialización en computación y maestría en modelado y simulación, que se siguen localmente en la U.L.A.

Este curso, por tanto, se dirige a un grupo amplio de profesionales con planes de desarrollo individual probablemente muy diferentes. Pero, precisamente, no puede haber mejor prueba para un curso acerca de un lenguaje que pretende ser universal, que exponerlo a un grupo así de amplio. En este caso, además, es particularmente motivante que ese grupo amplio está conformado por personas interesadas en tecnologías, quienes han aceptado el reto de *desafiar la complejidad* y de convertirse en *agentes del cambio tecnológico*.

2.2. **El contexto del curso.** Este curso constituye un importante desafío para quienes participamos en él y no sólo por la condición de dictado a distancia. En nuestro entorno social, los estudios de lógica no están muy arraigados. Sin poder discutir las razones de ese desarraigo, tiene uno que decir que, simplemente, no hay tradición lingüística y mucho menos lógica. Se puede apreciar el fenómeno al revisar cómo escribimos y como hablamos en público, con un enorme desdén y descuido por lo que se decimos. Ya no confiamos en la palabra (hablada o escrita) y en consecuencia, creemos que cualquier cosa puede ser un mensaje (ilustraremos estas afirmaciones con algunos ejemplos).

En este contexto, tenemos todos que hacer un enorme esfuerzo para completar un curso como este. Los instructores haremos nuestra parte. Pero queremos pedirle a los estudiantes que, por favor, consideren y procuren seguir los siguientes principios:

2.3. **El principio básico en la comunicación.**

La mayor barrera para la comunicación es el suponer.

Si Ud. dice algo y su interlocutor no lo entiende, muy probablemente se debe a que Ud. supuso que su interlocutor sabía algo que no sabe. Hay que ser preciso (pero sin olvidar que se puede uno perder en detalles irrelevantes). El principio general es: *considerar al interlocutor. Debemos tratar de precisar lo que el interlocutor sabe y lo que necesita saber para entender el mensaje.*

Por otro lado, en este mundo moderno, todos tenemos una enorme presión para absorber información. Todos tenemos demasiados mensajes que procesar. Así que quizás la mayor consideración para con el interlocutor sea: *ser breve.*

Para ser breve con éxito, debe uno ser claro. Así claridad puede afrentarse con precisión, como comentábamos antes. Es difícil convertirse en maestro equilibrista de la precisión y la claridad (sobretodo cuando hay relativamente pocos practicantes alrededor). Quizás un poco de lógica pueda ayudar.

2.4. **La lógica de la conversación.** En [2], Keith Devlin se refiere al trabajo del filósofo y lógico británico Paul Grice. En 1967, Grice[3] formuló una serie de *máximas* para guiar la conversación, que se pueden resumir así:

El principio básico (al conversar o escribir) es:

Al comunicarse, coopere.

Según Grice, ese principio de cooperación se puede abordar desde cuatro puntos de vista que él denominó: cantidad, calidad, relación y manera.

Cantidad se refiere a la cantidad de información que el hablante (escritor) debe dar. Las máximas son:

1.- *Haga que su contribución* [al comunicarse] *sea tan informativa como sea necesario.*

2.- *No permita que su contribución sea más informativa de lo necesario.*

Acerca de **calidad,** las máximas son**:**

1.- *Trate de decir lo que es cierto.*

2.- *No diga lo que Ud. cree que es falso.*

3.- *No afirme aquello sobre lo que no tiene evidencia adecuada.*

Acerca de **relación**, la máxima es:

1.- *Sea relevante.* (Detrás de esta máxima hay todo un universo por explorar).

Acerca de **manera**, las máximas son las siguientes. Noten que es difícil ser preciso con estas reglas. En el caso de manera, en particular, las máximas de Grice parecen redundantes (¿Ud. que opina?):

1.- *Sea claro.*

2.- *Evite las expresiones obscuras.*

3.- *Evite la ambigüedad.*

4.- *Sea breve.*

5.- *Sea ordenado.*

2.5. **Los primeros dos principios para escribir con claridad.** Debemos insistir que es difícil legislar acerca de comunicarse con claridad. La audiencia tiende a creer 1) que al dictar reglas, pretende uno que las personas se comporten como robots, sin atender a la substancia de lo que dicen y 2) que se está matando la posibilidad de ser creativo con en lenguaje. A la primera parte de la objeción no necesitamos responder porque, como dijimos antes, creemos que nos hemos vuelto tan descuidados con el lenguaje que poco atendemos a su substancia. A la segunda parte sólo podemos responder parcialmente. Se requiere una discusión extensa para rebatirla. Pero puede uno decir que los mejores deportistas normalmente se apegan a las reglas de cada deporte y no por ello dejan de lograr prodigios.

El profesor Joseph Williams, de la Universidad de Chicago, se ha atrevido a sugerir algunas reglas que podrían ayudar a mejorar el estilo de la escritura (hacerlo más claro). Sus libros [4] van dirigidos a los angloparlantes, pero las reglas son tan lógicas, que se traducen fácilmente al español. El dice: "Los lectores probablemente sentirán que están leyendo prosa que es clara y directa cuando:

(1) Los sujetos en las oraciones se refieren a los agentes [de la trama o situación que se describe].

y

(2) Los verbos que acompañan a los sujetos se refieren a las acciones en las que los agentes se involucran" (.ibid, pág. 21).

Como dijimos, estos son sólo guías generales. No pueden ser reglas fijas ó terminantes. No siempre los mejores sujetos son los agentes ó los verbos se refieren a acciones. Consideren la siguiente oración:

Este objeto me pertenece.

Se trata de una afirmación clara y tajante. Sin embargo, ni el verbo se refiere a un acción (a menos que admitamos *pertenecer* como un tipo de acción), ni el agente (la persona detrás de "me") es el sujeto. Alguien podría alegar que una forma más apegada a las reglas anteriores es:

Yo soy el dueño de este objeto.

Pero esta es una oración más larga que, sin embargo, no logra eludir el problema de que el verbo no se refiere a una acción (a menos que admitamos *ser* como un tipo de acción). Todavía uno puede apelar a la más compacta:

Yo poseo este objeto

Pero esta oración es menos clara que las otras dos, debido a la multiplicidad de interpretaciones del verbo *poseer*.

Así que debemos seguir confiando en nuestra intuición (inteligencia?) para usar las reglas. Pero si consideramos que lo importante es ayudar a nuestros lectores, quizás las reglas anteriores nos sirvan en un momento de duda. Podemos también agregar otra guías de acción que también pueden ser útiles, por lo menos para los efectos del curso.

En mucho de nuestra formación como escritores en español, se insiste en la necesidad de:

(1) Escribir en forma *impersonal* ("se indica...", en lugar de, "indicamos..". "Se cree..", en lugar de "creen..", y otras formas similares).

(2) Usar la forma *pasiva*, en lugar de la activa (precisamente para evitar la primera persona). Por ejemplo, decimos "Se ha considerado el estudio de..", en lugar de, "Consideramos el estudio de..".

Al parecer, una buena parte de los académicos creen que esta forma de escribir es más elegante, formal y hasta más clara y estos atributos la hacen la ideal para reportar el trabajo científico.

No podemos argumentar en contra del alegato de elegancia (entre gustos y colores..), pero tenemos un poderoso argumento en contra del de claridad: *En la forma impersonal-pasiva [de redacción] los agentes se vuelven invisibles.* El lector no puede establecer con *claridad* **quienes son los responsables de las acciones**.

En cuanto a formalidad, la forma impersonal-pasiva no lo es más que la personal-activa. Ambas son igualmente traducibles a un lenguaje formal, con la diferencia que la última tendría que incluir una declaración explícita de los agentes. Esto es un poco más difícil de hacer, pero no es imposible. Quizás el alegato de formalidad sea un residuo de aquella creencia positivista en una verdad objetiva independiente de los agentes y alcanzable a través de la lógica formal (que apenas recientemente ha abordado el problema de incluir a los agentes).

En cualquier en caso, en este curso vamos a favorecer (y a insistir) en el uso de la forma personal-activa, simplemente por considerarla más clara y precisa (No obstante, el uso del "nosotros" en lugar del "yo" es aconsejable, para evitar ofender al lector con demasiadas ínfulas egocéntricas).

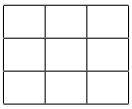
2.6. **Notación y sintaxis.** Como se puede apreciar, las normas que estamos sugiriendo no se refieren únicamente a la "substancia" de lo que se dice, sino también a la forma de decirlo. Esta es una observación muy importante: Algunos creen que la substancia no se puede mostrar sino a través de la forma y por ellos debemos ser muy cuidadosos con las estructuras, palabras y hasta las letras que usamos al escribir.

Si alguien nos recomienda que seamos "superficialmente corteses" quizás creamos que nos aconseja ser cínicos o hipócritas y que, por tanto, debemos rechazar su consejo. Sin embargo, si la cortesía debe estar en algún sitio es en la superficie, donde puede ser parte y cumplir su cometido en la interacción entre las personas. Algo parecido ocurre con la claridad al escribir: tiene que verse y sentirse para ser

efectiva. De nada sirve que el escritor use un sofisticado (y quizás elitesco) léxico en su mensaje. Si olvida emplear la *notación* adecuada para su auditorio, el mensaje no llegará.

Quienes programamos computadores hemos aprendido lo importante que es la estructura (diagramación, indentación, márgenes y otros detalles) de un texto. Dependemos de esas estructuras para poder leer con facilidad. Si esta afirmación les parece exagerada consideren el siguiente ejercicio (similar a uno descrito por Dennett en [1], pág. 296):

SOLO EN SUS MENTES (no escriban, por favor), consideren las palabras HPU AHL ZDA e imagínelas escritas, letra por letra, a lo largo de las **columnas** de una tabla 3x3 como la siguiente:



¿Pueden leer las **filas** del arreglo?. Normalmente, quienes hacen este ejercicio deben esforzarse mucho para extraer el mensaje. Son sólo 3 palabras de 3 letras cada una, pero nos cuesta mucho leerlas si no les vemos escritas en la forma "correcta" (a pesar de que somos capaces de ordenarlas con el "ojo de la mente"). Este ejercicio pretende ilustrar la importancia de la estructura al escribir. Márgenes, espacios en blanco y tamaño de la letras, tienen todos un enorme efecto sobre la capacidad para entender de nuestros lectores.

Cerramos esta sección, repitiendo la observación de que las anteriores son guías generales. No son reglas definitivas, terminales e inapelables del cómo comunicarse. Son reglas para ayudarnos a comunicarnos mejor y respecto a ese criterio debe ser evaluadas. Si no sirven, deséchenlas.

3. Introducción al PROLOG

Habiendo definido las reglas del juego, ahora vamos a conocer la arena de juego. En este curso los estudiantes emplearán un lenguaje de programación de computadores: PROLOG.

3.1. **¿Qué es el PROLOG?.** Es un lenguaje para programación lógica. Esto significa que la sintaxis del lenguaje corresponde a algún lenguaje lógico (en este caso, a un subconjunto de la lógica de primer orden). Además, el computador procesando (interpretando) código PROLOG está realizando una forma de razonamiento lógico. En particular, computar con PROLOG es equivalente a deducir, con lógica de primer orden, ciertas oraciones, siguiendo cierta regla de inferencia lógica que estudiaremos en el curso.

Antes de embarcarnos en la exploración del substrato teórico del PROLOG y de toda la programación lógica, conozcamos algunos detalles operacionales de toda plataforma de desarrollo para PROLOG (que normalmente incluye un interpretador del lenguaje y algunas herramientas para manipular los archivos que contienen los textos PROLOG).

3.2. **¿Como se escribe, se guarda y se consulta un programa en PROLOG?.** Para PROLOG se ha definido un estándar ISO que norma toda su sintaxis, incluyendo los comandos para manipular archivos y otras interacciones con el sistema operativo. Por tanto un programa PROLOG es completamente transportable entre distintas plataformas de computación.

Para escribir un programa PROLOG, simplemente emplee un editor de texto cualquier (o use el que provee el ambiente de desarrollo). El texto de un programa PROLOG se compone de **cláusulas**, oraciones lógicas que termina con un punto y que se muestran más adelante (y que serán cuidadosamente estudiadas en el curso).

Guarde el texto en un archivo ASCII con la extensión que le indique la plataforma particular. Algunas extensiones comunes son .pl , .pro , .p .

Como dijimos antes, PROLOG posee su propio interpretador que convierte al computador en un razonador lógico (el código PROLOG se puede compilar, pero no lo haremos en este curso. Discutiremos al respecto en las listas de correo).

Al invocar el interpretador PROLOG, aparecerá una ventana similar a aquellas usadas para interactuar con los sistemas operativos con lenguajes de comandos. El sistema espera por comandos (que en el caso de PROLOG son más bien preguntas, como veremos) con este mensaje de espera (prompt): **?**.

Para abandonar el sistema se usa la instrucción **halt**, así:

```
? halt.
```

y "enter".

Para procesar programas PROLOG, lo primero que uno debe hacer es cargar el archivo que el sistema interpretará y con el cual responderá nuestras preguntas. En la jerga PROLOG esto se conoce como "consultar" y se invoca del siguiente forma:

```
? consult( archivo ).
```

ó

```
? [archivo].
```

y "enter", donde archivo es el nombre del archivo que contiene el texto PROLOG a interpretar. Noten, por favor, el punto al final de la pregunta. Siempre debe haber uno.

Al invocar la consulta, el interpretador chequeará la sintaxis del texto PROLOG y reportará errores y mensajes de precaución, si es necesario. Si los primeros ocurren (mensajes de error) el programador tendrá que revisar y modificar el texto PROLOG. De otra forma no podrá ejecutarlo.

Cuando ocurren mensajes de precaución (`Warnings`) el código puede ser ejecutado, pero probablemente ocurran cosas que el programador no había previsto. Así que es aconsejable también revisar el código como indiquen los "warnings".

Si todo va bien con la "carga" del archivo, el comando de consulta culminará cuando el sistema responda:

```
yes.
```

Aprenderemos a apreciar esta respuesta, que indica que PROLOG ha podido culminar satisfactoriamente la tarea ó, lo que es lo mismo, ha podido responder afirmativamente la pregunta que le hicimos. Consecuentemente, cuando PROLOG no logra hacer lo que se le pide, la repuesta es:

```
no.
```

3.3. **Primeros ejercicios con PROLOG.** Para ilustrar la programación PROLOG, construiremos en esta sesión un pequeño programa lógico y realizaremos algunas consultas. El primer programa lógico registra el árbol genealógico de una familia y pretende mostrarles como PROLOG puede servir como un manejador de bases de datos.

3.3.1. *El ejercicio del árbol genealógico.* Consideren el siguiente código PROLOG:

```
hij_(abraham, teraj).
```

```
hij_(najor, teraj).
hij_(aram, teraj).
% hij_(sara, teraj).
hij_(lot, aram).
hij_(hija_mayor, lot).
hij_(hija_menor, lot).
hij_(moab, hija_mayor).
hij_(moab, lot).
hij_(ben_ammi, hija_menor).
hij_(ben_ammi, lot).
hij_(ismael, abraham).
hij_(ismael, agar).
hij_(isaac, abraham).
hij_(isaac, sara).
hij_(batuel, najor).
hij_(batuel, melca).
hij_(rebeca, batuel).
hij_(esau, isaac).
hij_(esau, rebeca).
hij_(jacob, isaac).
hij_(jacob, rebeca).
```

Este texto contiene una definición de la relación **hijo o hija de**. Así, `hij_(X, Y)` debe leerse como "X es hijo o hija (o descendiente inmediato) de Y". Observe que esto es similar a una tabla de una base de datos. Cada átomo hij_, corresponde a un registro de la base de datos. Uno puede colocar toda esa información y la siguiente en un archivo:

```
masculino(teraj).
masculino(abraham).
masculino(najor).
masculino(aram).
masculino(lot).
masculino(moab).
masculino(ben_ammi).
masculino(ismael).
masculino(isaac).
masculino(batuel).
masculino(esau).
masculino(jacob).


femenino(sara).
femenino(agar).
femenino(hija_mayor).
femenino(hija_menor).
femenino(melca).
femenino(rebeca).
```

Y entonces, uno puede interactuar y extraer información de un archivo con el texto anterior.

Por ejemplo, después de cargar el archivo, puede uno preguntar al sistema si Rebeca es hija de Batuel así:

```
?  hij_(rebeca, batuel).
```

Más interesante todavía es poder preguntar si acaso cierto personaje tiene un hijo o hija:

```
?  hij_(X, abraham).
```

La "X" es una variable. Literalmente, estamos preguntando si existe algún X tal que la relación hij_ se cumple entre ese X y `abraham`. Las variables en PROLOG se indican con cadenas de caracteres cuya primer letra es una mayúscula. Esta es la razón por la cual todos los nombres están en minúscula y no comienzan con Mayúscula como indica el buen español.

Las respuestas de PROLOG ante esos dos tipos de preguntas son similares. En ambas aparece el "yes" de confirmación. Sin embargo, en el segundo caso, la respuesta va acompañada de una indicación de un valor que permite que la respuesta se responda afirmativamente.

Nada de lo anterior supera a las facilidades que ofrece un manejador cualquiera de base de datos. La diferencia comienza a surgir cuando uno usa las relaciones básicas anteriores, para definir nuevas relaciones por medio de reglas que en lo sucesivo llamaremos cláusulas:

```
hija( X, Y ) :- hij_(X, Y), femenino( X ).
```

Esta es la forma PROLOG de escribir la cláusula: X es hija de Y **si** X es descendiente inmediata de Y **y** X es del sexo femenino. Por supuesto, muchas otras relaciones se pueden definir:

```
hijo(X, Y) :- hij_(X, Y), masculino( X ).
padre( X , Y ) :- hij_( Y, X ), masculino( X ).
madre( X, Y ) :- hij_( Y, X ), femenino( X ).
abuela( X, Y ) :- madre( X, Z ), hij_( Y, Z ).
abuelo( X, Y ) :- padre( X, Z ), hij_( Y, Z ).
descendiente( X, Y ) :- hij_( X, Y ).
descendiente( X, Y ) :- descendiente( X, Z ), descendiente(Z, Y).
```

La última cláusula introduce una facilidad muy importante en un lenguaje de programación: recursión. Para definir la noción de descendencia, se apela a la misma definición de descendencia. Computacionalmente esto sólo es factible cuando, en la definición, se cuenta también con una cláusula como la penúltima que nos permite **terminar** la deducción. Hablaremos mucho de la recursión a lo largo de este curso.

Para completar el ejercicio, pedimos a los estudiantes que construyan las definiciones adecuadas para responder a las preguntas:

```
?  tío(lot, Y). % Quien es tío de Lot.
?  tia(Y, sara)  % Quien es sobrino o sobrina de Sara.
```

```
?  prim_(X, Y) % Quien es primo o prima de quien
```

Un recurso del sistema PROLOG que puede ser muy útil para responder estas y otras preguntas es el comando `findall`, que explicamos a continuación:

Una pregunta en PROLOG puede tener más de una respuesta. Si uno quiere **encontrar TODAS las respuestas** a una pregunta puede *hacerla a través* del comando `findall`, cuya sintaxis es:

`findall(` *`Variable ó estructura, PREGUNTA, lista de respuestas`* `).`

Por ejemplo, para preguntar quienes son los hijos de Abraham, puede uno decirle a PROLOG:

```
?  findall( X, hijos( X, abraham ), L ).
```

y observar el "valor" asignado a L.

Si el tiempo no apremia, prueben también la pregunta:

```
?  findall( X, descendiente( X, abraham), L ).
```

y observen que ocurre. ¿Tendrá que ver con la promesa de Dios a Abraham?.

En este ejercicio, hemos usado PROLOG como el lenguaje del manejador de una base de datos *deductiva*. En el curso, aprenderemos a usar PROLOG, como un lenguaje de programación de propósito general. Vamos por él.

References

1. Daniel C. Dennett, *Consciousness explained*, Penguin Books, 1991.
2. Keith Devlin, *Goodbye, descartes. the end of logic and the search for a new cosmology of the mind.*, John Wiley and Sons, Inc., 1997.
3. H. P. Grice, *Logic and conversation*, Syntax and Semantics 3: Speech Acts (P. Cole and J. Morgan, eds.), Academic Press, 1975.
4. Joseph M. Williams, *Style. toward clarity and grace*, The University of Chicago Press, 1990.

Centro de Investigación y Proyectos en Simulación y Modelos, Universidad de Los Andes, Venezuela

E-mail address: jacinto@ula.ve