Proyecto 2
La no decibilidad de la Lógica de Predicados
Lógica Computacional 2019-2

Profesor: Fernando Abigail Galicia Mendoza
Ayudante: Leonardo Hernández Cano
Laboratorista: Francisco Emmanuel Anaya González

Licenciatura en Ciencias de la Computación
Facultad de Ciencias, UNAM

Hernández Leyva Mirén Jessamyn
Ruiz López Jorge Antonio

Marzo 2019

1. Introducción

Antes de comenzar a estudiar a profundidad éste tema necesitamos realizarnos la siguiente pregunta ¿por qué la lógica de predicados es necesaria si ya contamos con la lógica proposicional?

Ej: Queremos formalizar la siguiente expresión, es decir, pasar del español a lógica proposicional:

"A algunos pacientes les caen bien todos los doctores. A ningún paciente le cae bien una enfermera. Por lo tanto ningún doctor es enfermera".

La intuicón nos dice que el argumento es correcto. Y su representación sería la que sigue:

$p, q /= r$

Lo cual es incorrecto, pues la conclusión no es consecuencia lógica de las premisas.

Así podemos notar qué no es posible concluir un argumento del leguaje natural con lógica propisicional, pero esto no significa que este sea incorrecto.

Concluimos que la lógica propisicional carece de expresibilidad, por tanto necesitamos un lenguaje de especificación formal más poderoso, es aqui cuando entra la lógica de predicados.

En lógica de predicados nuestro ejemplo se expresaria de la siguiente manera:

- Predicados:
P(x) = X es paciente
D(x) = X es doctor
E(x) = X es enfermera
B(x,y) = a X le cae bien Y

$\exists x \forall y \ P(x) \land D(y) \land B(x, y)$
$\forall x, \forall y \ P(x) \land E(y) \rightarrow \neg(B(x, y))$

$\forall x \ D(x) \rightarrow \neg E(x)$

Podemos notar varios elementos nuevos que no encontramos en la lógica proposicional:

- Los cuantificadores $(\forall, \exists)$
- Los predicados (ej. P(x), D(x), E(x), ...)

Ahora, hablemos de los cuantificadores y el por que hacen a la lógica de predicados indecidible por medio del siguiente ejemplo:

Ej. Queremos verificar la siguiente propiedad sobre listas usando la LPO. Para cualesquiera listas l_1, l_2 que almacenan enteros, se cumple:

$$length(l_1 + + l_2) = length(l_1) + length(l_2)$$

Supongamos que tenemos un un intérprete para fórmulas de LPO e implementado el modelo del universo de listas que almacenan enteros revisara caso por caso que la propiedad se cumpliese. Y que tomamos la siguientes funcines: $length(x)$ la cual devuelve la longitud de la lista x y $suma(x, y)$ la cual nos devuelve la suma de enteros. Metemos el la siguiente formula

$$\forall x, y(length(suma(x, y)) = suma(length(x), length(y))$$

Tenemos el cuantificador universal, esto quiere decir que la propiedad se cumple para todo elemento del modelo. Intuitivamente al ser un modelo de números enteros, y al ser estos infinitos, nuestro interprete iría revisando caso a caso (numéro a numero) y nunca terminaria y por tanto es indecidible.

2. Lógica de Predicados

i. Sintaxis

Vamos a darle formalidad a las LPO y cual es su sintaxis: dependiendo de la estructura semántica que tengamos será necesario agregar símbolos particulares para denotar objetos y relaciones entre objetos. De esta manera el alfabeto consta de dos partes ajenas entre sí:

Parte común:

- Variables: $Var = x_1, ..., x_n, ...$
- Constantes lógicas: $\top, \bot$
- Conectivos lógicos: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Cuantificadores: $\forall, \exists$

- Simbolos auxiliares: (,) y ,
- Simbolo de igualdad: $=$

Signatura de un lenguaje:

$$\mathcal{L} = \mathcal{P} \cup \mathcal{F} \cup \mathcal{C}$$

- El conjunto de los predicados: $\mathcal{P} = P_1, P_2, ...P_n, ...$ donde cada $P_i^{(m)}$ tiene su respectivo indice (número de argumentos m).
- El conjunto de las funciones $\mathcal{F} = f_1, ..., f_n$ donde cada $f_i^{(m)}$ tiene su respectivo indice (número de argumentos m).
- El conjunto de las contantes $\mathcal{C} = c_1, c_2, ..., c_n, ...$

Por otro lado tenemos al conjunto de los terminos los cuales son aquellas expresiones que representarán objetos en la semántica y su definición. Denotado: $TERM_{\mathcal{L}}$

Se define recursivamente como:

$$t ::= x|c|f(t_1, ..., t_m)$$

- Las constantes son terminos $(c_1, ..., c_n, ...)$.
- Las variables son terminos $(x_1, ..., x_n, ...)$.
- Si $f^{(m)}$ es un símbolo de función y $t_1, ..., t_m$ son términos entonces $f(t_1, ..., t_m)$ es un término.
- Son todas.

Despues tenemos a las formulas atomicas ($ATOM_{\mathcal{L}}$), dadas por:

- Las constantes lógicas: $\top, \bot$
- Las expresiones de la forma $P_1(t_1, ..., t_n)$ donde $t_1, ..., t_n$ son términos.
- Las expresiones de la forma $t_1 = t_2$ si el lenguaje cuenta con igualdad.

Y por ultimo tenemos al conjunto de las formulas de expresiones compuestas aceptadas en un lenguaje $\mathcal{L}$
Denotado: $FORM_{\mathcal{L}}$

Se define recursivamente como:

- Si $\varphi \in ATOM_{\mathcal{L}}$ entonces $\varphi \in FORM_{\mathcal{L}}$.
- Si $\varphi \in FORM_{\mathcal{L}}$ entonces $(\neg\varphi) \in FORM_{\mathcal{L}}$.
- Si $\varphi, \psi \in FORM_{\mathcal{L}}$ entonces $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi) \in FORM_{\mathcal{L}}$
- Son todas.

Los cuantificadores se aplican a la mínima expresión sintácticamente posible. De manera que la precedencia se sigué de la siguiente forma:

$$\forall x\varphi \rightarrow \psi \text{ es } (\forall x\varphi) \rightarrow \psi$$
$$\exists y\varphi \wedge \forall w\psi \rightarrow X \text{ es } (\exists y\varphi) \wedge (\forall w\psi) \rightarrow X$$

Una de las mayores razones para usar lógica de predicados o cualquier otro lenguaje es poder demostrar propiedades, por lo que nos interesa formalizar esto con la sintaxis de LPO, para esto se tiene Principio de Inducción Estructural para $TERM_{\mathcal{L}}$ y el Principio de Inducción Estructural para $FORM_{\mathcal{L}}$, los cuales se definen como sigue:

Principio de Inducción Estructural sobre $TERM_{\mathcal{L}}$

Sea $\mathcal{P}$ una propiedad acerca de términos.

- Base de Inducción:
  - Si $x \in Var$ entonces $\mathcal{P}$ es válida para x
  - Si $c \in \mathcal{C}$ entonces $\mathcal{P}$ es válida para c.
- Hipótesis de inducción: suponer que se cumple $\mathcal{P}$ para cuales quiera terminos en $TERM_{\mathcal{L}}$.
- Paso Inductivo: mostrar que si $f \in \mathcal{F}$ es un símbolo de función de índice n entonces $f(t_1, ..., t_n)$ cumple $\mathcal{P}$.

Principio de Inducción Estructural para $FORM_{\mathcal{L}}$

Sea $\mathcal{P}$ una propiedad acerca de términos.

- Caso base: mostrar que toda fórmula atómica tiene la propiedad $\mathcal{P}$
- Hipótesis de inducción: suponer que φ y *psi* cumplen $\mathcal{P}$.
- Paso inductivo: Mostrar que:
  - $(\neg\varphi)$ tambien lo cumple $\mathcal{P}$.
  - $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$ tambien cumplen $\mathcal{P}$.

Alcance de los cuantificadores

Al definir nuestro lenguaje, podemos notar que nuestras fórmulas tenemos cuantificadores seguidos de una variable que será la que se está afectando en los predicados. Dicho esto necesitamos dar una definición de el *alcance o ligado* que tiene un cuatificador sobre una variable (para poder saber en que predicado lo estámos afectando).

El alcance de un cuantificador se define como sigue:

Dada una cuantificación $\forall x\varphi$ o $\exists x\varphi$, la presencia de x en los cuantificadores es la variable que liga al cuantificador, es decir, la variable que va a ser "afectada", mientras que la fórmula φ se llama *alcance, ámbito o radio de acción* de dicho cuantificador.

Una vez dicho lo anterior podemos definir lo que es una *variable ligada* y lo que es una *variable libre*.

Una *variable libre* podemos definirla de la siguiente manera:

$$fv\left(\forall x\varphi\right) = fv\left(\varphi\right) \setminus \{x\}$$

Es decir, las *variables libres* de una fórmula (cuantificada) podemos obtenerlas como las variables de la fórmula φ (la fórmula de alcance) menos las apariciones de la variable que aparece precedida del cuantificador. Una

variable ligada es aquella que no es *libre*.

Notación:

- fv variables libres.
- bv variables ligadas.

.

Ej: Obtendremos las variables libres y ligadas de la siguiente fórmula.

$$\forall x \left(P\left(x, y\right) \rightarrow Q\left(f\left(x\right)\right)\right) \wedge Q\left(a\right) \rightarrow \exists z R\left(y, z\right)$$

Variables libres: Nos fijamos en las apariciones de las variables de cada cuantificador.

- $\forall x \Longrightarrow \left(P\left(x, y\right) \rightarrow Q\left(f\left(x\right)\right)\right)$
- $\exists z \Longrightarrow \left(R\left(y, z\right)\right)$

Ya que vimos cuales son las variables afectadas por cada cuantificador a nuestra fórmula original "quitamos" cada una de esas variables y las restantes serán las variables libres en φ. Entonces tenemos que las variables libres de φ son: y en la primer parte, a(puede ser cte.) y y en la segunda parte. Así, $fv(\varphi) = y$. Esto es porque a es una cte.

Variables ligadas: Sólo nos fijamos en las apariciones de las variables de cada cuantificador.

- $\forall x \Longrightarrow \left(P\left(x, y\right) \rightarrow Q\left(f\left(x\right)\right)\right)$
- $\exists z \Longrightarrow \left(R\left(y, z\right)\right)$

Entonces $bv(\varphi) = x, z$.

Sustitución

Recordemos que en la logíca proposicional teníamos una herramienta muy útil y es que podíamos realizar una sustitución textual al momento de formalizar. Ahora veamos que es lo que sucede en la lógica de primer orden al intentar realizar una sustitución sobre la fórmula siguiente:

$$(\exists x Q\,(x, y))\,[x := f\,(y)]$$

Primero, como idea intuitiva diremos que la sustitución significa cambiar los elementos de la izquierda por los de la derecha de la igualdad ($[x := f\,(y)]$ como aquí.), aunque daremos una definición formal más adelante.

De la sustitución sobre la fórmula deberíamos obtener:

$$(\exists f\,(y)\,Q\,(f\,(y)\,, y))\,[x := f\,(y)]$$ ☞

Pero tenemos un problema, la variable sobre la que estamos tratando de relizar la sustitución es una variable ligada y no podemos afectar a variables que no sean "ajenas" a la fórmula sobre la que estemos aplicando la sustitución.

Además de no ser una fórmula válida en LPO ya que los cuantificadores solo pueden ser aplicados a variables.

Ahora después de ver la dea intuitiva en el ejemplo anterior, vamos a definir la sustitución sobre los términos y sobre las fórmulas.

Sustitución sobre términos:

La sustitución $[\vec{x} := \vec{t}]$ a un término r se denota como $r[\vec{x} := \vec{t}]$ se obtiene de reemplazar simultáneamente todas las presencias de x_i en r por t_i. Se define recursivamente como sigue:

$$x_i[\vec{x} := \vec{t}] = t_i \text{ si } 1 \leq i \leq n$$
$$z[\vec{x} := \vec{t}] = z \text{ si } z \neq x_i \text{ y } 1 \leq i \leq n$$
$$c[\vec{x} := \vec{t}] = c \text{ si } c \in C \text{ i.e. c es cte.}$$
$$f(t_1, ...t_m)[\vec{x} := \vec{t}] = f(t_1[\vec{x} := \vec{t}], ..., t_n[\vec{x} := \vec{t}]) \text{ con } f^{(m)} \in F$$

Entonces la sustitución de términos es simplemente una sustitución textual como en la lógica porposicional.

Sustitución sobre fórmulas:

Para definir la sustitución sobre una fórmula intentaríamos hacer lo mismo qe con los términos como sigue:

$$(\forall y\varphi)[\vec{x} := \vec{t}] = \forall(y\varphi[\vec{x} := \vec{t}])(\varphi[\vec{x} := \vec{t}])$$

Pero al realizar la sustitución textual obtenemos algo como esto:

$$(\forall x P(y, f(x)))[x, y := g(y), z] = \forall g(y) P(z, f(g(y)))$$

Pero tenemos un problema y como se mencionó antes, la cuantificación $(\exists, \forall)$ sobre los términos está solamente para variables. Entonces la sustitución en formulas no debe ser una sistitución textual como en términos. Aunque una solución rápida que podríamos encontrar es definirla solamente para variables que estén libres como sigue:

$$(\forall x\varphi)[\vec{x} := \vec{t}] = \forall x(\varphi[\vec{x} := \vec{t}]_x)$$

donde $[\vec{x} := \vec{t}]_x$ denota la sustitución que elimina al par $x := t$ de $[\vec{x} := \vec{t}]$

Ej: $[z, y, x := y, f(x), c]_x = [z, y := y, f(x)]$

$(\forall x P(y, f(x)))[x, y := g(y), z]$

$= \forall x(P(y, f(x)[x, y := g(y), z]_x)$

$= \forall x(P(y, f(x)[y := z])$

$= \forall x P(z, f(x))$

Así podemos garantizar que siempre obtendremos una fórmula.

Captura de variales libres:

$$\forall x(\exists y(x \neq y))$$

Al momento de calcular $(\exists y(x \neq y)[x := y]$ obtendremos:

$$(\exists y(x \neq y)[x := y] = \exists y((x \neq y)[x := y]_y) = \exists y((x \neq y)[x := y]) = \exists y(y \neq y)$$

De manera que con esta definición no podemos ser capaces de garantizar la verdad de una fórmula φ en algún caso en particular.

Ahora veamos como se realiza la sustitución después de haberla definido formalmente.

Ej:
1. $(\exists x Q\,(x, y))\,[x := f\,(y)]$

$= \exists f(y) Q(f(y), y)$

2. $(\forall y(Q(y) \to R(z, y)[y := f(w)]))[z, w := f(x), a]$

$= \forall f(w)(Q(f(w)) \to R(z, f(w)))[z, w := f(x), a]$

$= \forall f(a)(Q(f(a)) \to R(f(x), f(a)))$

α-equivalencia

Como pudimos notar en los ejemplos anteriores no siempre podemos realizar una sustitución, en especal cuando una de las variables a sustituir es una variable ligada y para esto podemos definir una nueva función llamada $\alpha - equivalencia$.

Primero recordemos que una función es parcial si no esta definida para todo su dominio. La sustitucion es una función parcial pues a la hora de usar cuantificadores ocurren cosas como la siguiente:

$$\forall p(R(q \wedge S(p,q)))[q := l(p)]$$

donde $p \in Var(l(p))$, lo cual hace que no se pueda aplicar la sustitución pues no es una variable ajena a la formula.

Sin embargo esto se puede solucionar ya que las variables ligadas no importan pues al cambiarles el nombre siguen significando lo mismo, a este renombramiento se le llama formalmente $\alpha - equivalencia$ y se define como sigue:

Decimos que dos fórmulas φ_1, φ_2 son α-equivalentes lo cual escribimos $\varphi_1 \sim_\alpha \varphi_2$ si y sólo si φ_1 y φ_2 difieren a lo más en los nombres de sus variables ligadas.

Ej: A continuación mostramos una formula equivalente a

$$\varphi_1 = \forall x \exists y(P(x,y) \to \forall w P(z, f(w)))$$

Las variables ligadas de esta fórmula son: x, y, z, y tomando la siguiente fórmula:

$$\varphi_2 = \forall n \exists m(P(n,m) \to \forall s P(s, f(w)))$$

Podemos notar que en esta las variables libres son: n, m y s donde x es n, y es m y s es z si comparamos con la primera. Por lo tanto que φ_1 y φ_2 difieren a lo más en los nombres de sus variables ligadas, por lo que son α-equivalentes.

Con esto, intituitivamente podemos notar que al momento de aplicar internamente la α-equivalencia sobre una fórmula estamos evitando la captura de variables libres y todas las variables ligadas cambian de nombre,

por lo que la función se vuelve total, pues se puede tomar cualquier formula y usando las α-equivalencias necesarias y aplicando la sustitución ya esta definida para todo su dominio.

Ej. Calularemos la siguiente sustitución:

$$\sigma = [x, z, y := z, f(y), b]$$

sobre la fórmula:

$$\neg\exists x(R(x, f(b)) \land \exists w(R(w, z) \land (\forall z(T(z) \to P(x, y, z)))))$$

Como podemos notar si aplicamos directamente la sutitución algunas variables ligadas saldrian afectadas pues tienen el mismo nombre que las que vamos sustituir, lo que haría que la formula perdiera su significado inicial. Aquí es cuando aplicamos la α-equivalencia (donde ahora z sera r) para evitar esto y nos resulta:

$$\neg\exists x(R(x, f(b)) \land \exists w(R(w, z) \land (\forall r(T(r) \to P(x, y, r)))))$$

Y ahora si procedemos a aplicar la sustitución, y optenemos:

$$\neg\exists z(R(z, f(b)) \land \exists w(R(w, f(y)) \land (\forall y(T(f(y)) \to P(z, b, f(y))))))$$

Semantica

Después de aplicar la sustitución a una fórmula hay más cuestiones que podemos preguntarnos y algo que puede interesarnos es la veracidad que puede o no tener una ffomula y a continuación darémos una idea intuitiva de como se determina si una fórmula es verdadera con el siguiente ejemplo: Trabajaremos sobre el mundo de los cubos, con los predicados y formulas definidos en la nota 6 y la siguiente formula:

(Hay un cubo verde sobre un cubo verde)
$$\exists x\exists y(V(x) \land V(y) \land S(x, y))$$

Sabemos que $\mathcal{I}(x)(V(x) \to L(x)) = 1$ pues tenemos por lo que estar libre quiere decir que ningun cubo esta sobre el, según la definición del predicado, por lo que en la formula a analizar $\mathcal{I}(x)(\exists x\exists y(V(x) \land V(y) \land S(x, y)) = 0$, pues existe un cubo verde y que esta sobre el cubo verde x, lo cual contradice que $\forall\, V(x) \to L(x)$, que como antes mencionamos se evalua a 1, así llegamos a una contradicción y sabemos que la formula es falsa, por lo tanto no es verdadera.

Interpretación de terminos

Ahora para formalizar la interpretación de terminos comenzaremos definiendo la interpretación de terminos con estos dos conceptos basicos:

Estado de la variable:

$$\sigma : \text{Var} \to M$$

Actualización de Estados:

$$\sigma[\vec{x}/\vec{m}](y) = \begin{cases} \sigma(y) \text{ si } y \notin \{x_1, ..., x_n\} \\ \\ m_i \text{ si } y = x_i 1 \leq i \leq n \end{cases}$$

Y ahora si la interpretación de terminos:
Sea σ un estado de las variables. Definimos la interpretación de terminos bajo σ como, $\mathcal{I}_\sigma : \text{TERM} \to |\mathcal{M}|$

$$\begin{gathered} \mathcal{I}_\sigma(x) = \sigma(x) \\ \mathcal{I}_\sigma(c) = \mathcal{I}(c) \\ \mathcal{I}_\sigma(f(t_1, ...t_n)) = f^{\mathcal{I}}(\mathcal{I}_\sigma(t_1), ..., \mathcal{I}_\sigma(t_n) \end{gathered}$$

Ej: Calcularemos la interpretación del siguiente termino $prod(8, sum(3, 4))$
Comenzamos dando la asignatura del lenguaje:

$prod = f^2(x_1, x_2)$
$f(2)x = \times^2$
$sum = g^2(y_1, y_2)$
$g(2)x+^2$
$\sigma(x_1) = 8$
$\sigma(x_2) = sum(y_1, y_2)$
$\sigma(y_1) = 3$
$\sigma(y_2) = 4$

Y posterioremte calculamos la interpretación:
$\mathcal{I}_\sigma(f^2(x_1, x_2))[x_1, x_2]$
$= f^2(8, sum(y_1, y_2))$
$= f^2(8, sum(3, 4))$
$= \times^2(8, sum(3, 4))$
$= \times^2(8, +^2(3, 4))$

$= \times^2(8,7)$
$= 56$

Ej 2. Buscamos bajo que modelos la fórmula $\exists x P(x) \to \forall x P(x)$ es satisfacible.
Primero definimos la asignatura del lenguaje:

$\mathcal{L} =< \mathcal{P}, \mathcal{F} >$
$\mathcal{P}\{\mathcal{P}^1\}$
$\mathcal{F} = \phi$

Buscamos el modelo viendo casos:

- El caso más sencillo, cuando M solo tiene un elemento:

  $\mathcal{M} =< M, \mathcal{I} >$
  $M = \{e\}$

  Si $\mathcal{P}^{\mathcal{I}} = \{e\}$ existe al menos un elemento en el modelo que lo cumple y por tanto $\mathcal{I}_\sigma(\exists x P(x)) = 1$ y al solo tener un elemento se cumple que $\mathcal{I}_\sigma(\forall x P(x)) = 1$ y así $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$.

  Por otro lado si $\mathcal{P}^{\mathcal{I}} = \phi$ entonces $\mathcal{I}_\sigma(P(e)) = 0$ entonces $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$ pues es una implicación y la premisa de la implicación es falsa.

- Cuando M tiene dos o más elementos:

  $\mathcal{M} =< M, \mathcal{I} >$
  $M = \{e_1, e_2, ...\}$

  Se abren otros tres casos

  - Cuando $\forall e_i \in M$ $\mathcal{P}^{\mathcal{I}} = \{e_i\}$ entonces $\mathcal{I}_\sigma(P(e_i)) = 1$ y por tanto $\mathcal{I}_\sigma(\exists x P(x)) = 1$ y tambien $\mathcal{I}_\sigma(\exists x P(x)) = 1$ por lo tanto $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$
  - Cuando $\mathcal{P}^{\mathcal{I}} = \phi$ entonces $\forall e_i \in M$ $\mathcal{I}_\sigma(P(e_i)) = 0$ y por tanto $\mathcal{I}_\sigma(\exists x P(x)) = 0$ entonces $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$ pues la premisa de la implicación es falsa.

- Cuando $\exists e_i \in M$ tal que $\mathcal{I}_\sigma(P(e_i)) = 1$ y $\exists e_j \in M$ tal que $\mathcal{I}_\sigma(P(e_j)) = 0$ entonces $\mathcal{I}_\sigma(\exists x P(x)) = 1$ pues hay al menos un elemento que cumple la propiedad $\mathcal{I}_\sigma(\forall x P(x)) = 0$ pues como mencionamos antes hay al menos un elemento que cumple la propiedad, por tanto no se satisface el $\forall$, entonces $\mathcal{I}_\sigma(\exists x P(x) \rightarrow \forall x P(x)) = 0$ y así encontramos el unico caso en el ue el modelo no satisface al lenguaje.

Tenemos otro caso, si queremos sustituir y aparte evaluar necesitamos usar los siguientes dos Lemas:

Lema de coincidencia de términos:
Sean $t \in TERM$ y σ_1, σ_2 dos estados de las variables tales que $\sigma_1(x) = \sigma_2(x)$ para toda variable x que figura en t. Entonces $\mathcal{I}_{\sigma_1}(t) = \mathcal{I}_{\sigma_2}(t)$
(Citamos la demostración de la nota 7 del curso.)

Lema de sustitución de términos:
Sean $r \in TERM, \sigma$ un estado de las variables, $[\vec{x} := \vec{t}]$ una sustitución y $m_1, ..., m_n \in M$ tales que $\mathcal{I}_\sigma(t_i) = m_i \ 1 \leq i \leq n$.
Entonces $\mathcal{I}_\sigma(r[\vec{x} := \vec{t}]) = \mathcal{I}_\sigma[\vec{x}/\vec{m}](r)$
(Citamos la demostración de la nota 7 del curso y las clases.)

Ej. $\mathcal{I}_\sigma(prod(8, x)[x := sum(z, w)])[z, w := 3, 4]$
$= \mathcal{I}_\sigma(prod(8, sum(z, w))[z, w := 3, 4]$
Y usando el Lema de sustitución de términos esto es equivalente a:
$= \mathcal{I}_\sigma(prod(8, sum(z, w)[z, w := 3, 4])$
$= \mathcal{I}_\sigma(prod(8, sum(3, 4))$
Y con la asignatura del lenguaje del ejemplo pasado calculamos la interpretación:
$prod = f^2(x_1, x_2)$
$f(2)x = \times^2$
$sum = g^2(y_1, y_2)$
$g(2)x+^2$
$\sigma(x_1) = 8$
$\sigma(x_2) = sum(y_1, y_2)$
$\sigma(y_1) = 3$
$\sigma(y_2) = 4$

Y posterioremte calculamos la interpretación:
$= \mathcal{I}_\sigma(prod(8, sum(3, 4))$

$= f^2(8, f^2(3,4))$
$= \times^2(8, +^2(3,4))$
$= \times^2(8,7)$
$= 56$

Si queremos interpretar las fórmulas tomamos primero σ un estado de las variables, entonces se define la interpretación de fórmulas respecto a σ, $I_\sigma : FORM \to \{0,1\}$ como sigue:

$I_\sigma(\bot) = 0$

$I_\sigma(\top) = 1$

$I_\sigma(P(t_1, ..., t_m)) = 1$ syss $(I_\sigma(t_1), ...I_\sigma(t_m)) \in P^I$

$I_\sigma(t_1 = t_2) = 1$ syss $I_\sigma(t_1) = I_\sigma(t_2)$

$I_\sigma(\neg\varphi) = 1$ syss $I_\sigma(p) = 0$

$I_\sigma(\varphi \wedge \psi) = 1$ syss $I_\sigma(\varphi) = I_\sigma(\psi) = 1$

$I_\sigma(\varphi \vee \psi) = 0$ syss $I_\sigma(\varphi) = I_\sigma(\psi) = 0$

$I_\sigma(\varphi \to \psi) = 0$ syss $I_\sigma(\varphi) = 1$ e $I_\sigma(\psi) = 0$

$I_\sigma(\varphi \leftrightarrow \psi) = 1$ syss $I_\sigma(\varphi) = I_\sigma(\psi)$

$I_\sigma(\forall x\varphi) = 1$ syss $I_{\sigma[x/m]}(\varphi) = 1 \forall m \in M$

$I_\sigma(\exists x\varphi) = 1$ syss $I_{\sigma[x/m]}(\varphi) = 1$ para algún $m \in M$

Lema 4 (Sustitución para fórmulas): Sean $\varphi \in FORM$, $\sigma : Var \to M, [\vec{x} := \vec{t}]$ una sustitución y $m_1, ..., m_n \in M$ tales que $m_i = I_\sigma(t_i)$ $1 \leq i \leq n$.

Dem: Ind/φ

Base: $\varphi = \top | \bot | P(t_1...t_n) | t_1 = t_2$

- $\varphi = \top$
  $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$

$I_\sigma(\top[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\top)$

Entonces tenemos:
$I_\sigma(\top[\vec{x} := \vec{t}]) = 1$ y $I_{\sigma[\vec{x}/\vec{m}]}(\top) = 1$
Por lo tanto se cumple con $\varphi = \top$

- $\varphi = \bot$
  $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$
  $I_\sigma(\bot\ [\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\bot)$

  Entonces tenemos:
  $I_\sigma(\bot\ [\vec{x} := \vec{t}]) = 0$ y $I_{\sigma[\vec{x}/\vec{m}]}(\bot) = 0$
  Por lo tanto se cumple con $\varphi = \bot$

- $\varphi = P(t_1...t_n)$
  $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$
  $I_\sigma(P(t_1...t_n)[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(P(t_1...t_n))$
  $I_\sigma(P(t_{1[\vec{x}:=\vec{t}]}...t_{n[\vec{x}:=\vec{t}]})) = I_{\sigma[\vec{x}/\vec{m}]}(P(t_1...t_n))$ pero esto es igual a $I_\sigma(P(t_{1[\vec{x}/\vec{m}]}...t_{n[\vec{x}/\vec{m}]}))$

  Por lo tanto se cumple con $\varphi = P(t_1...t_n)$

- $\varphi = (t_1 = t_2)$
  $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$
  $I_\sigma(t_1[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(t_1) = I : \sigma(t_2[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(t_2)$
  pero por lema 2 esto ocurre syss $I_{\sigma[\vec{x}/\vec{m}]}(t_1 = t_2)$

  Por lo tanto se cumple con $\varphi = (t_1 = t_2)$

Hipótesis de Inducción:

$$I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$$
$$I_\sigma(\psi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\psi)$$

Paso Inductivo:

$$\neg\varphi|\varphi * \psi|\forall x\varphi|\exists v\varphi$$

- $I_\sigma(\neg\varphi[\vec{x} := \vec{t}]) = 1$ syss $I_\sigma(\varphi[\vec{x} := \vec{t}]) = 0 = I_{\sigma[\vec{x}/\vec{m}]}(\neg\varphi)$ syss $I_{\sigma[\vec{x}/\vec{m}]}(\varphi) = 1$

Por lo tanto se cumple para $\neg\varphi$

- $I_\sigma(\varphi \wedge \psi[\vec{x} := \vec{t}]) = 1 syss I : \sigma(\varphi[\vec{x} := \vec{t}]) = 1$ y $I : \sigma(\psi[\vec{x} := \vec{t}]) = 1$ pero por hipótesis tenemos que

  $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$ y $I : \sigma(\psi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\psi)$ se cumple, i.e. $I_\sigma = 1$ de ambos. Por lo tanto $\varphi \wedge \psi$ se cumple.

- $I_\sigma(\varphi \vee \psi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi \vee \psi) = 1$ syss $I_\sigma(\varphi[\vec{x} := \vec{t}]) = 1$ o $I_\sigma(\psi[\vec{x} := \vec{t}]) = 1$ pero por hipótesis tenemos que

  $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi)$ y $I_\sigma(\psi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\psi)$ se cumplen, i.e. $I_\sigma = 1$ de ambos. Por lo tanto $\varphi \vee \psi$ se cumple.

- $I_\sigma(\varphi \to \psi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi \to \psi) = 0$ syss $I : \sigma(\varphi[\vec{x} := \vec{t}]) = 1$ y $I_\sigma(\psi[\vec{x} := \vec{t}]) = 0$ pero por hipótesis de inducción tenemos que $I_\sigma(\psi[\vec{x} := \vec{t}]) = 1$ entonces tenemos que $I_\sigma(\psi[\vec{x} := \vec{t}]) = 1$ y $I_\sigma(\varphi[\vec{x} := \vec{t}]) = 1$, entonces se cumple para $\varphi \to \psi$

- $I_\sigma(\varphi \leftrightarrow \psi[\vec{x} := \vec{t}]) = I_{\sigma[\vec{x}/\vec{m}]}(\varphi \leftrightarrow \psi) = 1$ syss $I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_\sigma(\varphi[\vec{x} := \vec{t}]) = I_\sigma(\psi[\vec{x} := \vec{t}])$ pero por hipótesis de inducción tenemos que $I_{\sigma[\vec{x}/\vec{m}]}(\varphi) = I_{\sigma[\vec{x}/\vec{m}]}(\psi)$ y ésto sucede syss $I_{\sigma[\vec{x}/\vec{m}]}(\varphi \leftrightarrow \psi) = 1$. Por lo tanto $\varphi \leftrightarrow \psi$ lo cumple.

- $I_\sigma(\forall x\varphi)([\vec{x} := \vec{t}])$

  Caso 1: Si $x \notin \vec{x} \cup vars(\vec{t})$

  Entonces $I_\sigma(\forall x\varphi)([\vec{x} := \vec{t}]) = I_\sigma \forall x(\varphi[\vec{x} := \vec{t}]) = 1$ syss $I_{\sigma[\vec{x}/\vec{m}]}(\varphi[\vec{x} := \vec{t}]) = 1 \forall m \in M$ pero de aquí tenemos que $I_{\sigma[x/m][\vec{x_i}/\vec{m_i}]}\varphi = I_{\sigma[x/m]}\varphi$ pero por hipótesis tenemos que $I_{\sigma[x/m][\vec{x_i}/\vec{m_i}]}\varphi = I_{\sigma[x/m]}\varphi = 1$ entonces se cumple.

  Caso 2: Si $x \in \vec{x} \cup vars(\vec{t})$

  Entonces aplicamos $\alpha - equivalencia$ y luego sustituimos como se hizo en el caso anterior y como $I_{\sigma[x/m][\vec{x_i}/\vec{m_i}]}\varphi = I_{\sigma[x/m]}\varphi = 1$ después de la sustitución sigue manteniendose en 1, entonces lo cumple.

- $I_\sigma(\exists x\varphi)([\vec{x} := \vec{t}])$

Análogo a $\forall x\varphi$

Sea una formula $\varphi \in LPO$ y $\mathcal{M} =< M, \mathcal{I} >$ una $\mathcal{L}$-estructura se dice que:

- φ es verdadera en $\mathcal{M}$ en $\forall\sigma : \text{Var} \to \mathcal{M}$ se cumple que $\mathcal{I}_\sigma(\varphi) = 1$. Y se denota como $\mathcal{M} \models \varphi$
- φ es satisfacible en $\mathcal{M}$ si $\exists\sigma : \text{Var} \to \mathcal{M}$ tal que $\mathcal{I}_\sigma(\varphi) = 1$
- φ es falsa en $\mathcal{M}$ si y solo si $\mathcal{M} \models \neg\varphi$

Si sabemos que la formula tiene un modelo que es satisfacible pero no sabemos cual es, para encontrar este es necesario seguir los siguientes pasos:

1. Damos la asignatura del lenguaje incluyendo los predicados y formulas que se van a usar.
2. Empezamos construyendo el modelo más simple posible (elemento a elemento).
3. Construimos las formulas y predicados deacuerdo a la semantica (la interpretación de las formulas).
4. Y por ultimo damos la justificación de por que funciona como modelo.

Ej 1: Buscamos un modelos que satisfaga $I_\sigma(\forall x P(x) \to \exists x P(x)) = 1$.
Definimos La asignatura del lenguaje como sigue:
$\mathcal{L} =< \mathcal{P}, \mathcal{F} >$
$\mathcal{P} = \{P^{(1)}\}$
$\mathcal{F} = \phi$

Ahora seguimos con el modelo, el cual como se puede notar solo utiliza un elemento, pues el predicado relacionado es solo uno el cual solo opera sobre una variable, por tanto, lo definimos el modelo cómo:
$\mathcal{M} =< M, \mathcal{I} >$
$M = \{e\}$

Construimos el predicado P de acuerdo a la interpretación de la formula que nos conviene.
$\mathcal{P}^{\mathcal{I}} = \{e\}$
Ya que por hipotesis tenemos que $I_\sigma(\forall x P(x) \to \exists x P(x)) = 1$ primero nos fijamos en todo estado en M y así garantizamos que

$$\forall m \in M I_\sigma[x/m] I_\sigma(P(e)) = 1$$

Y posteriormete lo verificamos para el existencial

$$\exists m \in M I_\sigma[x/m] I_\sigma(P(e)) = 1$$

$\therefore \forall x P(x) \to \exists x P(x)) = 1$

Y así encontramos el modelo.

Podemos notar que esta formula no solamente es satisfacible si no que es universalmente valida, lo demostramos a continuación:

Sea $\mathcal{M}$ una $\mathcal{L} - estructura$ P.d que $\mathcal{M} \models \forall x P(x) \to \exists x P(x))$ si y solo si $\forall x P(x) \to \exists x P(x)) = 1$

- Caso 1. Supongamos que $I_\sigma(P(x)) = 0$ entonces al ser una implicación el resto no nos interesa pues el subsecuente es verdadero, así $\forall x P(x) \to \exists x P(x)) = 1$.
- Caso 2. Supongamos que $I_\sigma(P(x)) = 1$ entonces la implicación es verdadera tanto en su consecuente como en su subsecuente, por lo tanto $\forall x P(x) \to \exists x P(x)) = 1$.

Ej 2. Buscamos bajo que modelos la fórmula $\exists x P(x) \to \forall x P(x)$ es satisfacible.
Primero definimos la asignatura del lenguaje:

$\mathcal{L} =< \mathcal{P}, \mathcal{F} >$
$\mathcal{P}\{\mathcal{P}^1\}$
$\mathcal{F} = \phi$

Buscamos el modelo viendo casos:

- El caso más sencillo, cuando M solo tiene un elemento:

  $\mathcal{M} =< M, \mathcal{I} >$
  $M = \{e\}$

  Si $\mathcal{P}^{\mathcal{I}} = \{e\}$ existe al menos un elemento en el modelo que lo cumple y por tanto $\mathcal{I}_\sigma(\exists x P(x)) = 1$ y al solo tener un elemento se cumple que

$\mathcal{I}_\sigma(\forall x P(x)) = 1$ y así $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$.

Por otro lado si $\mathcal{P}^\mathcal{I} = \phi$ entonces $\mathcal{I}_\sigma(P(e)) = 0$ entonces $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$ pues es una implicación y la premisa de la implicación es falsa.

- Cuando M tiene dos o más elementos:

  $\mathcal{M} =< M, \mathcal{I} >$
  $M = \{e_1, e_2, ...\}$

  Se abren otros tres casos

  - Cuando $\forall e_i \in M$ $\mathcal{P}^\mathcal{I} = \{e_i\}$ entonces $\mathcal{I}_\sigma(P(e_i)) = 1$ y por tanto $\mathcal{I}_\sigma(\exists x P(x)) = 1$ y tambien $\mathcal{I}_\sigma(\exists x P(x)) = 1$ por lo tanto $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$
  - Cuando $\mathcal{P}^\mathcal{I} = \phi$ entonces $\forall e_i \in M$ $\mathcal{I}_\sigma(P(e_i)) = 0$ y por tanto $\mathcal{I}_\sigma(\exists x P(x)) = 0$ entonces $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 1$ pues la premisa de la implicación es falsa.
  - Cuando $\exists e_i \in M$ tal que $\mathcal{I}_\sigma(P(e_i)) = 1$ y $\exists e_j \in M$ tal que $\mathcal{I}_\sigma(P(e_j)) = 0$ entonces $\mathcal{I}_\sigma(\exists x P(x)) = 1$ pues hay al menos un elemento que cumple la propiedad $\mathcal{I}_\sigma(\forall x P(x)) = 0$ pues como mencionamos antes hay al menos un elemento que cumple la propiedad, por tanto no se satisface el $\forall$, entonces $\mathcal{I}_\sigma(\exists x P(x) \to \forall x P(x)) = 0$ y así encontramos el unico caso en el ue el modelo no satisface al lenguaje.

Clasificación de formulas

Diremos que dos formulas son equivalentes cuando:

$\forall M$ $L - estructura$ y $\forall \sigma : Var \to M$ se tiene que $M \models \varphi$ syss $M \models \psi$

La principal diferencia que tiene esto con la lógica proposicional es que en LPO dos fórmulas son equivalentes si su interpretación es la misma para cualquier modelo y para cualquier estado, mientras que en la lógica proposicional solamente nos fijamos en que su interpretación sea la misma.

Ahora tratemos de ver que esto se cumple:

$\forall x\varphi \equiv \forall y(\varphi[x := y])$

Esta prueba puede realizarse por inducción como sigue:

Base: $\varphi = \top | \perp | P(t_1...t_n) | t_1 = t_2$

Notemos que para la base en los casos de $\top$ $\perp$ es facil ver que se cumple pues I_σ es la misma en ambos casos 1 y 0 respectivamente. Después para $P(t_1...t_n)$ solamente tenemos que aplicar una sustitución en el lado izquierdo y una "doble" en el lado derecho como se vio en el lema 4. Para $t_1 = t_2$ como tenemos que ver que $\forall x\varphi \equiv \forall y(\varphi[x := y])$ entonces sabemos que $I_\sigma[x := t](t_1) = 1$ syss $I_\sigma[x := t](t_2) = 1$.

Hipótesis de Inducción: $\forall x\varphi \equiv \forall y(\varphi[x := y])$ y $\forall x\psi \equiv \forall y(\psi[x := y])$

Paso Inductivo: $\neg\varphi | \varphi * \psi | \forall z\varphi | \exists z\varphi$

Para $\neg\varphi$ tenemos que $I_\sigma[x := t](\neg\varphi) = 1$ syss $I_\sigma[x := t](\varphi) = 0$ entonces al aplicar la sustitución se conserva por a H.I.

Para $\varphi * \psi$ solamente tenemos que verificar la interpretación de la fórmula (con cada operador binario) una vez aplicada la sustitución como se hizo en el Lema 4.

Para los cuantificadores $\forall z\varphi | \exists z\varphi$ tenemos el mismo caso al aplicar la sustitución se mantiene por H.I. y seguimos como se hizo en el Lema 4, manejandonos cuando φ "es un cuantificador" y al aplicarlo a otro cuantificador entonces podemos ver que se mantiene de ambos lados al aplicar la sustitución.

Indecidibilidad

La lógica de predicados es indecidible pues cuando buscamos una solución al problema de validez universal este es indecidible ya que tendríamos que probar que para cualquier modelo que exista éste será cierto siempre, lo cual hace que inclusive un computador poderoso se cicle infinitamente. Esto es enunciado en el Teorema de indecibilidad de Church:

"El problema de validez universal es indecidible. Es decir, no puede existir un algoritmo que reciba un enunciado φ como entrada y decida si $\models \varphi$".

Lo mismo ocurre con las equivalencias lógicas, las concecuencias lógicas y saber si una formula es satisfacible, ya que aún sin el problema de la validez universalsi tiene demasiadas formulas y/o predicados el buscar un modelo se va a hacer exponencialmente dificil.

Conclusiones

A veces requerimos formalizar las cosas sobre lo que hablamos en español, tal como se vio en la lógica proposicional, pero mientras la estudiabamos nos dimos cuenta que en algunas ocasiones noveíamos demasiado limitados en cuanto a las expresiones que si podemos formalizar y las que no, como lo puede ser con expreziones que hablen de "cuantos".

Ej:
"Para cualquier número existe uno que es mayor que él"

Ésta es una expresión que no nos era posible formalizar en la LP. Para ésto es que comenzamos a estudiar la lógica de primer orden (o lógica de predicados) que es una herramienta más poderosa.

A lo largo de este estudio nos dimos cuenta que podemos cuantificar una fórmula que se consta de variables, predicados y cuantificadores. Sobre dichas formulas también definimos nuevas "operaciones" como la sustitución "de valores" y de la misma forma definimos el ligado y alcance de un cuantificador y la $\alpha - equivalencia$ para cuando no nos era posible sustituir textualmente.

A pesar de que ésta es una herramienta mucho más poderosa que la lógica proposicional ésta también puede no sernos suficiente en algunos casos. Por ejemplo algo que nos interesa saber siempre es la veracidad o satisfacibilidad de una fórmula, pero en LPO si ésta tiene como cuantificador un $\forall x(\varphi)$ con $\varphi \in FORM$ entonces tendríamos o podríamos encontrar un modelo que haga verdadera dicha fórmula, el problema es que si quisieramos implementarlo esto nos sería imposible ya que por más buena que fuese nuestra implementación, ésta se ciclaria ya que trataria de evaluar una infinidad de elementos en nuestro universo de discurso (Ej: Como puede ser trabajar una propiedad sobre los números naturales).

Con ésto concluimos, además, que a pesar de ser una herramienta mucho más poderosa, esta tambien es indecidible.

Sintáxis:

– Los indices los definimos como enteros para poder usarlos como variables.
type Ind = Int

– El nombre de una función o un predicado es una cadena.
type Nombre = String

– Define los términos como una variable o una función de n términos. (v in V)
data Term = V Ind | F Nombre [Term] deriving (Show, Eq)

– Define las fórmulas como True, False, predicados, igualdad, operadores binarios,
– cuantidicadores.
data Form = TrueF
| FalseF
| Pr Nombre [Term]
| Eq Term Term
| Neg Form
| Conj Form Form
| Disy Form Form
| Imp Form Form
| Equiv Form Form
| All Ind Form
| Ex Ind Form

– Función que te dice las variables libres de una formula. Checa caso a caso.
– Las variables libres de una formula son todas aquellas que no están cuantificadas.
fv :: Form -> [Ind]

– Recibe un término y nos devuelve una lista de indices (usamos "map" porque
– aplica la función a cada 1).
varT :: Term -> [Ind]

– Función que nos devuelve las varibales ligadas de una formula.
bv :: Form -> [Ind]

– Cerradura de universal.
– La cerradura añade un cuatificador a cada cosa que no lo tenga.
aCl :: Form -> Form

– Mete el "para todo" a cada elemento.
aClaux :: Form -> [Ind] -> Form

La cerradura existencial es análoga.

– Definimos la sutitución como (Indice, Termino).
type Subst = [(Ind, Term)]

– Verfica si una sustitución es válida.
verifSus :: Subst -> Bool

– Sustitución para terminos.
apsubT :: Term -> Subst -> Term

– Sustitución de Formulas.
apsubF :: Form -> Subst -> Form

– Verifica si hay elementos repetidos en una lista de elementos comparables.

```
tieneRep :: (Eq a) => [a] -> Bool
```

Semántica:

– Definimos la interpretación de fórmulas como una función que toma un nombre,
– una lista de elementos del universo y devuelve un elemento del universo.

```
type IntF a = Nombre -> [a] -> a
```

– Definimos la interpretación de predicados como una función que toma un nombre,
– una lista de elementos del universo y nos dice si los elementos en nuestro universo
– se relacionan.

```
type IntR a = Nombre -> [a] -> Bool
```

– Definimos el estado de un elemento del universo como: sigma: Var -> M

```
type Estado a = Ind -> a
```

– L-Estructura

```
type Estructura a = ([a], IntF a, IntR a)
```

– Actualización de estado. Dado un estado, una variable y un elemento
– esto nos devuelve el estado actualizado de dicha varibale.

```
actEst :: Estado a -> Ind -> a -> Estado a
```

– Interpretación de fórmulas. Dada una estructura, un estado y una fórmula
– determina la veracidad de dicha fórmula bajo una interpretación dad y el Estado
– acuatlizado

```
iForm :: Eq a => Estructura a -> Estado a -> Form -> Bool
```

– Satisfacibilidad de una fórmula. Dada una Estructura, un estado de las variables
– y una fórmula nos dice si la fórmula es satisfacible con ese modelo dado.
satForm :: Eq a => Estructura a -> Estado a -> Form -> Bool