Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Resúmen I

Introducción a la Inteligencia Artificial

Autor:
Arroyo, Joaquín
Co-autor:
Bolzan, Francisco

2 de mayo de 2023

1. Introducción

1.1. ¿Qué es la IA?

Es la parte de las Ciencias de la Computación que se ocupa del diseño de sistemas inteligentes, esto es sistemas que exhiben características que asociamos con la inteligencia en las conductas humanas. Feigenbaum y Barr, 1980

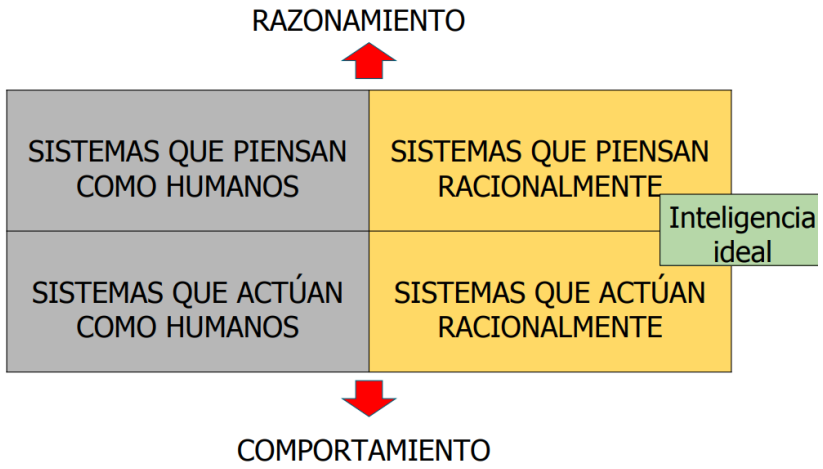
La rama de la Ciencias de la computación que se ocupa de la automatización de la conducta inteligente. Luger y Stubblefield, 1993

Es la Ciencia e Ingeniería de hacer máquinas inteligentes (especialmente programas). Esto está relacionado a la tarea de usar computadoras para entender la inteligencia humana, pero la IA no tiene que limitarse a métodos que son biológicamente observables. J. Mc Carthy, 1998

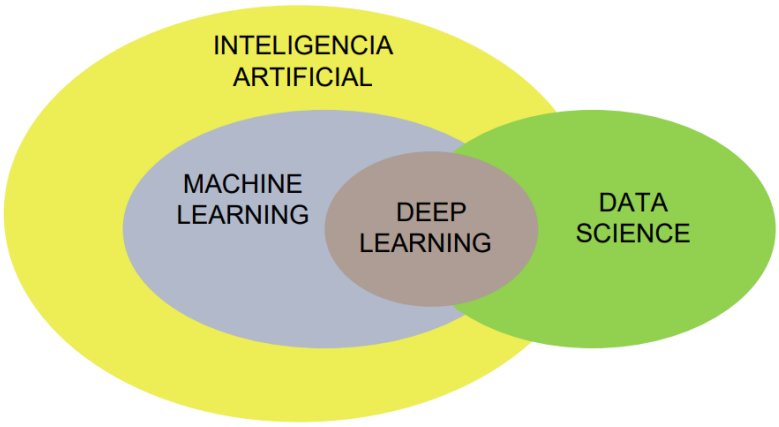
1.2. Modelos de IA

- Modelos cognitivos(Semánticos-Deductivos): Ciencia cognitiva, son transparentes al usuario, faciles de modificar/incrementar. Por ejemplo, sistemas basados en el conocimiento (KBS), herramientas semánticas y agentes deliberativos.
- Modelos conexionistas(Inductivos): Ciencia de Datos. Por ejemplo, Redes Neuronales, Deep Learning y Data Science.

La inteligencia ideal a conseguir, debe estar entre sistemas que piensen racionalmente, y sistemas que actúen racionalmente, teniendo en cuenta el siguiente cuadro:



La idea es que la IA debe simular el comportamiento humano, a nivel de procesos cognitivos, y construir programas inteligentes de la forma mas eficiente.



1.3. Test de Turing

Es una prueba propuesta por Alan Turing para evaluar si una máquina puede exhibir comportamiento inteligente indistinguible del comportamiento humano. Dicho test se realiza de la siguiente manera: un juez humano interactúa con una máquina y con un ser humano a través de un sistema de comunicación que oculta su identidad. El juez hace preguntas a ambos, y debe determinar cuál de los dos, la máquina o el ser humano, está respondiendo a las preguntas. Si la máquina logra engañar al juez y hacer que este crea que es el ser humano, entonces se considera que ha pasado el Test de Turing y se considera que tiene inteligencia similar a la humana.

Sin embargo, es importante tener en cuenta que el Test de Turing ha sido objeto de debate y críticas. Algunos argumentan que no es una prueba definitiva de la inteligencia de una máquina, ya que puede haber máquinas que sean capaces de engañar a un juez sin tener una verdadera inteligencia. Otros argumentan que el Test de Turing es un punto de referencia útil para evaluar la inteligencia de las máquinas, pero no es la única medida válida.

1.4. Agentes

Agente es todo aquello que percibe su ambiente mediante sensores y que responde o actúa mediante efectores.

Un Agente Inteligente, debe hacer siempre lo correcto de acuerdo a sus percepciones, y es quel que emprende la mejor acción posible frente a una situación dada.

En los sistemas Multi Agentes, cada agente tiene información y capacidades limitadas para resolver un problema. No hay un control global del sistema, los datos están descentralizados y la computación es asincrónica. Proveen mas robustez, eficiencia y permiten la interoperatividad de sistemas existentes

2. Búsqueda

2.1. Resolución de problemas

Se formulan los problemas mediante espacios de estados(Representación formal), y se aplican distintas técnicas de búsqueda(Secuencias de operadores sobre el espacio de estados).

Una solución es una secuencia de acciones(plan) que permite transformar el estado inicial, en un estado meta.

Para formular los problemas, se siguen los siguientes pasos:

- Establecer la meta
- Conjunto de estados (inicial, estructuras de datos)
- Acciones ($estado_i \to estado_j$)

Por ejemplo, **8-puzzle:**

- Estado inicial: S = $\{A \in M^{(3\times 3)}[\![0,8]\!]\}$
- Estado objetivo: G = $\begin{pmatrix} 1 & 2 & 3 \\ 8 & 0 & 4 \\ 7 & 6 & 5 \end{pmatrix}$
- Sea E un estado cualquiera, y sea $\alpha_{ij} = 0$

  $up(E) = \begin{cases} E, & \text{si i} = 1 \\ E', & \text{otro caso} \end{cases}$

  Donde $\alpha'_{ij} = \begin{cases} \alpha_{(i'-1)j'}, & \text{si } (i',j') = (i,j) \\ 0, & \text{si } (i',j') = (i-1,j) \\ \alpha_{i'j'}, & \text{en otro caso} \end{cases}$

  Luego $down(E)$, $right(E)$ y $left(E)$ son análogos.

Al representar el problema en un espacio de estados, obtenemos el Grafo de Espacios de Estados, este es una representación matemática de un problema de búsqueda.

- Sus nodos son configuraciones del mundo(abtracciones)
- Arcos representan sucesores(resultados de las acciones)
- El test de goal puede corresponderse a un conjunto de nodos meta, o ser uno solo

Al realizar una búsqueda sobre el espacio de estados, obtenemos el Árbol de búsqueda. Este es un árbol de planes 'what if' y sus salidas, el estado inicial está en el nodo raíz, y los hijos corresponden a los sucesores.

Las estrategias de búsqueda deben:

- Ocasionar cambios
- Ser sistemáticas, es necesario un cambio global, esto evita la reiteración de secuencias de operadores poco apropiadas.
- Ser eficientes, con respecto a completitud, y complejidad (más de tiempo que de espacio)

2.2. Búsqueda sin información

Distintas estrategias de búsqueda sin información:

- BFS: Agrega todas las expansiones al final de la cola (FIFO)

  Es óptimo y completo
  Tiempo $\mathcal{O}(b^d)$
  Espacio $\mathcal{O}(b^d)$
  Donde b es el factor de ramificación y d es la profundidad del arbol

- DFS: Agrega todas las expansiones al inicio de la cola (LIFO)

  No es óptimo y ni completo, puede atascarse en bucles
  Tiempo $\mathcal{O}(b^m)$
  Espacio $\mathcal{O}(b.m)$
  Donde m es la profundidad máxima del arbol

- USC: Expande el nodo con menor costo(g) de ruta asociado.

  Si $g(n) = profundidad(n) \implies BFS = UCS$

  Es óptimo y completo (Óptimo si el costo no disminuye)
  Tiempo $\mathcal{O}(b^d)$
  Espacio $\mathcal{O}(b^d)$

- Limitada por profundidad: DFS con límite(l) de profundidad fijo

  No es óptimo, y es completo cuando $l \geq d$
  Tiempo $\mathcal{O}(b^l)$
  Espacio $\mathcal{O}(b.l)$

- Profundización iterativa: DFS cambiando el límite de profundidad en cada iteración

  Es óptimo y completo
  Tiempo $\mathcal{O}(b^d)$
  Espacio $\mathcal{O}(b.d)$

2.3. Búsqueda con informacíon

El agente posee información sobre el problema, para poder elegir operadores más convenientes. Distintas estrategias de búsqueda con información:

- 1ero el mejor: Se expande el nodo de mejor valor en la función heurística $f(n)(min, max)$, esta función puede incorporar conocimiento de dominio.

  Si $f(n) = g(n)$ tenemos UCS
  Si $f(n) = h(n)$ tenemos Greedy Search.

- Greedy Search: Manteniendo toda la frontera, se expande el nodo de menor costo teniendo en cuenta solo $h(n)$. Si no usamos una buena heurística, entonces el algoritmo puede no ser óptimo ni completo. Es suceptible a dar pasos en falso

  Tiempo $\mathcal{O}(b^m)$
  Espacio $\mathcal{O}(b^m)$
  Donde m es la profundidad máxima del arbol.
  Si h es buena la complejidad temporal disminuye

- A^*: Se toma $f(n) = g(n) + h(n)$. Una heurística admisible nunca sobreestima el costo de llegar a la meta. Si h es admisible, f nunca sobreestima el costo real de la mejor solución pasando por n. Una búsqueda con h admisible es óptima y completa

  Si $h(n) = 0$ tenemos UCS, en este caso, se expande igualmente en todas las direcciones
  Si $h(n) > 0$, entonces se expande principalmente en dirección a la meta, pero se asegura de no perder optimalidad

Toda heurística consistente es admisible, pero no siempre vale la vuelta.
h es admisible $\Leftrightarrow \forall\ n$, $h(n) \leq c^*(n)$ donde (c^* es el costo de llegar al objetivo partiendo desde n)
h es consistente $\Leftrightarrow \forall\ n\ y\ sucesor\ n'$, $h(n) \leq c(n, n') + h(n')$ donde (c costo de n a n')
Si A^* utiliza una función $h(n)$ admisible, entonces siempre va a encontrar un nodo meta óptimo.
Para probar si h es admisible, o probamos directamente admisibilidad, o probamos consistencia.
A^* es óptimamente eficiente, ningun otro algoritmo expande menos nodos. Usualmente, este algoritmo se queda sin memoria antes de tiempo, por lo que existen variantes que vienen a solucionar este problema:

- A^*PI: Por búsqueda iterativa. Se mantienen solo nodos con $f < limite$, se itera dicho límite.
- A^*SRM: acotada por memoria simplificada. Se descartan nodos de f altos según requerimiento de memoria. Los valores descartados se memorizan en ancestros y se regeneran sii todas las demás rutas son peores.

2.4. Algoritmos de mejoramiento iterativo

Otros algoritmos de búsqueda con información, de mejoramiento iterativo:

- Hill Climbing: A partir de un estado, se realiza un bucle que constantemente se desplaza hacia dirección ascendente, hasta encontrar una solución o atascarse.

  No mantiene árbol de búsqueda, descarta información de ruta, y depende de la estructura de la 'superficie' del espacio de estados.

  Tiene problemas como, máximos locales, mesetas y crestas. Las posibles soluciones son, empezar con distintas configuraciones iniciales, backtracking, saltos, y aplicar varias reglas antes de evaluar.

- Endurecimiento simulado: Proceso de optimización global en sistemas de comportamiento estocástico, con alguna problabilidad que es la función 'temperatura' (va descendiendo) con lo cuál la conducta no es completamente determinista.

  Se elige un movimiento al azar, si es mejor se ejectua, caso contrario, se ejecuta con probabilidad que decrezca exponencialmente con lo 'malo' del movimiento y la 'temperatura'.

  La 'temperatura' cambia de acuerdo a un programa, si esta desciende de forma lenta, se va a encontrar un óptimo global.

2.5. Búsqueda mediante satisfacción de restricciones (CSP)

Los estados se definen mediante los valores de un conjunto de variables, y el objetivo se especifica mediante un conjunto de restriccipones que los valores deben satisfacer.

Solución → especificar valores para todas la varibales tal que satisfagan todas las restricciones.

Definición: (V, D, R), donde:

- V son las varibales $V = \{v_1, ..., v_n\}$
- D son los dominios de cada variable (continuos-discretos) $D = \{D_1, ..., D_n\}$
- R son las restricciones (de k-variables, binarias) (obligatorias-preferencia) $R = \{R_1, ..., R_k\}$ y $R_j \subset D_{j1} \times ... \times D_{jk}$

Cada restricción plantea relaciones entre los valores de una/algunas variables.

Complejidad: NP-Completos. Debemos reducir el espacio de búsqueda.

Algoritmos: Se pueden utilizar los vistos anteriormente, pero tienen una mayor eficiencia algunos diseñados especificamente para estos problemas, como lo son verificación anticipada, consistencia de arco, etc.

Un ejemplo de CSP:

Problema de las n-reinas:

Variables: V = $\{r_i | i \in [\![1, n]\!]\}$

Dominios: $r_i \in [\![1, n]\!]$

Restricciones:

- La restricción por columnas no hace falta, ya que tenemos una reina en cada columna.
- $r_i \neq r_j$ (restricción por fila)
- $|r_i - r_j| \neq |i - j|$ (restricción por diagonales)

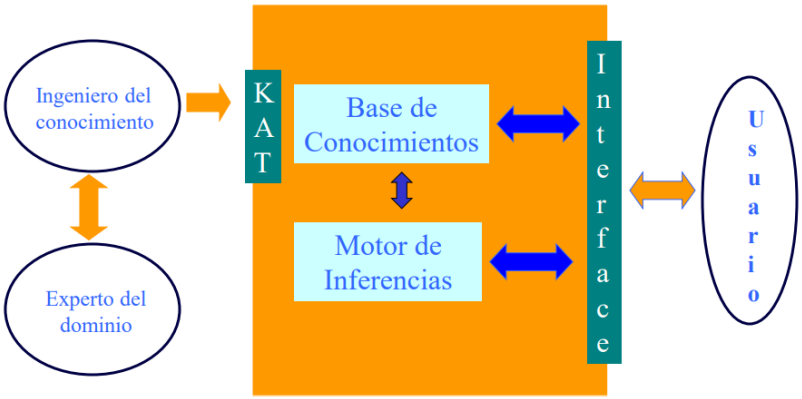
3. Ontologías

3.1. Introducción

Los Sistemas Basados en Conocimiento(SBC) son sistemas de software que mantienen una gran cantidad de conocimiento en Bases de Conocimiento(BDC) y que incluyen métodos para acceder a dicha información.

Un Sistema Experto, es un SBC que contiene el conocimiento utilizado por humanos expertos en algún dominio específico.

La estructura básica de un SE es la siguiente:



donde:

- La BDC queda definida en términos del esquema de representación elegido para incorporar el conocimiento de dominio.
- El Motor de Inferencias, es la estructura de control que contiene el programa que gestiona la BDC y otros mecanismos necesarios para administrar un sistema interactivo.
- La Ingeniería del Conocimiento, es eun conjunto de conocimiento y técnicas que permiten aplicar el saber científicop a la utilización del conocimiento. Se centra en el desarrollo, funcionamiento y mantenimiento de los SE.

Las ventajas de los SBC son las siguientes:

- El conocimiento no se pierde.
- Se reduce el espacio de búsqueda con heurísticas para que el problema sea tratable en un tiempo razonable.
- Posibilidad de justificar el razonamiento seguido.
- Hacer el conocimiento disponible en ambientes hostiles o con carencia de especialistas.
- Aumento de fiabilidad.

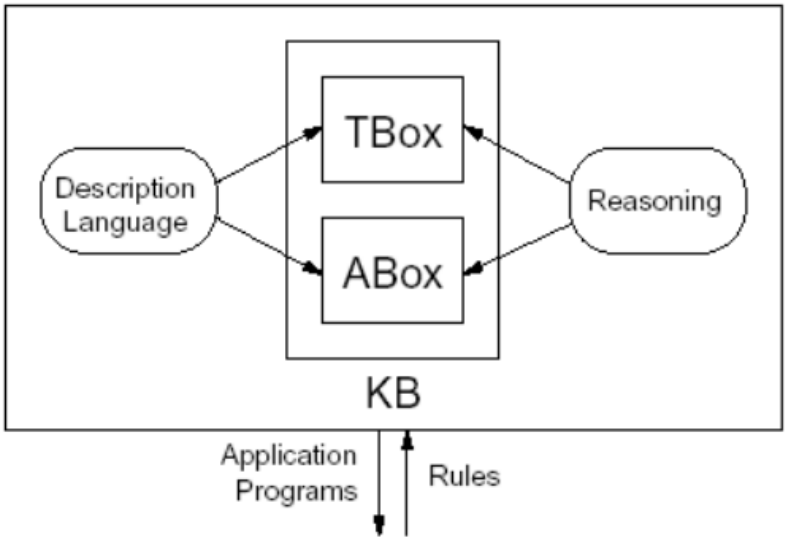
y sus inconvenientes son:

- La adquisición del conocimiento es dificil y cara. (Modelos inductivos/AA).
- La reutilización no es simple.
- Falta de creatividad y sentido común.
- Necesidad de aprendizaje y adaptación(AA).

Una BDC debe permitir con eficiencia: Añadir y modificar sentencias a ella, y responder preguntas.

Una Ontología es una especifiación explícita de una conceptualización. Una conceptualización es una abstracción, una vista simplificada del mundo que queremos representar. Provee una estructura para describir un dominio de la cual puede construirse una KB. En IA, es un vocabulario de representación especializado para algún dominio, es un cuerpo de conocimiento que describe algún dominio.

Están compuestas por, conceptos, propiedades, restricciones sobre propiedades y relaciones, las cuáles se ubican en la TBox o en la ABox.



3.2. Lógica Descriptiva(DL)

Son una familia de formalismos de representación del conocimiento. Definen los conceptos relevantes de un dominio y los relacionan para especificar propiedades.

Se diseñaron como una extensión de marcos y redes semánticas, los cuales carecían de semántica basada en lógica formal.

Conceptos básicos:

- Bloques de contrucción básicos: Predicados unarios, binarios y constantes.
- El poder expresivo del lenguaje está restringido a un conjunto pequeño de constructores para construir conceptos complejos y roles.
- El conocimiento implícito puede inferirse automáticamenmte con la ayuda de procedimientos de inferencia.

Los DL están compuestos por dos bases de conocimientos, la TBox y la ABox:

En la TBox se encuentran las sentencias describiendo conceptos jerárquicos (Cada empleado es una persona).

En la ABox se encuentran las sentencias que indican a donde pertenecen los individuos de la jerarquía (Bob es un empleado).

Lenguaje básico $\mathcal{AL}$:

- Sintaxis:

$$
\begin{array}{lll}
C, D & \longrightarrow & A \mid \quad \text{-concepto atómico} \\
 & & \top \mid \quad \text{-concepto universal} \\
 & & \bot \mid \quad \text{-concepto bottom} \\
 & & \neg A \mid \quad \text{-negación atómica} \\
 & & C \sqcap D \mid \quad \text{-intersección} \\
 & & \forall R.C \mid \quad \text{-restricción de valor} \\
 & & \exists R.\top \quad \text{-cuantificación existencial limitada}
\end{array}
$$

-Ejemplos:

Person ⊓ ¬Female	Personas no femeninas
Person ⊓ ∃hasChild.⊤	Personas que tienen hijos
Person ⊓ ∀hasChild.⊥	Personas que no tienen hijos
Person ⊓ ∀hasChild.Female	Personas que tienen solo hijas

La semántica es dada por medio de una interpretación I que consiste de un conjunto no vacío ΔI (el dominio de interpretación) y una función de interpretación que le asigna a todo concepto atómico A un conjunto $A^I \subseteq \Delta I$ y a cada rol atómico R una relación binaria $R^I \subseteq \Delta I \times \Delta I$

La función interpretación se puede extender a las descripciones de conceptos en forma inductiva.

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.\ (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in R^{\mathcal{I}}\}.
\end{aligned}$$

Dos conceptos C y D son equivalentes (C≡D) si $C^I = D^I$.

- La unión es interpretada como: $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$.
- La cuantificación existencial completa ∃R.C es interpretada como:

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$$

- Y las restricciones numéricas:

$$(\geqslant n\, R)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \;\middle|\; |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\right\},$$

$$(\leqslant n\, R)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \;\middle|\; |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\right\},$$

Los Axiomas Terminológicos establecen como los conceptos y roles se relacionan unos con otros. Sean C y D conceptos, R y S roles.

Axiomas de inclusión:

$$C \sqsubseteq D \quad (R \sqsubseteq S)$$

Axiomas de igualdades:

$$C \equiv D \quad (R \equiv S)$$

Una interpretación I satisface una inclusión $C \sqsubseteq D$ si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

$C \equiv D \qquad C^{\mathcal{I}} = D^{\mathcal{I}}.$

La semántica para ABox, esta dada por:

Sean a, b y c individuos, empleando conceptos como C y roles como R, entonces se pueden añadir a la ABox las siguientes afirmaciones:

- $C(a)$ (de conceptos)
- $R(b, c)$ (de rol)

La interpretación I mapea cada nombre individual a un elemento $a^I \in \Delta^I$.

Asumimos que nombres de individuos distintos denotan sujetos distintos. La interpretación I satisface el conecpto de afirmación $C(a)$ si $a^I \in C^I$ y satisface $R(b, c)$ si $(b^I, c^I) \in R^I$.

Si T es un conjunto de axiomas, I satisface T sii I satisface cada elemento de T.

Si I satisface un axioama decimos que es un modelo de ese axioma.

Una interpretación satisface ABox A si satisface cada afirmación de A.

Una ABox A es consistente respecto a una TBox T si hay una interpretación que es modelo de A y de T.

La diferencia entre Ontología y BDC es que una ontología provee la estructura básica alrededor de la cual una BDC puede construirse. Una ontología proporciona un conjunto de conceptos y términos para poder describir algún dominio, mientras que una BDC utiliza estos términos para representar lo que es verdadero sobre algún mundo hipotético o real.

Una ontología especifica un conjunto de restricciones que declaran lo que debería cumplirse necesariamente en cualquier mundoposible. Una descripción de un mundo legal es un mundo posible que satisface las restricciones. La interpretación de una ontología se define como la colección de todas las descripciones de mundos legales.

3.3. Representación y razonamiento

La ingeniería de Ontología representa el conocimiento de un dominio utilizando ontologías, esta:

- Define conceptos en el dominio (classes)
- Ordena los conceptos en una jerarquía (subclasssuperclass hierarchy)
- Define atributos y propiedades (slots) de las clases y las restricciones en sus valores
- Define individuos y completa sus valores de atributos-propiedades

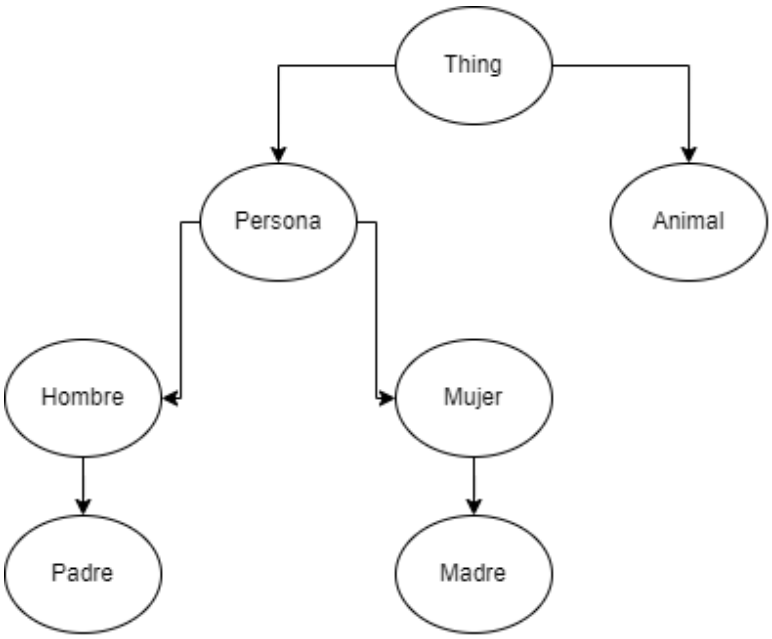
El paso a paso para desarrollar una Ontología:

- Determinar el dominio y alcance de la ontología
- Considerar la reutilización de ontologías existentes
- Enumerar términos importantes
- Definir las clases – jerarquías
- Definir las propiedades de las clases: slots
- Definir “constraints”/restricciones
- Crear Instancias

Las ’Competency Questions’ son las preguntas que pensamos que nuestra Ontología debe poder responder.

El razonamiento es importante porque, permite realizar chequeos de consistencia, detectar relaciones que estaban implícitas, realizar queries, etc.

Ejemplo de Ontologia:



Todas las flechas son *is_a*.

Persona y Animal son clases disjuntas.
Persona $\sqcap$ Animal $\equiv \emptyset$

Propiedades:
hasChild : Persona $\rightarrow$ Persona
isChild $\equiv$ inversa de hasChild
isDescendant : Persona $\rightarrow$ Persona (Transitiva)
hasPet : Persona $\rightarrow$ Animal
isPet $\equiv$ inversa de hasPet
isParent $\equiv$ Padre $\sqcup$ Madre
isBioMother : Madre $\rightarrow$ Persona (Funcional)
isRelated : Persona $\rightarrow$ Persona (Simétrica)

Clases definidas:
Padre $\equiv$ Hombre $\sqcap$ hasChild.$\top$
Hombre and hasChild min 1
Madre $\equiv$ Mujer $\sqcap$ hasChild.$\top$
Mujer and hasChild min 1

DL Queries:
isPerson $\equiv$ Persona
isFather $\equiv$ Padre
isMother $\equiv$ Madre
isGrandparent $\equiv$ Persona $\sqcap \exists$ hasChild.(hasChild.$\top$)
Persona and hasChild.hasChild
likesAnimals $\equiv$ Persona $\sqcap \exists$ hasPet.$\top$
Persona and (hasPet min 1)
isJuliaDescendent $\equiv$ Persona $\sqcap$ ($\exists$ isDescendant.{Julia})