El uso de Prolog en el aula: de lógica a inteligencia artificial[1]

The use of Prolog in the classroom: logic to artificial intelligence

Sonia Milena Gil Suárez*

Diana Isabel Martínez Buitrago**

Fecha de recepción: 24 de mayo de 2013
Fecha de aceptación: 15 de junio de 2013

Resumen

Con base en nuestra experiencia docente, se evidencia la necesidad de darle a Prolog más relevancia en el curso inicial de Lógica Matemática, mostrando cómo a través del uso de Prolog es posible desarrollar muchas ideas en el alumno, entre las que se destacan el entendimiento de ejercicios netamente procedimentales y la especificación para resolver situaciones en forma declarativa. Este proceso de enseñanza puede generarse construyendo una base sólida de conocimiento tanto en lógica proposicional, como en lógica de primer orden. Es por esto que se busca exponer en este artículo algunas de las alternativas para emplear Prolog (y otros lenguajes de programación declarativa como Lisp) con fines educativos, teniendo en cuenta que éste es considerado, entre otros, un lenguaje de programación, una herramienta de desarrollo de sistemas expertos, un soporte para la enseñanza de programación lógica y una herramienta que le brinda al alumno un mejor acercamiento a la Inteligencia Artificial y una posible interconexión entre estas dos áreas.

1 (Presentado en CICOM 2013)

* MSc. Ciencias - Astronomía, Matemática. Docente de la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. Universidad Nacional de Colombia. Bogotá, Colombia. Correo electrónico: smgils17@gmail.com

** MSc. Ciencias - Matemáticas. Docente de la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. Universidad Nacional de Colombia. Bogotá, Colombia. Correo electrónico: dianamb19@hotmail.com

Palabras clave: Lógica de primer orden, Prolog, Lisp, Inteligencia Artificial, enseñanza.

Abstract

Based on our teaching experience, is evidence of the need to give more relevance Prolog in the initial course of Mathematical Logic, showing how through the use of Prolog is possible to develop many ideas in students, among which stand out the understanding of exercises purely procedural and specification for resolving situations declaratively. This learning process can be generated by building a solid knowledge base in both propositional logic, as in first-order logic. This is why it seeks to expose in this article some of the alternatives to using Prolog (and other declarative programming languages like Lisp) for educational purposes, given that it is considered, among others, a programming language, a tool for development of expert systems, a support for teaching programming logic and a tool that gives the student a better approach to artificial intelligence and a possible interconnection between these two areas.

Key words: First-order logic, Prolog, Lisp, Artificial Intelligence, teaching

1. Introducción

Dentro de las propuestas tecnológicas de programación[3] se diferencian las de tipo imperativo, declarativo y las orientadas a objetos; cada una con ciertas características particulares que se enfocan a resolver diferentes tipos de problemas (ver Figura 1).

En el proceso de aprendizaje de programación, la Inteligencia Artificial puede verse desde varias perspectivas. Una de ellas es la que se basa en el hecho de tratar de explicar procesos basándose en la implementación de algoritmos para controlar diferentes cosas. Teniendo en cuenta esta característica, la IA proporciona respuestas a preguntas en varios tipos de lenguaje. Estos lenguajes deben permitir:

- Representar el conocimiento especializado
- Representar el conocimiento heurístico
- Realizar inferencias a partir del conocimiento representado, para obtener conclusiones.

Dentro de los lenguajes básicos de IA se destacan:

- Prolog (declarativo)
- Lisp (funcional)
- SmallTalk (orientados a objetos)

Cabe anotar que Prolog hace parte de los lenguajes lógicos y declarativos, lo que lo diferencia enormemente de otros lenguajes más populares tales como Fortran, Pascal, C o Java. Estos lenguajes declarativos tienen la

3 También llamadas paradigmas de programación.

ventaja de ser razonados matemáticamente, lo que permite el uso de mecanismos matemáticos para optimizar el rendimiento de los programas.

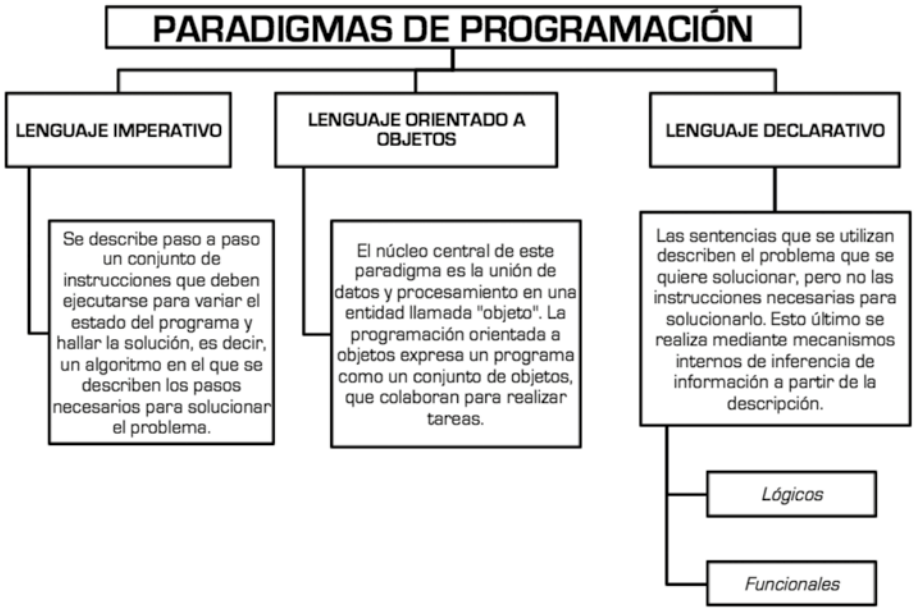
En este tipo de lenguaje se diferencian dos, a saber, los lenguajes lógicos como Prolog y los lenguajes funcionales como Lisp. Actualmente el paradigma de programación más usado en todos los niveles es la orientación a objeto; es quizás esta la razón por la cual se le ha restado importancia a Prolog.

La programación lógica se basa en el concepto de predicado o relación entre elementos. La programación funcional se fundamenta en el concepto de función visto como una evolución de los predicados, aspecto que le brinda a este tipo de programación un corte más matemático.

El lenguaje de programación lógica por excelencia es Prolog, un lenguaje ideado en los años 70 en Francia, y que tenía como objetivo inicial el procesamiento de lenguajes naturales que, en este momento, cuenta con diversas variantes. En la l*ó*gica matem*á*tica se encuentra la clave para entender este lenguaje. Esta área de la matemática proporciona la manera más sencilla para el intelecto humano de expresar formalmente problemas complejos y de resolverlos mediante la aplicación de reglas, hipótesis y teoremas. De ahí que el concepto de "programación lógica" resulte atractivo en diversos campos donde la programación tradicional presenta grandes dificultades.

La mayoría de los lenguajes de programación lógica se basan en lógica de primer orden o lógica de predicados, los cuales son un lenguaje formal con cuantificadores que tienen el poder suficiente para describir muchos hechos matemáticos y consecuencias al entrelazarlos.

Figura 1. Propuestas tecnológicas de programación



Fuente: elaboración propia.

En este orden de ideas Prolog resulta ser una herramienta relevante en el proceso de aprendizaje de deducciones, algoritmos y sistemas expertos; una de las áreas de la inteligencia artificial en la que se busca resolver un problema con un conjunto de programas que tiene un dominio o una base de conocimientos específicos.

El objetivo central de este artículo es rescatar todas aquellas características que hacen de Prolog una herramienta educativa *eficiente* en el proceso de construcción de conocimiento en ramas como programación, algoritmos, inteligencia artificial y establecer conexiones entre los programas académicos para evitar que estos procesos sean aislados entre las diferentes asignaturas del programa de Tecnología en Sistematización de Datos.

2. Programación declarativa

La programación declarativa es una forma de programación en donde se especifica *qué* debe hacerse y no *cómo* debe hacerse. A partir de este hecho el programador expresa las preguntas que deben ser respuestas por el programa, pero no índica el orden en que han de realizarse las acciones.

Kowalski define entonces un programa como la unión de *lógica* y *control*, donde el componente lógico determina el significado y el de control la eficiencia [8]. Es aquí donde la lógica cobra importancia, puesto que la característica fundamental de la programación declarativa es el uso de la lógica como lenguaje de programación. Se puede considerar entonces un programa como un sistema formal, con un conjunto de fórmulas lógicas que resultan de las especificaciones del problema que se quiere resolver y una semántica operacional que permitirá ejecutar los programas.

Dentro de la programación declarativa se distinguen:

- La programación lógica, que se fundamenta en fragmentos de la lógica de predicados: lógica de cláusulas de Horn.
- La programación funcional, que se basa en el concepto de función matemática y su definición mediante ecuaciones que constituyen el programa.

Algunas ventajas y desventajas de los lenguajes declarativos son [11]:

1. Los programas declarativos incluyen menos detalles que los programas imperativos.
2. En los programas imperativos un porcentaje muy alto del código está dedicado a controlar la secuencia de ejecución del programa (con comandos como if, while, repeat), mientras que en un lenguaje declarativo no existen este tipo de instrucciones. La repetición de operaciones se expresa mediante la recursión y la elección, entre varias alternativas es más abstracta.
3. Dado que el tiempo de desarrollo y el número de errores de un programa son directamente proporcionales al número de líneas escritas, independientemente del paradigma utilizado, los lenguajes declarativos ofrecen menores costes de desarrollo y mayor fiabilidad de los programas.
4. Una consecuencia de utilizar un paradigma declarativo es generalmente una mayor ineficiencia en el tiempo de ejecución de los programas.

2.1. Lógica de predicados

El conocimiento que se aplica para resolver un determinado tipo de problemas puede expresarse de diferentes formas, en particular de manera declarativa donde se especifican

los objetos, las propiedades y las relaciones generales, dejando al cuidado del agente que ha de resolver los problemas la aplicación de mecanismos generales de razonamiento. El tipo de conocimiento puede ser factual[4] o normativo[5], entre otros. Para representar el conocimiento se requiere de un lenguaje que presente ciertos requisitos: una sintaxis, una semántica y una pragmática.

La lógica formal proporciona un lenguaje para representar el conocimiento y brinda modelos para implementar procesos de razonamiento. Entre las lógicas base se encuentran la de proposiciones y la de predicados, distintos tipos para representar diversas conceptuaciones[6].

La lógica de proposiciones presenta ciertas limitaciones, ya que no permite representar razonamientos basados en propiedades, relaciones de tipo general y razonamientos que requieren conocimientos generales. En este caso las variables proposicionales quedan limitadas para el lenguaje. Se requiere de predicados, términos y cuantificadores, en otras palabras, se necesita de una lógica de predicados [ver figura 2].

Figura 2. Componentes de L_1

Fuente: elaboración propia.

Para representar una lógica de predicados de primer orden, una conceptuación debe estar formada por:

1. Universo del discurso, *U*: conjunto de objetos.
2. Conjunto de relaciones, *R* ($R \subseteq U^n$).
3. Conjunto de funciones, *F* ($F: U^n \rightarrow U$).
4. Conjunto de conocimientos sobre el dominio:

- Conocimientos factuales: relaciones y funciones definidas extensionalmente.
- Conocimientos normativos: relaciones y funciones definidas intencionalmente (enunciados compuestos con *y*, *o*, *no* y restricciones sobre los valores de enunciados: *si… entonces…*).

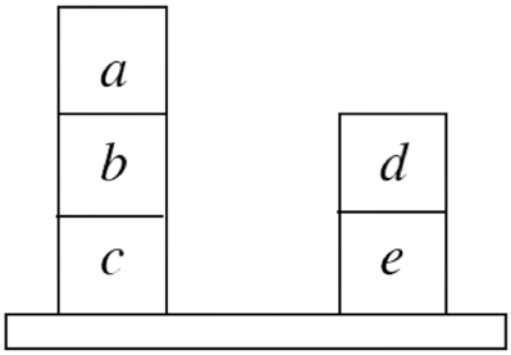
En el aula de clase estos conceptos parecen ser complicados para el estudiante, pero se pueden presentar bajo ejemplificaciones que les permitan verlos de manera sencilla y agradable. Una de estas situaciones es el mundo de bloques que se expone a continuación:

4 Memoria episódica: recuerdo de eventos (episodios experimentados: ayer me mordió un perro, dónde estuve el domingo, cómo fui a Tokio el año pasado. (Quillian, 1968).

5 Memoria semántica: guarda vocabulario, hechos, relaciones, procedimientos. Sin referencia a cómo, dónde o cuándo se han adquirido esos conocimientos: los perros muerden, las discotecas aturden, para viajes largos es mejor el avión. Procede de procesos de abstracción y generalización. (Quillian, 1968).

6 Formalmente, una conceptuación es una terna que consta de un universo del discurso, un conjunto de funciones de base y un conjunto de relaciones de base en ese universo del discurso. *Genesereth y Nilsson: Logical Foundations of Artificial Intelligence, 1987.*

Figura 3. Mundo de bloques



Fuente: elaboración propia.

La representación en lógica de predicados en este caso está dada por:

i. Universo del discurso: {a, b, c, d, e}.
- Torre A: compuesta por lo bloques *a, b, c.*
- Torre B: compuesta por lo bloques *d, e.*

ii. Propiedades, relaciones, restricciones, leyes.
- Relación encima de: *enc*
- Relación libre: *lib*
- Relación bajo: *baj*
- Relación más arriba que: *maq*
- Función sobre: *sob*

iii. Conjunto de conocimientos:

1. Conocimiento factual
- $enc(c) = b$; $enc(b) = a$; $enc(e) = d$.
- $lib(a)$; $lib(d)$.
- $baj(c)$; $baj(e)$.
- $maq(a, b)$; $maq(b, c)$; $maq(a, c)$; $maq(d, e)$.
- sob(a, b); sob(b, c); sob(d, e).

2. Conocimiento Normativo
- $y = enc(x) \Leftrightarrow sob(y, x)$.
- $lib(x) \Leftrightarrow (\nexists y)(sob(y,x))$.
- $maq(x, y) \Leftrightarrow sob(x, y) \vee (\exists z)(sob(z, y) \wedge maq(z, y))$.

Al establecer esta base de conocimiento, se puede pensar en preguntarse cosas acerca de ella, como por ejemplo: *¿Existe un bloque que se encuentre abajo del bloque b?* Esta pregunta expresada en lógica de predicados resulta:

$$\exists X\ baj(X, b)$$

Entonces, la respuesta debe ser Si. Considerando esta pregunta en Prolog la respuesta a la consulta debería ser Yes, según los hechos de la base de conocimiento. Aunque en Prolog, la respuesta va más allá:

```
?- bajo (X, b)
X=c
```

Con el entendimiento de estos conceptos clave, el estudiante podrá tener éxito en su paso a Prolog, pues una vez se haya establecido la conexión entre la sintaxis de lógica de predicados y la de Prolog, será mucho más sencillo implementar situaciones en el lenguaje de programación.

La lógica de primer orden es una poderosa herramienta del lenguaje matemático, ya que permite resolver e interpretar diversos problemas a partir de inferencias. Estas inferencias tienen respuesta afirmativa o negativa a partir de la lógica de predicados y cuando se implementa en Prolog, no solo se obtiene dicha respuesta sino que además es posible obtener un elemento con dicha característica. Por ejemplo cuando se habla de la inferencia:

Todos los hombres son mortales,
y dado que Sócrates es un hombre,
podemos concluir que Sócrates es mortal.

Esta se puede modelar como:

$(\forall_x (hom(x) \rightarrow mor(x)) \wedge hom(socrates)) \rightarrow mor(socrates)$,

que es una fórmula de la lógica de predicados de primer orden, donde *hom* y *mor*, son predicados, *x* es una variable cuantificada y *socrates* es una constante.

Desde otro punto de vista podríamos pensar en la misma fórmula anterior pero en un esquema de consulta donde:

$(\forall_x (hom(x) \rightarrow mor(x)$ (1)

$hom(socrates)$ (2)

son oraciones, y

Q: $mor(socrates)$?

es una consulta.

Luego, si suponemos las oraciones (1) y (2) como hechos verdaderos, y preguntamos por la consulta Q, la respuesta debiera ser SI.

Esto resulta ser clave ya que Prolog sigue un esquema similar de hechos y consultas.

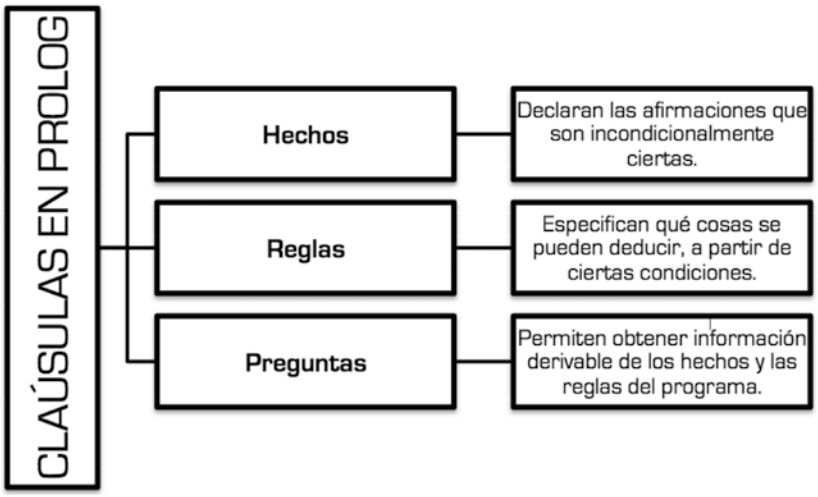
2.2. Prolog

El lenguaje Prolog fue desarrollado en la década de los setentas por Alain Colmerauer en la Universidad de Marsella. El nombre de Prolog (PROgrammation en LOGique) significa "Programación en lógica", indicando que sus bases descansan en la Lógica Matemática. Este lenguaje hace parte de la programación declarativa y se centra en un conjunto relativamente pequeño de mecanismos básicos que incluyen la unificación y el retroceso (backtracking) automático. Estas herramientas constituyen un marco de programación flexible y poderoso que facilita la solución de muchos problemas en el área de la Inteligencia Artificial [1].

Prolog es un lenguaje ideal para construir sistemas expertos al ser descriptivo. Este programa tiene una base de conocimiento y es capaz de manipularlo para obtener conclusiones. Una *base de conocimiento* está compuesta de hechos y reglas que permiten acercarse al problema a formalizar; para resolver el problema se pueden hacer preguntas cuya respuesta se puede encontrar en esta base. Para entender de manera adecuada lo que Prolog llama *hechos* se debe tener claro el concepto de predicado de la lógica de primer orden. Los predicados funcionan perfectamente cuando son claros los términos a analizar y las relaciones que pueden establecerse entre ellos. Si esto se da, Prolog permitirá obtener información adicional utilizando las *reglas*. En este caso las *reglas* son todas las relaciones de inferencia entre los predicados utilizando conectivas pero teniendo en cuenta que la conectiva dominante es la condicional [ver figura 4].

Es importante notar que si un "nuevo" programador desea obtener frutos al acercarse a Prolog, será relevante que su interpretación de la lógica de primer orden sea lo bastante clara para comprender el entorno del programa y los alcances que este tiene con respecto a la base de datos que esté manejando.

Figura 4. Elementos de Prolog



Fuente: elaboración propia.

El sistema establecido en Prolog tiene capacidad para inferir, a petición del usuario, consecuencias lógicas del conocimiento que se le ha proporcionado. Adicional a esto, se distinguen dos tipos de significados en los programas de Prolog:

- El *significado declarativo*, el cual está relacionado con las relaciones definidas en el programa y sirve para determinar cuál será la salida final del programa.
- El *significado procedural*, que establece cómo se va a obtener esta salida, es decir, cómo evalúa Prolog sucesivamente las relaciones definidas en el programa.

2.2.1. Un ejemplo

La lógica matemática en el aula debe reforzar los métodos en cuanto a la forma de razonamiento del estudiante. Los métodos deductivos, inductivos y analógicos o comparativos deben enseñarse paralelamente para que el estudiante al acercarse a problemas matemáticos, sea capaz de asimilarlos con toda la seguridad del caso.

Un ejemplo clásico en Prolog y con el cual se conecta fácilmente al estudiante con la lógica de predicados es el del árbol genealógico. Lo que para la lógica de primer orden es el universo del discurso, para Prolog equivale a la *Base de conocimiento*. Para este ejemplo, podemos considerar la muy conocida familia *The Simpsons*, familia propuesta como base para un quiz de inglés en la que se requiere establecer parentescos entre los diferentes miembros de la misma[7] [ver figura 5].

7 Ver http://www.englishexercises.org/makeagame/viewgame.asp?id=453

Figura 5. Simpsons Family Quiz

Fuente: adecuación propia de la imagen existente en la página web http://www.englishexercises.org/makeagame/viewgame.asp?id=453

Si se quiere modelar información genealógica acerca de esta familia, se pueden utilizar los siguientes predicados llamados en Prolog *hechos* y las siguientes *reglas* que relacionan por medio de conectivas estos predicados.

```
 /* --------- Hechos ---------*/

/*  --   hombre(X): X es un hombre --*/
hombre(abraham).
hombre(clancy).
hombre(herb).
hombre(homer).
hombre(bart).

/*  -- mujer(Y): Y es una mujer --*/
mujer(mona).
mujer(jackie).
mujer(marge).
mujer(patty).
mujer(selma).
mujer(lisa).
mujer(maggie).
mujer(ling).

/*  -- padre(X, Y): X es padre de Y --*/
padre(abraham, homer).
padre(abraham, herb).
padre(homer, bart).
padre(homer,lisa).
padre(homer, maggie).
padre(clancy, marge).
padre(clancy, patty).
padre(clancy, selma).

/*  -- madre(X, Y): X es madre de Y --*/
madre(mona, homer).
madre(jackie, marge).
madre(jackie, patty).
madre(jackie, selma).
madre(marge, bart).
```

```
madre(marge, lisa).
madre(marge, maggie).
madre(selma, ling).

/* esposos(X, Y):  El hombre X es esposo de la mujer Y */
esposos(abraham, mona).
esposos(clancy, jackie).
esposos(homer, marge).

/*------ Reglas ------*/

/* hermanos(X,Y):  X  es hermano de Y  */
hermanos(X,Y) :- padre(Z,X) , padre(Z,Y),   X \=Y.

/* abuelo(X,Y):  X  es abuelo de Y  */
abuelo(X,Y) :- padre(X, Z), madre(Z,Y).
abuelo(X,Y) :- padre(X, Z), padre(Z,Y).

/* abuela(X,Y):  X  es abuela de Y  */
abuela(X,Y) :- madre(X, Z), madre(Z,Y).
abuela(X,Y) :- madre(X, Z), padre(Z,Y).

/* tio(X,Y):  X  es el tio de Y  */
tio(X,Y) :- hombre(X), padre(Y, Z) , hermanos(X, Z).

/* tia(X,Y):  X  es la tia de Y  */
tia(X,Y) :- mujer(X), padre(Y, Z) , hermanos(X, Z).
```

Como se observa en este ejemplo, con las variables X, Y, Z se han establecido relaciones entre los predicados. Además, en la regla

abuelo(X,Y) :- padre(X, Z), madre(Z,Y).

X es abuelo de Y, si X es padre de Z y Z es madre de Y. Se puede observar que un mismo predicado puede estar definido por varias cláusulas gracias a la lógica de predicados.

Además en la cláusula anterior, se diferencian las *variables universales* X e Y, y la variable existencial Z. Esto se puede comprender a través de la equivalencia de la cláusula mencionada con la fórmula en lógica de predicados:

$$\forall_{x\,y\,z}(padre(x, z) \wedge madre(z, y) \rightarrow abuelo(x, y))$$
$$\forall_{x\,y}(\exists_{x}(padre(x, z) \wedge madre(z, y)) \rightarrow abuelo(x, y)).$$

Por otro lado el mecanismo de consulta que Prolog utiliza es muy interesante visto desde la lógica de primer orden. Por ejemplo, puede evaluarse cuál es el significado de

?- padre(X, bart).

En esta situación, cuando aparece una variable en una consulta, Prolog la interpreta como un existencial, es decir, equivale a

$$\exists X\ padre(X, bart)$$

La respuesta de Prolog debería ser Yes, según los hechos de la base de conocimiento, pero al igual que en el ejemplo del mundo de bloques, la respuesta va más allá pues en este caso el resultado es:

?- padre(X, bart).
X=homer

2.2.2. Otras aplicaciones de Prolog

Luego de haber motivado al estudiante con un ejemplo como el anterior, y luego de haber hecho evidente la conexión entre lógica de predicados y Prolog, se hace necesario manifestar que las aplicaciones de Prolog tie-

nen un alcance mucho mayor. Entre estas aplicaciones se encuentran [4]:

1. Sistemas expertos: permite almacenar datos y conocimiento, sacar conclusiones lógicas, tomar decisiones, aprender de la experiencia y los datos existentes, comunicarse con expertos humanos u otros sistemas expertos, argumentar las decisiones tomadas y realizar acciones.
2. Procesamiento de Lenguaje Natural (NLP): Prolog es un lenguaje muy abierto para formular algoritmos de lingüística computacional. Para el procesamiento de Lenguaje Natural (NLP), se requiere la manipulación de símbolos (palabras, fonemas y otras partes del lenguaje), basados en objetos estructurados (secuencias, árboles, grafos); Prolog facilita la expresión de operaciones sobre los símbolos y estructuras sin preocuparse de la representación y manipulación interna del computador.
3. Gramáticas independientes del contexto (G.I.C.): las frases del lenguaje son secuencias finitas de unidades léxicas ("tokens" o palabras) que vendrán representadas como listas de átomos Prolog. En este caso, una gramática es un sistema de reglas que define las frases del lenguaje.

2.3. Lisp

Teniendo en cuenta que en la programación declarativa se distinguen la *programación lógica* y la *programación funcional*, y que ya se expusieron algunas características que hacen de Prolog una herramienta educativa muy ´útil; no podemos dejar de lado la programación funcional. En este tipo de programación el lenguaje representativo es Lisp.

Lisp se originó en los años cincuenta, de la mano de John McCarthy. Su nombre proviene de "LISt Processing" (Procesamiento de listas) y junto con Prolog y Scheme hace parte del grupo de lenguajes expertos. Aunque en la actualidad abundan los lenguajes expertos, de acuerdo a cada aplicación específica, toman la base sintáctica de los anteriores. Este lenguaje se aplica en varias áreas:

- Robótica
- Inteligencia Artificial
- Procesamiento de Lenguaje Natural
- Demostración automática de teoremas.

Lisp es un lenguaje funcional que se apoya en la utilización de funciones matemáticas para el control de los datos. Cada función en Lisp y cada programa que generemos con él vienen dados en forma de lista. Por esta razón los datos no pueden diferenciarse sintácticamente de los programas. Los programas se comportan como funciones en el sentido matemático, dependiendo exclusivamente de sus datos de entrada.

Una de las razones por las que el Lisp está especialmente dotado para la programación en inteligencia artificial (IA), es precisamente porque su código y todos los datos tienen la misma estructura en forma de lista. Este lenguaje se sigue utilizando con frecuencia en investigación y fue considerado durante varios años el lenguaje modelo para la investigación de la inteligencia artificial (IA), aunque Prolog ha ganado terreno durante los últimos años.

2.3.1. Aplicaciones en matemáticas

Entre las más populares aplicaciones para matemáticas escritas en Lisp se encuentran:

1. *Maxima*: este sistema de álgebra computacional es un motor de cálculo simbólico que cuenta con un amplio conjunto de funciones para hacer manipulación simbólica de polinomios, matrices, funciones racionales, integración, derivación,

manejo de gráficos en 2D y 3D, manejo de números de coma flotante muy grandes, expansión en series de potencias y de Fourier, entre otras funcionalidades[8].

2. *ACL2*: es un lenguaje de programación, una lógica matemática para especificar y demostrar formalmente propiedades de los programas escritos en dicho lenguaje y un demostrador automático de teoremas que asiste al usuario en dicha tarea. *ACL2* usa el mismo lenguaje tanto para la implementación de los programas como para la especificación de sus propiedades. La lógica de *ACL2* es un subconjunto de la lógica de primer orden. Sus fórmulas no tienen cuantificadores y las variables de una fórmula se consideran (implícitamente) universalmente cuantificadas[9].

3. Conclusiones

- El método aplicado en el aula implica un proceso de ordenamiento y una dirección del pensamiento y de la acción para lograr unos objetivos previamente determinados. La técnica es la manera de utilizar los recursos didácticos para lograr una mayor efectividad en el aprendizaje del educando.
- La aplicación de Prolog en educación resulta atractiva por sus posibilidades como potenciador del pensamiento lógico. Sin embargo presenta la dificultad de una sintaxis compleja.
- Prolog se utiliza en educación como soporte para programas desarrollados en él. El hecho de que Prolog sirva de base a sistemas expertos en IA, le da una dimensión importante a este lenguaje.
- Prolog resulta ser una herramienta interesante cuando se tiene una fundamentación adecuada de la lógica matemática, esto hace referencia a tener claras varias pautas: ¿quién aprende?, ¿para qué aprende el estudiante?, ¿cómo aprende el estudiante?
- Prolog y Lisp fueron lenguajes creados con el objetivo de supuestos beneficios cognitivos que de ellos podían derivarse: desarrollar destrezas en la resolución de problemas, ayudar a la construcción de procedimientos o facilitar la compresión de los procesos inteligentes.
- Actualmente estos lenguajes han bajado su popularidad, no están de moda. Sin embargo, la lógica de predicados y Prolog funcionan muy bien paralelamente.
- Existen muchas aplicaciones que interrelacionan estos lenguajes de programación y otras áreas de la matemática en las cuales se utilizan programas implementados allí con el fin de resolver otras situaciones que no tienen que ver con deducción lógica.
- En este artículo se muestra que, aunque en otras entidades educativas tienen como fortaleza el trabajo con Prolog, en la Universidad Distrital se está forjando un camino para la enseñanza de estas herramientas tecnológicas que generen en el estudiante la capacidad de autodirigir su aprendizaje y transferirlo a otros ámbitos de su carrera y de su vida.

8 Tomado de http://es.wikipedia.org/wiki/Maxima Abril 15 de 2013

9 Tomado de http://es.wikipedia.org/wiki/ACL2, Abril 15 de 2013

4. Referencias

[1] Altamirano Carmona, Edgar, *Apuntes de programación en Prolog*, Universidad Autónoma de Guerrero, 2003.

[2] Bergin, Thomas J. & Gibson, Richard G., *History of Programming Languages II*, New York, ACM Press, Addison-Wesley, 1996.

[3] Colmerauer, Alain & Roussel, Philippe, *La naissance de Prolog*, July, 1992.

[4] Futch, Edgares, *Programación en Prolog para Inteligencia Artificial*, Congreso de Centro América y Panamá del Institute of Electrical and Electronics Engineers, Inc. CONCAPAN XXIII , Noviembre de 2003.

[5] García Mondaray, Sergio, *Programación declarativa: Manual básico de teoría*, 2008.

[6] Hernández E., Germán Ricardo, *PROLOG : Reflexiones sobre su potencial educativo*, Boletín de Informática Educativa Vol. 2, No. 2, Proyecto SIIE, Colombia, 1989.

[7] Kowalski, R. A, *The early years of logic programming*, Communications of the ACM, Volumen 31, No. 1, January, 1988.

[8] Kowalski, R. A, *Lógica, Programación e Inteligencia Artificial*, Ed. Diaz de Santos, 1986.

[9] Llorens Largo, Faraón & Castel de Haro, Ma. Jesús, *Prácticas de lógica - Prolog*, Universidad de Alicante, 1996-2001.

[10] Martínez Velarde, Juan, *El lenguaje PROLOG*, Periódico ABC, Madrid, 12 de Octubre de 1986.

[11] Peña Marí, Ricardo, *La programación declarativa*, Universidad Complutense de Madrid, 2009-2010.

[12] Pérez R., Jorge, *Prolog, Inteligencia Artificial*, Universidad de Talca, 2005.

[13] Soler Toscano, Fernando, *Modelos Formales de explicación en Lógica e Inteligencia Artificial*, Tesis Doctoral, Universidad de Sevilla, 2005.