Centro Universitario
UAEM Valle de México

INGENIERÍA EN SISTEMAS Y COMUNICACIONES

UDA: INTELIGENCIA ARTIFICIAL

TEMA: REPRESENTACIÓN DEL CONOCIMIENTO

ELABORÓ: DR. EN C. HÉCTOR RAFAEL OROZCO AGUIRRE
CU UAEM VM

Programa de Estudio Por Competencias
Inteligencia Artificial

1. IDENTIFICACIÓN DEL CURSO

ESPACIO ACADÉMICO: CENTRO UNIVERSITARIO UAEM VALLE DE MÉXICO								
PROGRAMA EDUCATIVO: INGENIERIA EN SISTEMAS Y COMUNICACIONES					**Área de docencia:** INGENIERIA APLICADA			
Aprobación por los H. H. Consejos Académico y de Gobierno		**Fecha:**			**Programa elaborado por:** MARICELA QUINTANA LOPEZ, SATURNINO JOB MORALES ESCOBAR		**Fecha de elaboración:** ENERO 2012	
Clave	**Horas de teoría**	**Horas de práctica**	**Total de horas**	**Créditos**	**Tipo de Unidad de Aprendizaje**	**Carácter de la Unidad de Aprendizaje**	**Núcleo de formación**	**Modalidad**
L32310	2	2	4	6	CURSO	OPTATIVA	INTEGRAL	PRESENCIAL
Prerrequisitos (Conocimientos Previos) MATEMÁTICAS DISCRESTAS,LÓGICA MATEMÁTICA			**Unidad de aprendizaje antecedente** NINGUNA			**Unidad de aprendizaje consecuente** NINGUNA		
Programas educativos en los que se imparte: INGENIERIA EN SISTEMAS Y COMUNICACIONES								

¿Cómo representamos lo que conocemos?

Para responder a esta pregunta, se requiere de un análisis para distinguir entre el CÓMO y el QUÉ.

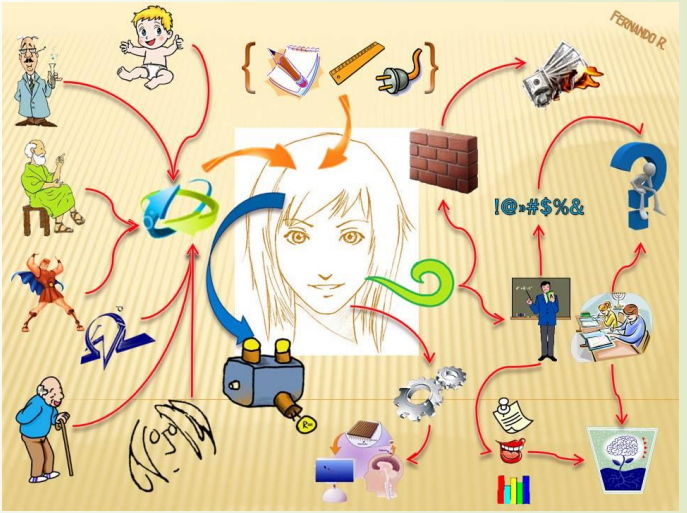
- CÓMO: Significa cómo realizamos alguna cosa.

- QUÉ: Significa qué cosas consideramos verdaderas o falsas.

Conocimiento y Representación

Son entidades o conceptos diferentes. Estos juegan un rol muy importante en nuestro sistema inteligente.

- Conocimiento: Es una descripción del mundo. El conocimiento determina la capacidad de un sistema por lo que sabe.
- Representación : Es la manera en que el conocimiento es codificado.

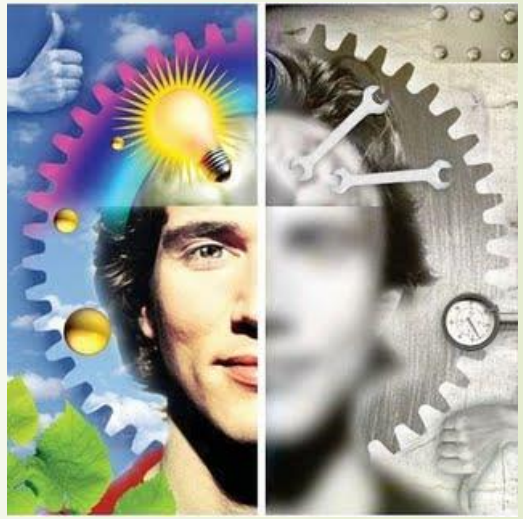


…Conocimiento y Representación

Para cada tipo de conocimiento se requiere diferente tipo de representación.

Los modelos o mecanismos de la representación de conocimiento están basados regularmente en:

- Lógica
- Reglas
- Redes semánticas
- Marcos



Los diferentes tipos de conocimiento requieren diferente tipo de razonamiento.

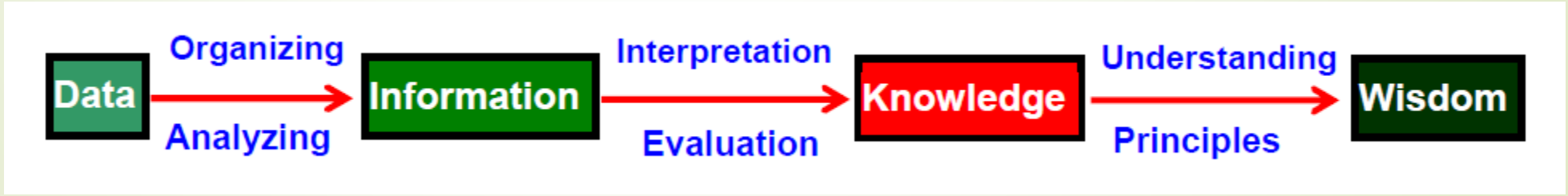
Conocimiento

El conocimiento es una progresión que inicia con los **Datos** los cuales tiene una limitada utilidad.

A través de la organización y análisis de los datos, éstos pueden convertirse para nosotros en I**nformación**.

La interpretación o evaluación de la información produce **Conocimiento**.

La comprensión de los principios relacionados al conocimiento se le llama **Sabiduría**.



Modelo de Conocimiento

El Modelo de Conocimiento muestra que, tanto el grado de conectividad y entendimiento se Incrementan . Es decir, vamos desde los datos, a través de la información y el conocimiento, hasta la sabiduría.

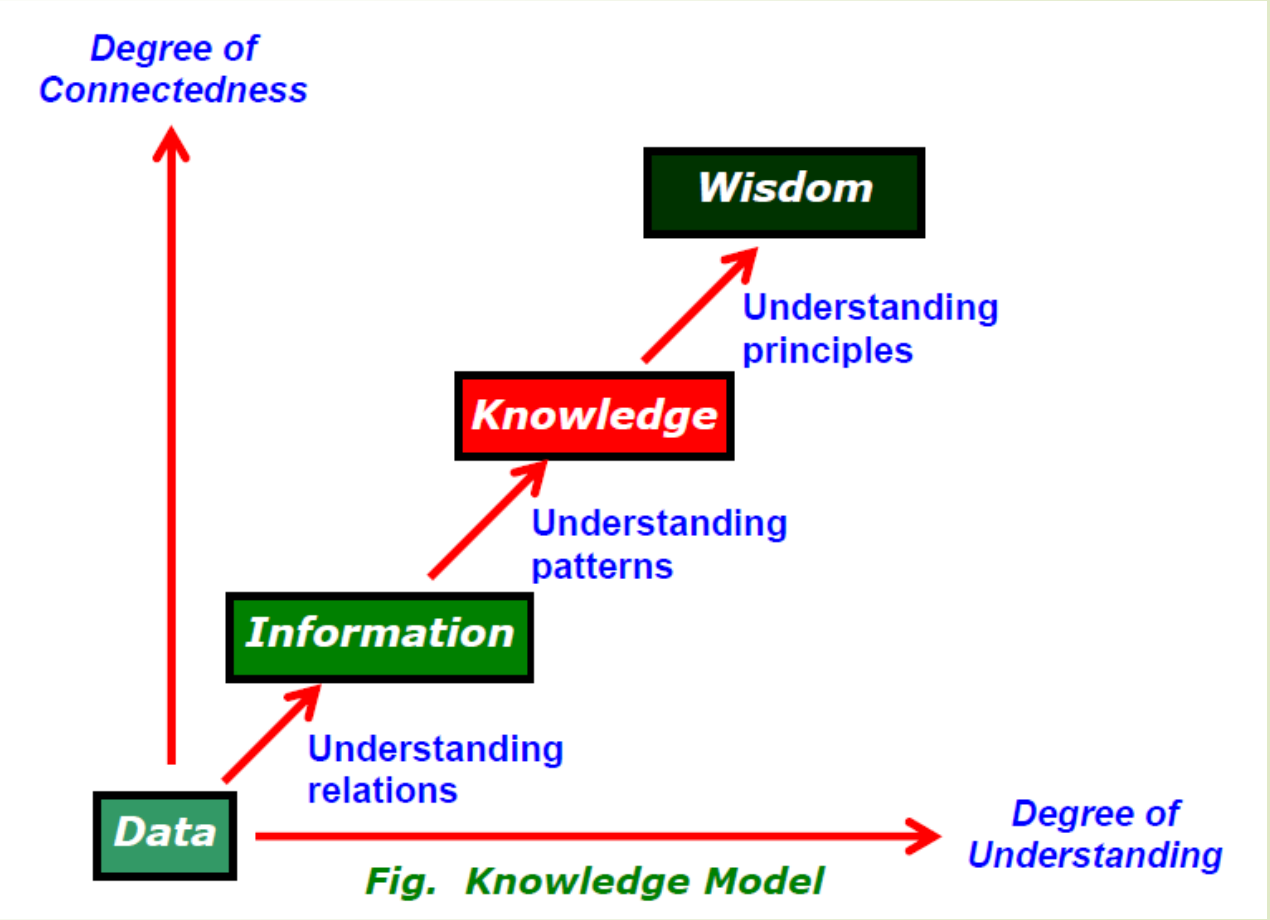


Fig. Knowledge Model

Modelo de Conocimiento

El Modelo de Conocimiento muestra que, tanto el grado de conectividad y entendimiento se Incrementan . Es decir, vamos desde los datos, a través de la información y el conocimiento, hasta la sabiduría.

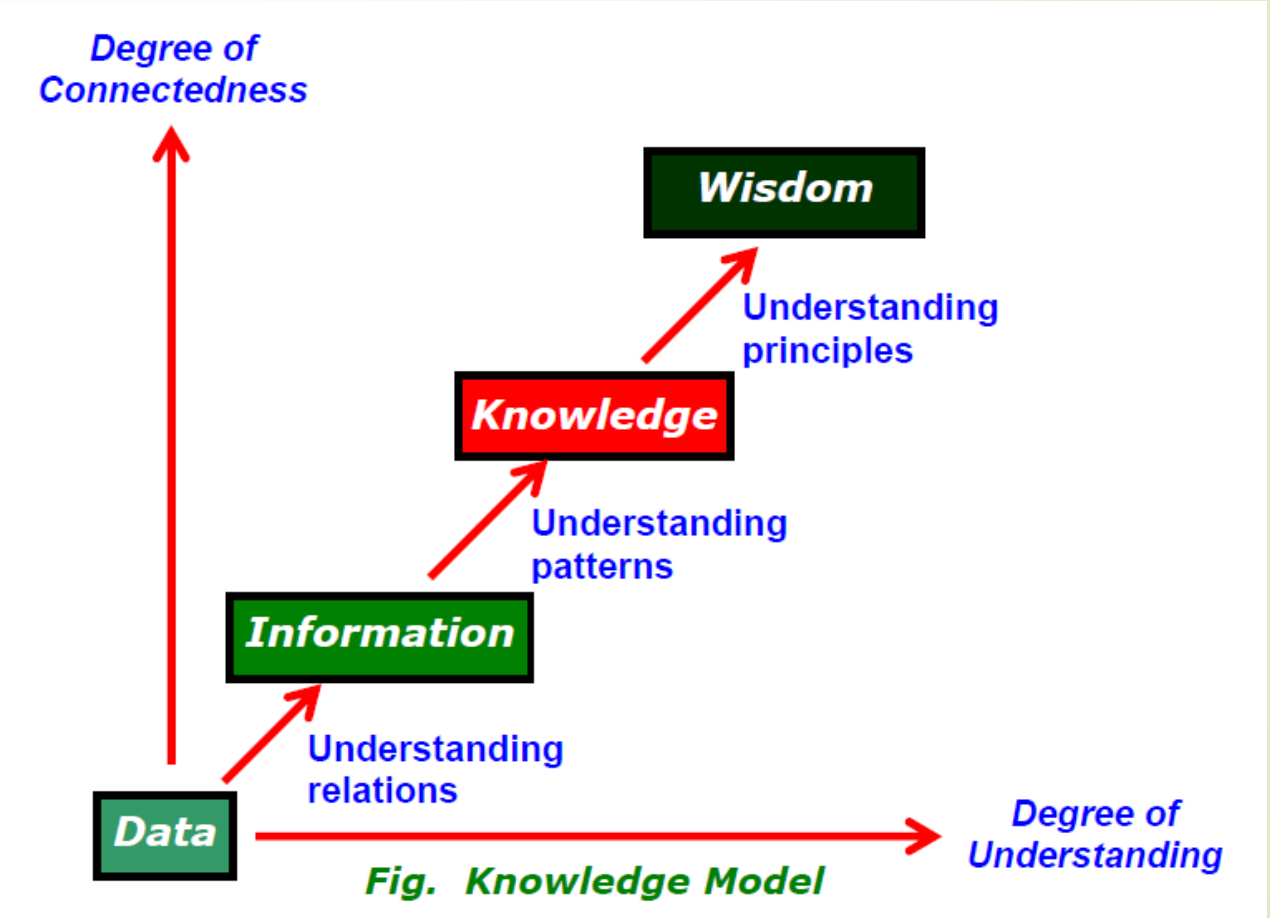


Fig. Knowledge Model

Fuente del conocimiento

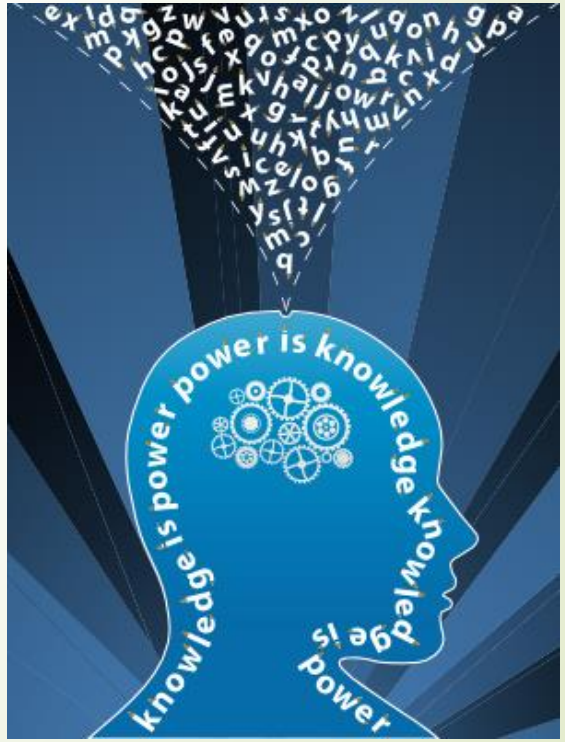
El conocimiento en cualquier campo es generalmente de dos tipos:

- Conocimiento publico:

  Definiciones, hechos y teorías.

- Conocimiento privado:

  reglas de oro (heurística).



Tipos de conocimiento

- Conocimiento de sentido común y conocimiento de sentido común informado: principios generales y conceptos del dominio.
- Conocimiento Heurístico: Se trata de una regla de oro o un argumento derivado de la experiencia.
- Conocimiento del dominio: Conocimiento especifico en un dominio.
- Metaconocimiento: Conocimiento acerca del conocimiento.

Tipos de conocimiento

- De acuerdo a su uso:
  - Conocimiento condicional: provee información de restricciones y prerrequisitos.
  - Conocimiento de utilidad: provee funciones de utilidad sobre estados futuros.
  - Conocimiento de acción: conduce al mejor curso de acción.
  - Conocimiento de objetivos: especifica alta utilidad sobre estados deseados.

Tipos de conocimiento

- Conocimiento de acuerdo a su naturaleza
  - Tácito: Conocimiento embebido en la mente a través de la experiencia
  - Explicito: fácil de extraer y codificar de diversas fuentes.
- El conocimiento puede ser permanente, estático(periodo) o dinámico.

Tipos de conocimiento

Knowledge Type	Meaning	Example
Permanent	Knowledge that never changes, like physical laws	The earth moves around the Sun
Static	Knowledge that is constant over a given period	Policies and procedures
Dynamic	Knowledge that is continuously changing	Prices of shares and gold

Características deseables del conocimientos

- Naturalidad: Facilidad de representar el conocimiento de forma natural.
- Transparencia: Facilidad de identificar el conocimiento almacenado.
- Adecuación y exhaustividad: Contener todos los elementos requeridos par resolver el problema.
- Modularidad: Facilidad de almacenamiento de los elementos del conocimiento.

Características deseables del conocimientos

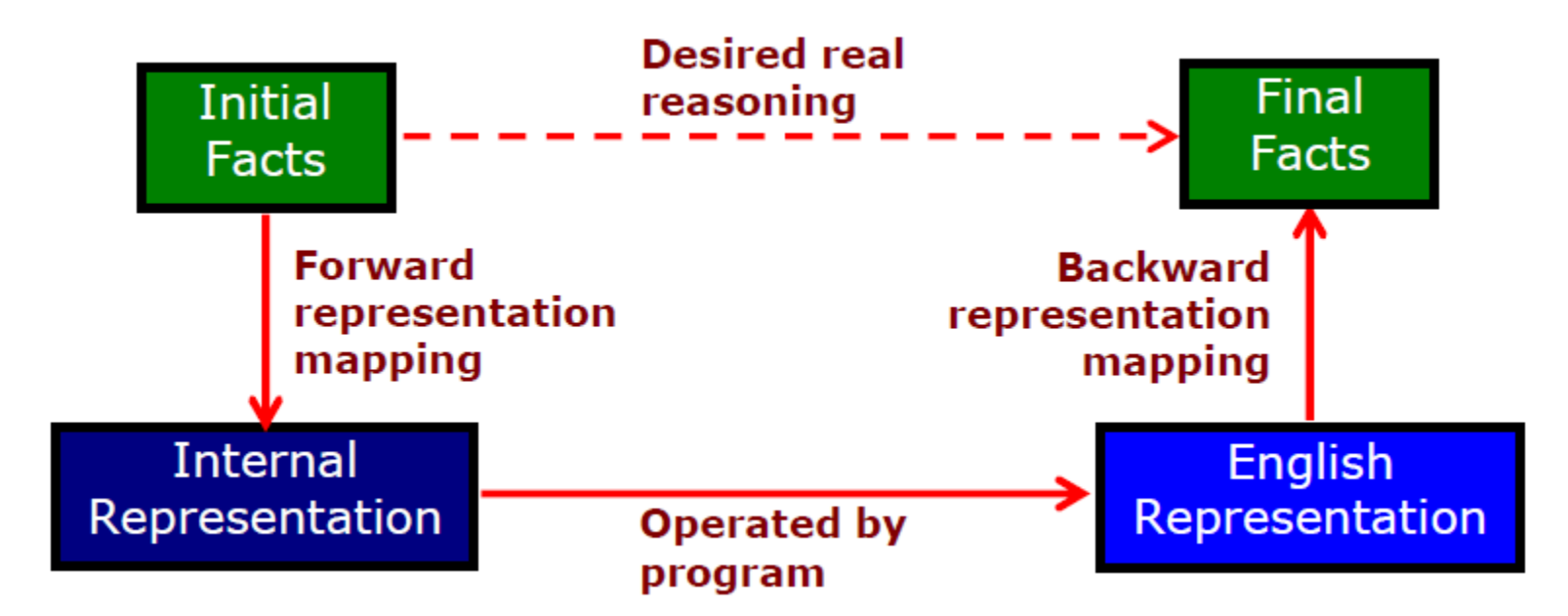
- Utilidad: Grado en el cual el conocimiento es útil para resolver un problema del dominio.
- Claridad: facilidad de representar el conocimiento directamente.
- Facilidad de operación, fácil acceso y eficiente.

Categorías del conocimiento

- Algunos psicólogos cognitivos organizan el conocimiento en dos categorías: el **Declarativo** y el **Procedimental**; algunos investigadores añaden una tercera: el **Estratégico.**
- Componentes declarativos: representación descriptiva
  - Hechos
  - Reglas
- Componentes procedimentales: resulta de las habilidades intelectuales para hacer las cosas.
  - Heurística y conocimiento de sentido común.

The Forward and Backward Representation

- La línea punteada indica el razonamiento abstracto.
- Las líneas continuas indican el razonamiento concreto.



Encadenamiento hacia adelante

- Entradas y datos son almacenados en la memoria de trabajo.
- Las entradas de trabajo disparan las reglas condicionales que cumplen las restricciones. Esas reglas ejecutan sus acciones.
- Las acciones pueden agregar nuevos datos a la memoria, y estas pueden disparar mas reglas.
- Es llamada también inferencia dirigida a los datos

Encadenamiento hacia adelante

- Es apropiado cuando :
- hay suficiente información acerca de un entorno para concluir
- Hay un estado inicial único
- Es difícil elaborar una meta para verificar.
- Cuando una meta es impredecible o sin importancia
- Puede ser utilizado en la fase de mejora de calidad en el proceso de desarrollo de software o mejora un proceso organizacional.

Encadenamiento hacia atrás

- Es apropiado cuando:
- La meta esta dada o es evidente
- Restricciones del entorno o los datos no son claros.
- Datos relevantes deben ser adquiridos durante el proceso de inferencia
- Existe un gran numero de reglas aplicables.

Encadenamiento hacia atrás

- Cualquier sistema de reglas con encadenamiento hacia atrás puede ser reescrito como un equivalente a un sistema de encadenamiento hacia adelante
- Frecuentemente ofrece mejor justificación o explicación de cómo llegar a una meta en particular
- Puede ser utilizado en sistemas para el aprendizaje, entrenamiento y diagnostico médico
- Es recomendable cuando hay pocos objetivos y poco numero de reglas sobre un gran numero de hechos.

Esquemas de Representación del Conocimiento

- Conocimiento **Relacional**.- Comparación de dos objetos basados en atributos equivalentes.
- Conocimiento **Heredado**.- Se obtienen a partir de la asociación de objetos.
- Conocimiento **Inferencial**.- Se infiere a través de las relaciones entre los objetos.
- Conocimiento **Declarativo**.- Enunciados en los cuales se especifica el conocimiento.
- Conocimiento **Procedimental**.- Se agrega información de control para usar el conocimiento.

RC usando Lógica de Predicados

Supuestos acerca de la Representación del Conocimiento (RC)

- Un comportamiento inteligente se puede lograr mediante la manipulación de las estructuras de símbolos.
- El lenguaje y el diseño facilitan operaciones sobre estructuras de símbolos, teniendo en cuenta la sintaxis y la semántica.
- Hacer inferencias, elaborar nuevas conclusiones a partir de los hechos existentes.

RC usando Lógica de Predicados

- Lógica.- se ocupa de la verdad de las declaraciones sobre el mundo. Generalmente cada declaración puede ser verdadera o falsa.

  La lógica incluye:

  - ✓ Sintaxis
  - ✓ Semántica
  - ✓ Procedimiento de inferencia

RC usando Lógica de Predicados

- ✓ Sintaxis.- Especifica los **símbolos** del lenguaje acerca de como pueden combinarse para formar sentencias. Los hechos acerca del mundo son representados como sentencias.
- ✓ Semántica.- Especifica como asignar un valor de verdad para una sentencia en base a su **significado** en el mundo
- ✓ Procedimiento de inferencia.- Especifica **métodos** para calcular nuevas sentencias a partir de las sentencias existentes.

RC usando Lógica de Predicados

- Lógica como lenguaje RC.- La lógica es un lenguaje para razonar, una colección de reglas utilizadas mientras se hace el razonamiento lógico.
- ✓ Lógica
- ✓ Problema de diseño del lenguaje RC:
  a) Representar objetos y las relaciones del dominio
  b) Preguntas y respuestas en un tiempo razonable
- ✓ Diferentes tipos de lógica
- ✓ Lógica proposicional y lógica de predicado

RC usando Lógica de Predicados

- Lógica Proposicional (PL).-

  Proposición: Una oración afirmativa de la cual podemos decir que es **verdadera** o **falsa** (pero no ambas!!)

  Ejemplo:

  a) El cielo es azul
  b) La nieve es fría
  c) 12 * 12 = 144

RC usando Lógica de Predicados

- Lógica Proposicional (PL)
- ✓ Las proposiciones son sentencias, que pueden ser verdadero o falso, pero no ambos.
- ✓ Una sentencia u oración es la unidad mas pequeña en PL
- ✓ Si la proposición es verdadera, entonces el valor de verdad es verdadero
- ✓ Si la proposición es falsa, entonces el valor de verdad es falsa.

RC usando Lógica de Predicados

- Lógica Proposicional (PL)

Sentence	Truth value	Proposition (Y/N)
"Grass is green"	"true"	Yes
"2 + 5 = 5"	"false"	Yes
"Close the door"	-	No
"Is it hot out side ?"	-	No
"x > 2" where x is variable	-	No (since x is not defined)
"x = x"	-	No (don't know what is "x" and "="; "3 = 3" or "air is equal to air" or "Water is equal to water" has no meaning)

RC usando Lógica de Predicados

- Lógica Proposicional (PL)

Conectores y símbolos en orden de prioridad

Connective	*Symbols*					Read as
assertion	P					"p is true"
negation	¬p	~	!		NOT	"p is false"
conjunction	p ∧ q	·	&&	&	AND	"both p and q are true"
disjunction	P v q	||	|		OR	"either p is true, or q is true, or both "
implication	p → q	⊃	⇒		if ..then	"if p is true, then q is true" " p implies q "
equivalence	↔	≡	⇔		if and only if	"p and q are either both true or both false"

RC usando Lógica de Predicados

- Lógica Proposicional (PL)

Tabla de verdad

p	q	¬p	¬q	p ∧ q	p v q	p→q	p ↔ q	q→p
T	T	F	F	T	T	T	T	T
T	F	F	T	F	T	F	F	T
F	T	T	F	F	T	T	F	F
F	F	T	T	F	F	T	T	T

RC usando Lógica de Predicados

- Lógica Proposicional (PL)
  - **Tautologías**

    Una proposición que siempre es verdadera es llamada tautología

    Ejemplo: **(p v ¬p)** es siempre verdadera independiente del valor de verdad de la proposición p

RC usando Lógica de Predicados

- Lógica Proposicional (PL)
  - **Contradicciones**

    Una proposición que siempre es falsa es llamada contradicción

    Ejemplo: **(p ^ ¬p)** es siempre falsa independiente del valor de verdad de la proposición p

RC usando Lógica de Predicados

- Lógica Proposicional (PL)

- **Contingencias**

  Una proposición es llamada contingencia, si esa proposición no es ni una tautología ni una contradicción.

  Ejemplo: (p v q)

RC usando Lógica de Predicados

- Lógica Proposicional (PL)

- **Antecedente, Consecuencia**

  En las sentencias condicionales, **p** → **q**,

  1ra. Sentencia "**if-**cláusula" (aquí p) es llamada antecedente)

  2da. Sentencia "**then**-cláusula" (aquí q) es llamada consecuencia

RC usando Lógica de Predicados

- Lógica Proposicional (PL)
  - Argumento.- es una demostración o prueba de alguna sentencia.

    Ej.: **Ese pájaro es un cuervo, por lo tanto, es negro**

  - ✓ **Premisa**: es una proposición que da razones, motivos o evidencia para aceptar alguna otra proposición, llamada conclusión.
  - ✓ **Conclusión**: es una proposición que supone estar establecida en base a otra proposición

RC usando Lógica de Predicados

- Lógica de predicados.

La principal debilidad de la lógica proposicional es su limitada habilidad para expresar conocimiento. Existen varias sentencias complejas que pierden mucho de su significado cuando se las representa en lógica proposicional. Por esto se desarrolló una forma lógica más general, capaz de representar todos los detalles expresados en las sentencias, esta es la ***lógica de predicados***.

RC usando Lógica de Predicados

- Lógica de predicados.

La lógica de predicados está basada en la idea de que las sentencias realmente expresan relaciones entre objetos, así como también cualidades y atributos de tales objetos.

Predicado: cada sentencia completa tiene dos partes, un **sujeto** y un **predicado.**

Ejemplo: Judy corre

RC usando Lógica de Predicados

- Lógica de predicados.
  - Operadores lógicos
    - ✓ Conjunción (**AND** - **&&**)
    - ✓ Disyunción (**OR** - **||**)

```
x < y || ( y < z && z < x)

true || ( true && true)
```

→ **True**

```
3 < 2 || ( 2 < 1 && 1 < 3)

False
```

RC usando Lógica de Predicados

- Lógica de predicados.
  - Cuantificadores
  - ✓ Universal ∀
    - ∀x "Establece que para todo X es verdad que…."
  - ✓ Exitencial ∃

RC usando Lógica de Predicados

- Lógica de predicados.
  - Universo de discurso

Ejemplo:

Si algunos trenes se retrasan entonces todos se retrasan

y sólo hablamos de trenes

$$(\exists x)\ \mathbf{R(x)} \rightarrow\ (\forall \mathbf{x})\ \mathbf{R(x)}$$

Todo número es par o impar

y sólo hablamos de naturales

$$(\forall \mathbf{x})\ (\mathbf{P(x)} \vee \mathbf{I(x)})$$

RC usando Lógica de Predicados

- Lógica de predicados.
- Ejemplos:

  ∀x esHombre(x)

  Será verdad cuando todos los objetos del dominio de discurso satisfagan el predicado esHombre.

  Si en nuestro dominio de discurso hay objetos “no hombres” (p.ej. Gatos), será falso.

RC usando Lógica de Predicados

- Lógica de predicados.
- Ejemplos:

  ∃x esHombre(x)

  Será verdad cuando exista algún objeto del dominio de discurso que satisfaga el predicado esHombre.

  Con tener un hombre en el dominio de discurso, será cierto, aunque también haya gatos…

Representación del conocimiento basado en **reglas**

- En el tema anterior se presentó el uso de la Lógica de Predicados.

  Los otros enfoques de la representación del conocimiento son:

- Reglas de producción
- Redes semánticas
- Frames

Reglas de producción

- Algunas veces llamadas reglas IF-THEN son las más utilizadas en KR.

- Las reglas de producción proveen la flexibilidad de combinar la representación declarativa y procedimental.

- Ejemplo:

  IF condition THEN action

  IF premise THEN conclusion

  IF preposition p1 y preposition p2 are true

  THEN proposition p3 is true

Ventajas de las reglas de producción

- Son modulares
- Cada regla define una pequeña e independiente pieza del conocimiento.
- Se pueden añadir nuevas reglas y las viejas pueden eliminarse.
- Las reglas son regularmente independientes de otras reglas.
- Las reglas de producción como mecanismo de representación del conocimiento son utilizadas en el diseño de muchos "**Sistemas basados en reglas**" también llamados "**Sistemas de producción**".

Tipos de reglas de producción

Existen tres tipos de reglas, definidas como las más utilizadas:

- Reglas **declarativas** de conocimiento:

  Estas reglas establecen todos los hechos y relaciones acerca del problema.

- Reglas **procedimentales** de inferencia:

  Estas reglas guían sobre como resolver un problema, mientras la certeza de los hechos sea conocida.

  Estas reglas son parte del *motor de inferencia*.

- **Meta-reglas** (Reglas para hacer reglas):

  Las meta reglas especifican cuales deben ser consideradas y en que orden ser llamadas.

3.3.1 Conocimiento Procedimental contra Declarativo

- Conocimiento procedimental: define el **cómo** hacer.

Ejemplo: *Para definir si Pedro o Roberto es más grande, primero encontrar sus edades.*

Aquí se describen tareas y métodos.

- Conocimiento declarativo: define el **qué**

Incluyen: Conceptos, objetos, hechos, proposiciones, modelos.

Ejemplo: *Un auto tiene cuatro ruedas; Pedro es más grande Roberto*

Aquí se describen hechos y cosas

Comparación

Procedural Knowledge	Declarative Knowledge
• Hard to debug	• Easy to validate
• Black box	• White box
• Obscure	• Explicit
• Process oriented	• Data - oriented
• Extension may effect stability	• Extension is easy
• Fast , direct execution	• Slow (requires interpretation)
• Simple data type can be used	• May require high level data type
• Representations in the form of sets of rules, organized into routines and subroutines.	• Representations in the form of production system, the entire set of rules for executing the task.

Comparación entre lenguajes

Procedural Language	Declarative Language
Basic, C++, Cobol, etc.	SQL
Most work is done by interpreter of the languages	Most work done by Data Engine within the DBMS
For one task many lines of code	For one task one SQL statement
Programmer must be skilled in translating the objective into lines of procedural code	Programmer must be skilled in clearly stating the objective as a SQL statement
Requires minimum of management around the actual data	Relies on SQL-enabled DBMS to hold the data and execute the SQL statement .
Programmer understands and has access to each step of the code	Programmer has no interaction with the execution of the SQL statement
Data exposed to programmer during execution of the code	Programmer receives data at end as an entire set

…comparación entre lenguajes

• More susceptible to failure due to changes in the data structure	• More resistant to changes in the data structure
• Traditionally faster, but that is changing	• Originally slower, but now setting speed records
• Code of procedure tightly linked to front end	• Same SQL statements will work with most front ends Code loosely linked to front end.
• Code tightly integrated with structure of the data store	• Code loosely linked to structure of data; DBMS handles structural issues
• Programmer works with a pointer or cursor	• Programmer not concerned with positioning
• Knowledge of coding tricks applies only to one language	• Knowledge of SQL tricks applies to any language using SQL

3.3.1 Programación Lógica

- La programación lógica ofrece un formalismo para especificar una computación en términos de las relaciones lógicas entre entidades.

- Programa lógico.- Es una colección de sentencias lógicas.

- Programador.- Describe todas las relaciones lógicas entre las diferentes entidades.

- Computación.- Determina si una conclusión en particular es consecuencia de estas sentencias lógicas.

Características de un programa lógico

Un programa es caracterizado por un conjunto de relaciones e inferencias.

- Programa.- Consiste de un conjunto de axiomas y una sentencia meta.
- Reglas de inferencia.- Determinan si los axiomas son suficientes para asegurar la certeza de la sentencia meta.
- Ejecución.- De un programa lógico corresponde a la construcción de una prueba de la sentencia meta con respecto a los axiomas.
- Programmer.- Especifica las relaciones lógicas básicas , no especifica la forma en que las reglas de inferencia son aplicadas.

Entonces: **LOGICA + CONTROL = ALGORITMOS**

Ejemplo de Sentencias Lógicas

- Sentencia:

  A grand-parent is a parent of a parent.

- Sentencia expresada en términos de lógica:

  A person is a gran-parent if she/he tiene un hijo y este hijo tes un padre.

- Sentencia expresada como lógica de primer orden:

  (for all) X : grandparent(x,y) :- parent(x,z), parent(z,y)

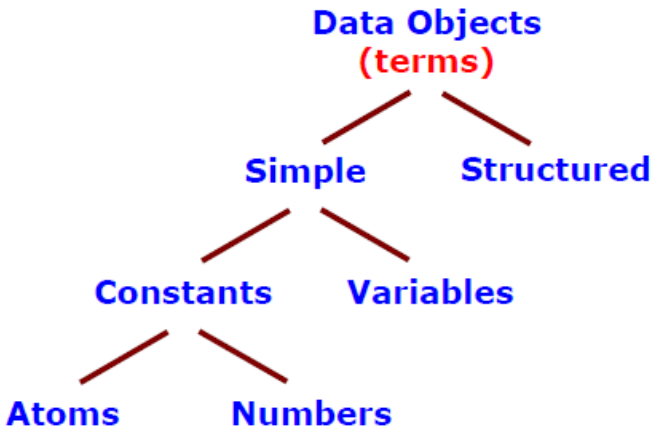
  Se leé: X es un grand-parent de y, si x es un padre de z y z es un padre de y

Lenguaje de Programación Lógica

- Un lenguaje de programación lógica incluye:
  - La Sintaxis
  - La semántica de los programas
  - El modelo computacional
- Prolog y logic data language (LDL) son ejemplos de lenguajes declarativos.
- Prolog (PROgramming LOGic).- Es el lenguaje más popular en el campo de la Inteligencia Artificial. Su popularidad se basa en la utilidad para representar conocimiento acerca del QUÉ y el CÓMO

Sintaxis y Terminología (referente a PROLOG)

- En cualquier lenguaje, la formación de componentes (expresiones, sentencias, etc.), esta guiada por reglas sintácticas.
- Los componentes están divididos en dos partes:
  - Componentes de datos ,y
  - Componentes de programa.
- Los componentes de datos son una colección de objetos de datos jerárquicos.



Data object of any kind is also called a *term*. A term is a constant, a variable or a compound term.

Simple data object is not decomposable; e.g. *atoms, numbers, constants, variables*.

Syntax distinguishes the data objects, hence no need for declaring them.

Structured data object are made of several components.

(a) Data Objects

- Los objetos de datos de cualquier tipo son llamados un **término**.
- Ejemplos de término:
  - Constantes : Enteros, PuntoFlotante, Atoms.
  - Variables : Símbolos
  - Términos compuestos: Función con uno o más argumentos.
  - Ground an nonGround: Los términos son Ground si contienen no variables (sólo constantes); de otra forma son nonGround.

    Goals son atoms or términos compuestos , y son generalmente nonGround

(b) Simple Data Objects

Atoms

‡ a lower-case letter, possibly followed by other letters of either case, digits, and underscore character.

e.g. **a** **greaterThan** **two_B_or_not_2_b**

‡ a string of special characters such as: + - * / \ = ^ < > : ~ # $ &

e.g. **<>** **##&&** **::=**

‡ a *string* of any characters enclosed within single quotes.

e.g. **'ABC'** **'1234'** **'a<>b'**

‡ following are also *atoms* **!** **;** **[]** **{}**

Numbers

‡ applications involving heavy numerical calculations are rarely written in Prolog.

‡ *integer* representation: **e.g. 0 -16 33 +100**

‡ *real numbers* written in standard or scientific notation,

e.g. **0.5** **-3.1416** **6.23e+23** **11.0e-3** **-2.6e-2**

Variables

‡ begins by a capital letter, possibly followed by other letters of either case, digits, and underscore character.

e.g. **X25** **List** **Noun_Phrase**

(c) Structured Data Objects

General Structures

- a structured term is syntactically formed by a functor and a list of arguments.
- functor is an atom.
- list of arguments appear between parentheses.
- arguments are separated by a comma.
- each argument is a term (i.e., any Prolog data object).
- the number of arguments of a structured term is called its *arity*.
- e.g. **greaterThan(9, 6)** **f(a, g(b, c), h(d))** **plus(2, 3, 5)**

(c) Structured Data Objects

General Structures

- a structured term is syntactically formed by a functor and a list of arguments.
- functor is an atom.
- list of arguments appear between parentheses.
- arguments are separated by a comma.
- each argument is a term (i.e., any Prolog data object).

the number of arguments of a structured term is called its *arity*.

e.g. **greaterThan(9, 6)** **f(a, g(b, c), h(d))** **plus(2, 3, 5)**

Special Structures

- In Prolog an ordered collection of terms is called a *list* .
- Lists are structured terms and Prolog offers a convenient notation to represent them:
  - Empty list is denoted by the atom **[ ]**.
  - Non-empty list carries element(s) between square brackets, separating elements by comma.

    e.g. **[bach, bee]** **[apples, oranges, grapes]**

(B) Componentes de Programa

- Un programa PROLOG es una colección de predicados o reglas.
- Un predicado establece una relación entre objetos.
- Cláusula.- Es una colección de palabras relacionadas gramáticamente.
- Predicado.- Es compuesto de una o más cláusulas.
- Las cláusulas constituyen sentencias, cada sentencia contiene una o más cláusulas.
- Una sentencia contiene dos partes: **sujeto** y **predicado**.

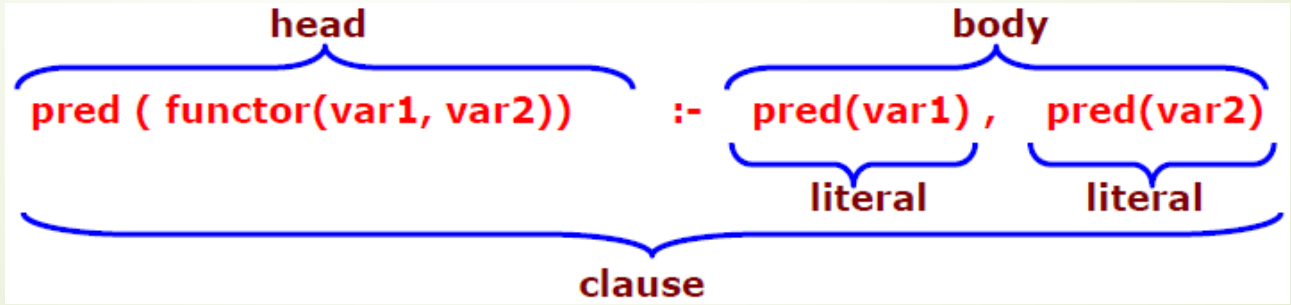
(b) Predicados y Cláusula

- Sintácticamente un predicado esta compuesto de una o más cláusulas.
- La forma general de las cláusulas es:

  <left-hand-side> :- <rigth-hand-side>

  donde LHS es una sola meta llamada "goal" y RHS es compuesta de una o más metas, separada por comas, llamada "sub-goals" sobre la meta del lado izquierdo.

  El símbolo " :- " es pronunciado como "este es el caso" o "tal que".
- La estructura de una cláusula en programación lógica:



(b) Predicados y Cláusula

- Las literales representan la posible elección de tipos primitivos del lenguaje en particular.

Algunos de estos tipos de literales son regularmente integers, floating point, Booleans y cadenas de carácter.

Example : **grand_parent (X, Z) :- parent(X, Y), parent(Y, Z).**

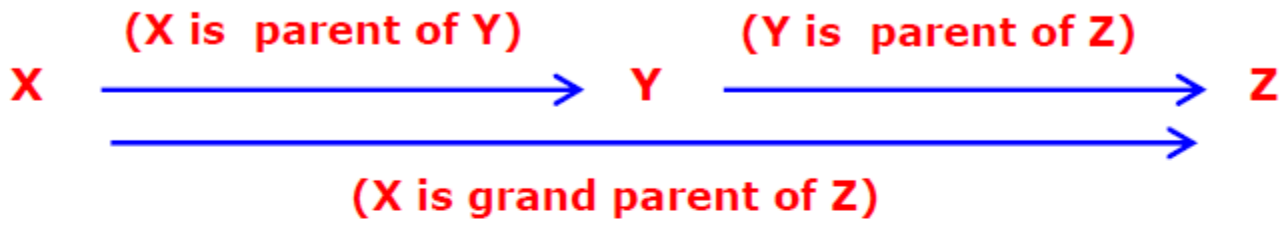
parent (X, Y) :- mother(X, Y).

parent (X, Y) :- father(X, Y).

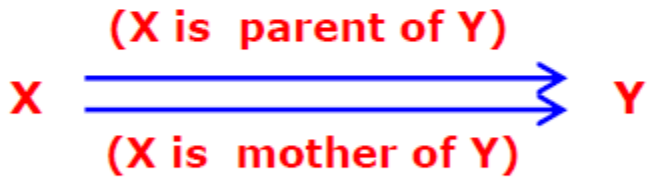
Read as if x is mother of y then x is parent of y

Interpretación

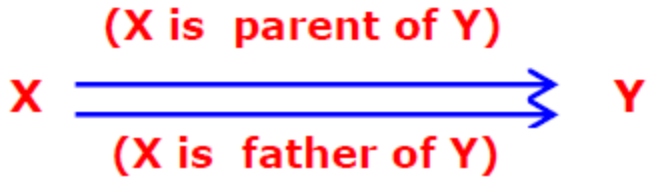
* An **individual "X"** is the **grand-parent of "Z"** if a **parent** of that same **"X"** is **"Y"** and **"Y"** is the **parent** of that **"Z".**



* An **individual "X"** is a **parent of "Y"** if **"Y" is the mother of "X"**



* An **individual "X"** is a **parent of "Y"** if **"Y" is the father of "X".**



Queries

- En Prolog los queries son sentencias llamadas directivas.

Syntactically, directives are clauses with an empty left-hand side.

Example : **? - grandparent(Q, Z).**

This query **Q** is interpreted as : **Who is a grandparent of Z ?**

By issuing queries **Q**, Prolog tries to establish the validity of specific relationships.

The answer from previous slides is **(X is grand parent of Z)**

Referencias

"Artificial Intelligence", by Elaine Rich and Kevin Knight, (2006), McGraw Hill companies Inc., Chapter 1-22, page 1-613.

"Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig, (2002), Prentice Hall, Chapter 1-27, page 1-1057.

"Computational Intelligence: A Logical Approach", by David Poole, Alan Mackworth, and Randy Goebel, (1998), Oxford University Press, Chapter 1-12, page 1-608.

"Artificial Intelligence: Structures and Strategies for Complex Problem Solving", by George F. Luger, (2002), Addison-Wesley, Chapter 1- 16, page 1-743.

"AI: A New Synthesis", by Nils J. Nilsson, (1998), Morgan Kaufmann Inc., Chapter 1-25, Page 1-493.

"Artificial Intelligence: Theory and Practice", by Thomas Dean, (1994), Addison-Wesley, Chapter 1-10, Page 1-650.

Related documents from open source, mainly internet. An exhaustive list is being prepared for inclusion at a later date.

Guión explicativo

- Esta presentación tiene como fin lo siguiente:
  - ¿Cómo representamos lo que conocemos?
  - Conocimiento y Representación
  - Conocimiento
  - Modelo de Conocimiento
  - Fuente de Conocimiento
  - Tipos de Conocimiento
  - Características del Conocimiento
  - Cuestiones de representación del Conocimiento
  - Lógica de Predicados
  - Reglas

Guión explicativo

- El contenido de esta presentación contiene temas de interés contenidos en la Unidad de Aprendizaje Inteligencia Artificial.

- Las diapositivas deben explicarse en orden, y deben revisarse aproximadamente en 24 horas, además de realizar preguntas a la clase sobre el contenido mostrado.