Lógica de Primer Orden

Basado en:
Computability and Logic
Boolos and Jeffrey

- Sintaxis
  - Símbolos lógicos
  - Símbolos no-lógicos
- Interpretación
- Satisfacción
- Implicación
- Equivalencia lógica
- Ejemplos

- La madre o padre de una persona es su ancestro
- El ancestro de un ancestro de una persona es su ancestro
- Sarah es la madre de Issac, y Issac es el padre de Jacobo
- Entonces, Sarah es ancestro de Jacobo.

Usando las fórmulas

- $\forall x \forall y((Pyx \vee Qyx) \rightarrow Ryx)$
- $\forall x \forall y(\exists z(Ryz \,\&\, Rzx) \rightarrow Ryx)$
- Pab & Qbc
- Rac

Formalización

- Símbolos lógicos

$$\sim \;\&\; \vee \;\rightarrow\; \leftrightarrow\; \forall\; \exists\; =(\,)\;,$$

- Más variables:
  - $x, y, z \ldots$

Símbolos no lógicos

- Constantes: a, b, c …
- Predicados: P, Q, R
  - Tienen aridad
  - Se evalúan a Falso o Verdadero
- Funciones
  - Tienen aridad
  - Se evalúan a constantes

Lenguaje

- Un lenguaje es un conjunto enumerable de símbolos no lógicos
- Ejemplo: El lenguaje aritmético, L*
  - Constante: 0
  - Predicados: </2
  - Función: ´/1 (sucesor), +/2, */2

Lenguage G

- $\forall x \forall y((Pyx \lor Qyx) \rightarrow Ryx)$
- $\forall x \forall y(\exists z(Ryz \,\&\, Rzx) \rightarrow Ryx)$
- Pab & Qbc
- Rac
- Lenguaje L_G:
  - Constantes: a,b,c
  - Predicados: P, Q, R

Fórmulas y términos

- Fórmulas: secuencias de símbolos que corresponden a oraciones *bien formadas* en español
  - Abiertas y cerradas
- Términos: secuencias de símbolos que corresponden a frases *bien formadas* en español
  - Abiertas y cerradas

Fórmulas de un lenguaje

- Las fórmulas dado un lenguaje corresponde a las fórmulas cuyos símbolos no lógicos son parte del lenguaje dado
  - Lenguaje + símbolos lógicos
- ¿Qué hay del lenguaje vacio?
  - Tiene un número infinito de fórmulas: $\forall x\ x = x$
- Las fórmulas dado un lenguaje son enumerables

Todo es cuestión de interpretación

- El cuadrado o cubo de un número es un potencia de ese número.
- La potencia de una potencia de un número es una potencia de ese número.
- 64 es el cubo de 4 y 4 es el cuadrado de 2.
- Entonces, 64 es una potencia de 2.

Las mismas fórmulas

- $\forall x \forall y((Pyx \lor Qyx) \rightarrow Ryx)$
- $\forall x \forall y(\exists z(Ryz \,\&\, Rzx) \rightarrow Ryx)$
- Pab & Qbc
- Rac
- Mismo lenguaje (L_G):
  - Constantes: a,b,c
  - Predicados: P, Q, R

Interpretación

- Definir que representan variables, constantes, predicados y funciones
  - Primer caso: variables son personas, a es Sarah, b es Issac y c es Jacobo. P es una relación entre madre a hijo, Q padre a hijo y R ancestro a descendiente.
  - Segundo caso: variables son enteros, a es 64, b es 4 y c es 2. P es el cubo, Q es cuadrado, R potencia.

Interpretación *M*

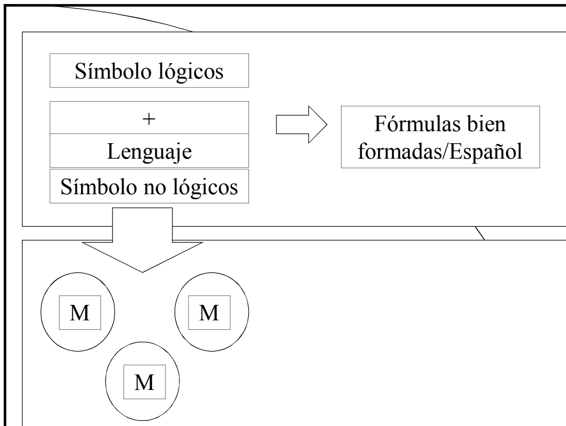
- Dominio o universo de discurso $|M|$
  - Todo x o
  - Todo x en *M*
- Denominación
  - Constantes C^M: un objeto en $|M|$
  - Predicados R^M: un conjunto de *n*-tuplas
  - Funciones F^M: una función de $|M|^n$ a $|M|$

Interpretación *G* para L_G

- Dominio $|G|$: todas las personas
- a^G Sarah, b^G Issac, c^G Jacobo
- P^G pares donde la primera persona es la madre de la segunda, similarmente Q^G y R^G
  - $\exists z(Pyz \,\&\, Qzx)$ "y es la abuela por padre de x"
  - $\sim\exists x\, Pxx$ "nadie es su propia madre"
  - $\exists x\, Qxx$ "alguien es su propio padre"

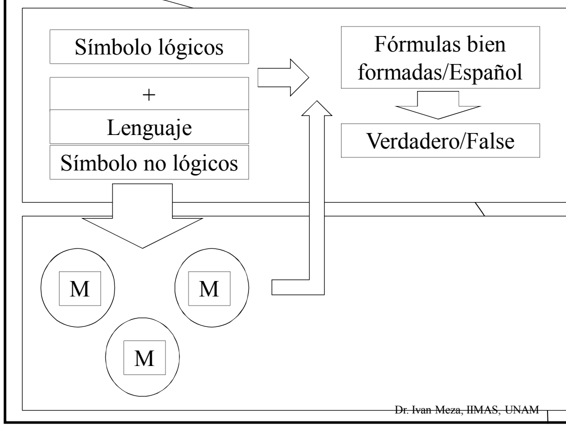
Interpretación *N** para *L**

- Dominio $|N^*|$: el conjunto de números naturales
- 0^{N^*} cero, $<^{N^*}$ menor que, $'^{N^*}$ función sucesor, $+^{N^*}$ suma, $*^{N^*}$ multiplicación
  1. $x * y$
  2. $0''$
  3. $\forall x \forall y(x \cdot y = 0 \rightarrow (x = 0 \vee y = 0))$
  4. $\forall x \exists y(x < y \,\&\, \sim\exists z(x < z \,\&\, z < y))$
  5. $\forall x(x < x' \,\&\, \sim\exists z(x < z \,\&\, z < x'))$



Valores de verdad

- Fórmula cerrada + una interpretación resulta en un valor verdadero o falso.
- El valor podrá ser diferente dependiendo de la interpretación
  - Cambiamos R^G de descendiente a ancestro
  - 1ª y 4ª fórmula son falsas
- No es posible encontrar interpretación que haga 1ª , 2ª y 3ª verdaderas pero la 4ª falsa.



Re-interpretación *N** para *L**

- Dominio |*N**|: racionales
  - 4 y 5 ambas falsas
- Dominio |*N**|: ½, 1, 3/2, 2, 5/2, …
  - 4 verdadera, 5 falsa
- No hay combinación que haga a 5 verdadera y 4 falsa
- Porque 4 es una consecuencia de 5

Términos bien formados

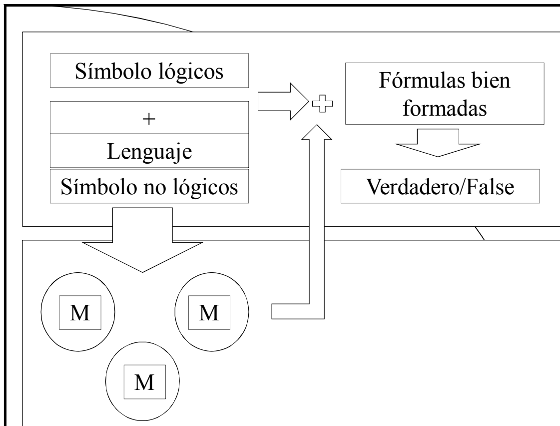
- Cualquier constante es un término: a,b,c
- Cualquier variable es un término: x,y,z
- Cualquier función con sus argumentos es un término: fxy, fyx,…
- Ninguna otra cosa es un término.

Fórmulas bien formadas

1. Si P es un predicado y $t_1,\ldots,t_n$ es una secuencia de n términos, entonces P $t_1,\ldots,t_n$ esta bien formada
2. Si F es una fórmula bien formada entonces ~F también lo es.
3. Si F y G son fórmulas bien formadas de Q, entonces (F & G), (F $\vee$ G), (F $\rightarrow$ G) , (F y $\leftrightarrow$G) y (F = G) también lo son.

Fórmulas bien formadas

4. Si F, es una fórmula bien formada entonces $\forall$x. F y $\exists$x F, también lo son
5. Sólo las expresiones que pueden ser generadas mediante las cláusulas 1 a 4 en un número finito de pasos son fórmulas bien formadas.



Tarski definition

- ¿Qué se necesita para que la fórmula F sea verdadera en la interpretación *M*?
  - *M* |= *F*
- Se lee: *M* hace verdadera a *F o*
  M satisface *F*

Denotacion en M

- c denota c^M
- $f(t_1, \ldots, t_n)$ denota $f^M(t^M_1,\ldots,t^M_n)$

M satisface a *F*

- M |= $R(t_1, \ldots, t_n)$ si y solo si $R^M(t^M_1,\ldots,t^M_n)$ lo es.
- M |= $t_1=t_2$ si y solo si $t^M_1=t^M_2$

Lenguaje de aritmética con *N**

- 0´ denota a 1
- 0´´ denota a 2
- 0´+0´´ denota a ??
- 0´´=0´´´´ es verdadera en *N*?*
- 0´+0´´<0´´´ es verdadera en *N*?*
- 0´+0´´=0´´´ es verdadera en *N*?*
- 0´+0

M satisface a *F (cont.)*

- M |= ~F si y solo si **no** *M* |= *F*
- M |= F&G si y solo si *M* |= *F* ***y*** *M* |= *G*
- M |= F∨G si y solo si *M* |= *F* ***o*** *M* |= *G*
- M |= F→G si y solo si *M* |= *G* ***si*** *M* |= *F*
- M |= F↔G si y solo si *M* |= *F* ***si y solo si*** *M* |= *G*

Lenguaje de aritmética con *N**

- 0=0 & 0<0´es verdadera en *N*?*
- 0=0 & 0<0 es verdadera en *N*?*
- 0=0 ∨ 0<0´es verdadera en *N*?*
- 0=0 ∨ 0<0 es verdadera en *N*?*
- 0=0 ∨ ~0<0 es verdadera en *N*?*

M satisface a *F (cont. 2)*

- M |= ∀xFx si y solo si para todo término *t* *M* |= *F (t)*
- M |= ∃xFx si y solo si existe un término *t* tal que *M* |= *F (t)*
- PROBLEMA no todos los elementos del domino tienen una representación como término

M hace verdadera a *F (cont. 3)*

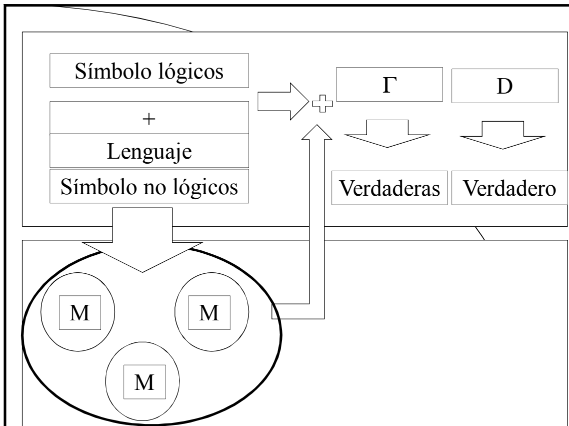
- M |= ∀xFx si y solo si para todo *m* en el dominio *M* |= *F [m]*
- M |= ∃xFx si y solo si existe *m* en el dominio tal que *t M* |= *F [m]*

Algunas implicaciones de las definiciones

- (F & G) es verdad si y solo si ~(~F ∨ ~G) es verdad
- (F ∨ G) es verdad si y solo si ~(~F & ~G) es verdad
- ∀xF es verdad si y solo si ~∃x ~ F es verdad
- ∃xF es verdad si y solo si ~∀x ~ F es verdad

Implicación

- ¿A todo esto a que llamamos implicación?
- Un conjunto de fórmulas lógicas Γ implica o tiene como consecuencia a la fórmula D si no hay interpretación que hace a cada fórmula en Γ verdadera pero D falsa.

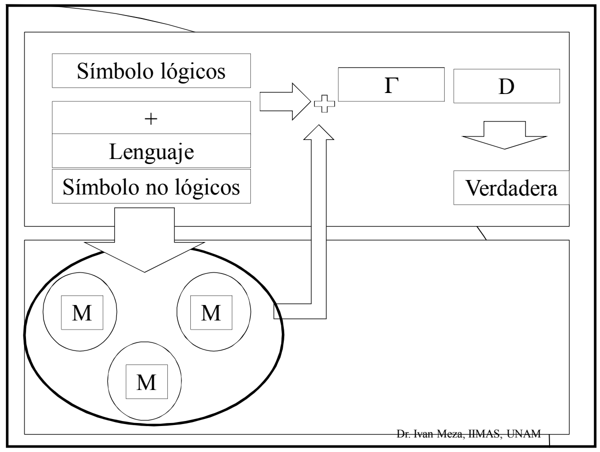


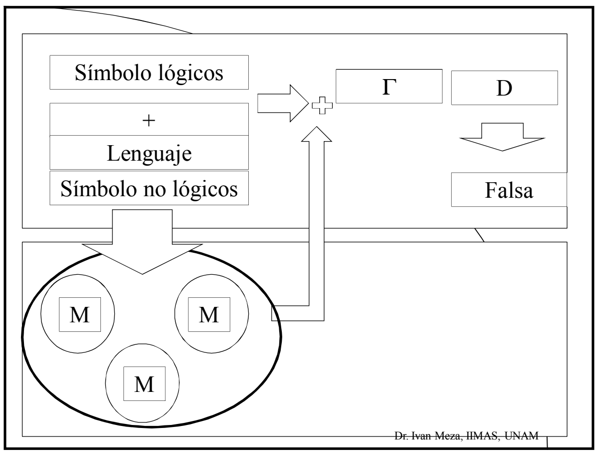
Ejemplos

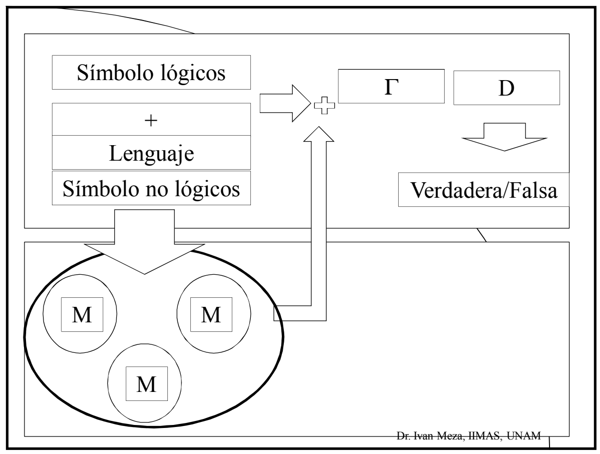
- ~~B implica B.
- B implica (B ∨ C) and C implica (B ∨ C).
- ~(B ∨ C) implica ~B and ~C.
- B(t) implica ∃x B(x).
- ~∃x B(x) implica ~B(t).
- s = t and B(s) implica B(t).

Más nociones meta-lógicas

- D es **valida** si no existe interpretación que la haga falsa
  - Γ implica D para cualquier Γ
- Γ es ***insatisfacible*** si no existe interpretación que la haga verdadera
- Γ es ***satisfacible*** si existe una interpretación que la haga verdadera







Equivalencia lógica

- Dos fórmulas son equivalentes para una interpretación *M* si tienen el mismo valor de verdad.
- Dos fórmulas son lógicamente equivalentes si tienen el mismo valor de verdad para todas las interpretaciones.

Teorías

- Conjunto de fórmulas (axiomas) en LPO
  - Ordenes, latices, grafos, algebras booleanas…
- Una interpretación para la teoría es un modelo de la teoría
- **Consistencia**: No hay contradicción de los axiomas de la teoría
- **Completa:** Cada axioma es un implicación del resto de axiomas

Teorema de incompletes de Gödel

- Teorías de LPO que contienen una porción considerable de la teoría de los números naturales no pueden ser consistentes y completas al mismo tiempo.

Sistema deductivo

- Es usado para demostrar usando únicamente sintaxis que una fórmula es una consecuencia de otra fórmulas
- Un sistema deductivo es:
  - correcto si todas las consecuencias derivadas son válidas
  - completo si todas las derivaciones son derivables
  - Efectivo si es posible verificar que una deducción es de hecho posible

Gödel otra vez

- LPO es indecidible
- No existe un proceso de decisión que determine sí una fórmula es lógicamente válida
- Sin embargo, implicación lógica es semi-decidible!!

  F |= G

  - Enumeramos todas las interpretaciones de F o derivamos todas las implicaciones de F hasta encontrar G

Reglas de inferencia

- Modus ponens

$$\frac{\alpha, \alpha \rightarrow \beta}{\beta}$$

Reglas de inferencia

- Introducción &

$$\frac{\alpha, \beta}{\alpha \& \beta}$$

Reglas de inferencia

- Eliminación universal

$$\frac{\forall x \alpha}{\alpha\{x / \tau\}}$$

Una deducción

- Bob es un búfalo
- Pat es un cerdo
- Los búfalos son más rápidos que los cerdos
- Introducción &
- Eliminación universal
- Modus ponens

- Bb
- Cp
- $\forall x \forall y Bx \& Cy \rightarrow Fxy$
- Bb & Cp
- $Bb \& Cp \rightarrow Fbp$
- Fbp

¿Qué podemos preguntar?

- ¿Una fórmula es verdadera dada una interpretación?
- ¿Estas fórmulas implican a esta otra dada una interpretación?
- ¿Qué interpretación hace verdadera está fórmula?
- ¿Esta formula es válida?
- ¿Esta formulas implican a esta otra?
- ¿Qué formulas implican estás fórmulas?

Unificación

- Una substitución σ unifica a p y q si $p\sigma=q\sigma$
  - Kjx U Kjt = Kjt donde σ=x/t
  - Kjx U Kyt = Kjt donde σ=x/t,y/j
- Idea: Si P unifica con un hecho P' con subtitución σ, entonces aplicar unificación a formulas del tipo $P\sigma \rightarrow Q\sigma$
  - Pa, $Px \rightarrow Qx$, $Pa \rightarrow Qa$

Modus Ponens Generalizado

$$\frac{p'_1, p'_2, \ldots, p'_n, (p_1 \,\&\, p_2 \,\&\ldots\&\, p_n \rightarrow \beta)}{\beta\sigma}$$

- Donde

$$p'_i = p_i\sigma$$

Ejemplo

- Fbp: Faster(bob,pat)
- Fps: Faster(pat,steven)
- $Fxy \& Fyz \rightarrow Fxz$
- σ=x/b,y/p,z/s
- Fbs: Faster(bob,steven)

Forward chaining

- Base de conocimiento = Axiomas que sabemos
- Algoritmo:
  - Cuando un nuevo hecho P aparezca
    - Para cada regla que unifique con P en la premisa
      - Si las otras premisas son conocidas entonces agregar la conclusión en la base de conocimiento

Ejemplo

Bufalo, Pig, Slug, Faster

- Bx&Py→Fxy
- Py&Sz →Fyz
- Fxy&Fyz →Fxz
- Bb (Bb&Py →Fby)
- Pp (Bb&Pp →Fbp, Pp&Sz →Fpz)
- Fbp (Fbp&Fpz →Fbz, Fxb&Fbp →Fxp)
- Ss (Pp&Ss→Fps)
- Fps (Fbp&Fps →Fbs, Fps&Fsz →Fpz, Fxp&Fps →Fxs)
- Fbs (Fbs&Fsz →Fbz, Fxb&Fbs →Fxs)

Backward Chaining

- Algoritmo:
  - Cuando alguien pregunta Q
    - Si el hecho Q' unifica con Q regresar Q'
    - Para cada formula cuyo consecuente Q' unifique con Q intentar probar cada una de las premisas usando backward Chaining
- Esta es la base de programación lógica, esto quiere decir Prolog

Ejemplo

Pig, Slug, Faster, Slimy, Creeps

- Py&Sz→Fyz
- Lz&Cz →Sz
- Pp
- Ls
- Cs
- Fps ? (Pp&Ss ? →Fps)
- Pp ?
- Ss ? (Ls&Cs ? → Ss)
- Ls
- Cs

Buenas y malas

- ForC y BckC son completas…
  - Pero solo para Horn clauses (Ps→Q)
  - PhD(x) →AltamenteCalificado(x)
  - ~PhD(x) →DineroPronto(x)
  - AltamenteCalificado(x) →Millonario(x)
  - DineroPronto(x) →Millonario(x)
- ¿Entonces por qué BckC y FckC no concluyen Millonario(ivan)?

Truco: Resolución

- BckC y ForC son semidecidibles:
  - Quiero BC |= α, si esto sucede en algún momento obtengo α
  - Pero si en realidad esto no sucede algunas veces dirá que sí y otras que nó
- Pero que tal si en lugar trato de mostrar:
  - BC & ~ α es insatisfacible !!!

Otras lógicas

- Lógica de *primer* orden
  - ¿Si hay primer, hay segundo, o zero?
- Lógica proposicional
  - ~ & ∨ → ↔
- Lógica de segundo orden
  - ~ & ∨ → ↔ ∀ ∃ = ( ) con ∀ ∃ para predicados
- Lógica modal
  - ~ & ∨ → ↔ ∀ ∃ = ( ) <> []