Notas de Lógica para Computación

Carlos Domingo

Enero, 1997

Capítulo 1

Lógica de Proposiciones

1.1 Proposiciones y proposiciones lógicas

La Lógica trata de las propiedades de las oraciones o proposiciones, de la composición de las proposiciones simples para formar compuestas, y de la deducción de unas proposiciones a partir de otras. Se irá viendo que la Lógica no abarca el lenguaje común sino que es como un modelo simplificado de tal lenguaje. Tal simplificación disminuye el **poder representativo**, pues sólo una parte de los enunciados del lenguaje común pueden expresarse, pero permite una mayor **precisión** en la expresión y un **poder deductivo** muy superior al del lenguaje corriente dentro del ámbito que puede representar.
Es conocido que a las proposiciones (o por lo menos a ciertas proposiciones) se les puede asignar el calificativo **falsa** o **verdadera**.
En este texto se representarán la proposiciones por letras minúsculas y los valores verdadero y falso por V y F respectivamente. Así p=F significa que la proposición p tiene el **valor de verdad** F, es decir que es falsa.

Ejemplo 1.1.1

Decir a cuales de las proposiciones siguientes se les puede asignar uno de los valores V o F. **Justificar la asignación**. Si hay dudas decir en base a qué se podría hacer una asignación y en los casos en que ello no sea posible decir cuales son las dificultades.

q= "3 es mayor que 2"	q=V
p= "El cielo está nublado"	p=__
s= "Estaba vivo pocos minutos antes de morir"	s=__
t= "Por favor, hable un poco más lento"	t=__
z= "Ojalá que llueva"	z=__
r= "¿Quién ha llegado?"	r=__
u= "La proposición u es falsa"	u=__
v= "El rey de Venezuela no es ateo"	v=__

En lo que sigue se tratará sólo de las proposiciones a las cuales puede asignarse un valor V o F. Tales proposiciones suelen denominarse **proposiciones lógicas**. Se llamarán

simplemente **proposiciones**. Se ve pues que la lógica trata solamente de una parte del lenguaje corriente o natural. Por otra parte queda claro que para muchas proposiciones el criterio de asignar el valor V o F necesita de conocimientos ajenos a la lógica.

1.2 Proposiciones simples y compuestas

En el lenguaje natural se pueden formar proposiciones compuestas de varias proposiciones. Usualmente están unidas por conjunciones (y,ni,o) por preposición y adverbio (si...entonces...) o por verbos (implica).
Se plantea el problema de si es posible asignar un valor de verdad a la proposición compuesta cuando se conocen los valores de verdad de las componentes.
Es usual llamar a las proposiciones simples **atómicas** y a las compuestas **fórmulas**.

Ejemplo 1.2.1

Vea si puede asignar un valor V o F a las siguientes proposiciones compuestas suponiendo diferentes valores para las componentes. Señalar dificultades y ambigüedades.

p="El perro está durmiendo y el gato está despierto".
q="Si un entero es múltiplo de 6 entonces es múltiplo de 3".
r="Si llega Pedro entonces se hace la reunión".
s="Lo dijo o no lo dijo".
t="Un pan es un pan o Ecuador está en Europa".
u="Es un satélite si y sólo si gira en torno a otro astro".
v="Cayó del árbol y se fracturó un hueso".
w="Se fracturó un hueso y cayó del árbol".
x="3 más 4 es 7 y 8 es divisible por 5".
y="No ha llegado"
z="No es cierto que Miguel ha llegado"
m="Llueve, implica que esta nublado"

En los ejemplos se puede ver que las proposiciones componentes están unidas por ciertos términos de conexión (conjunciones, preposiciones, verbos, adverbios etc.) y que la verdad de la proposición compuesta depende de la naturaleza de dichos conectivos (y, o, si.. entonces, si y sólo si). Se ve también el uso del modificador **no** o **no es cierto que**. En la Lógica Proposicional se introducen conectivos para formar proposiciones compuestas. Tales conectivos no se corresponden exactamente con los del lenguaje corriente antes mencionados. Se los define de manera que sean útiles en las Matemáticas y la Computación. Los conectivos se considerarán operadores (como los de la Aritmética) que actuando sobre una o dos proposiciones producen una nueva proposición. El valor de verdad de la proposición compuesta depende de los valores de verdad de los operandos. Veamos como se definen tales valores y una justificación informal de las definiciones.

- Operador unario **negación**: $\neg$ o bien $-$.

  Equivale al **no** del lenguaje natural. Puesto delante de una proposición cambia su valor de verdad. Si $p = F$ entonces es $\neg p = V$. Si $p = V$ entonces $\neg p = F$.

- Operador binario conjunción $\wedge$.

  Es semejante al conectivo **y**. Toda ambigüedad se elimina definiendo $p \wedge q$ como verdadero si y solamente si $p = V$ y $q = V$, es decir si ambos componentes son verdaderos. Se ve en los ejemplos 1.2.1 v w que en el lenguaje corriente el conector **y** puede tener un significado de que el primer componente expresa la causa o antecede del segundo. Tal significación no se atribuye al conector $\wedge$. El ejemplo 1.2.1 x podría ser considerado como parcialmente cierto pero poniendo el conector $\wedge$ se lo considera falso ya que la lógica proposicional no acepta otros valores que V o F para el valor de verdad de las proposiciones.

- Operador binario **disyunción**: $\vee$.

  Es semejante al conectivo **o** pero no exclusivo. La ambigüedad se elimina definiendo $p \vee q$ como verdadero si $p = V$ o bien $q = V$ o bien si ambos son V, es decir, si algún componente es verdadero. Las expresiones ordinarias con "o" suponen una relación de exclusión entre las dos proposiciones, tal como se ve en el ejemplo anterior.

- Operador binario **implicación**: $\supset$.

  Es semejante al conectivo **si...entonces** del lenguaje natural. En que forma se debe asignar el valor de verdad a $p \supset q$ cuando se dan valores a p y q ha sido objeto de muchas discusiones desde la antigüedad.[1] Es más inteligible el problema si razonamos partiendo de la verdad de la proposición compuesta $p \supset q$. Supongamos que la compuesta es verdadera. Entonces si p es verdadera es claro que también lo es q según el significado corriente del si..entonces. Por otra parte si p es verdadera y q es falsa es claro que la implicación que se está asegurando en la compuesta $p \supset q$ es falsa, es decir no es correcto hacer tal implicación. Tenemos pues:

  $$\text{Si } p = V \text{ y } q = V \text{ resulta } p \supset q = V$$
  $$\text{Si } p = V \text{ y } q = F \text{ resulta } p \supset q = F$$

  El problema se presenta cuando p es falsa, en cuyo caso el lenguaje corriente nada aclara, pero el modelo lógico debe asignar un valor.

  Es claro que $p \supset p$ debe ser siempre verdadera, cualquiera sea el valor de p ("Si Aristóteles era francés entonces Aristóteles era francés", es verdadera). Para que eso ocurra debe admitirse que si ambos operandos son falsos la implicación es verdadera. (esto se expresa en la ironía del lenguaje común:" Si este texto de Lógica está claro entonces yo soy el Emperador de China", donde se supone la implicación verdadera y ambas proposiciones falsas). Se tiene pues:

  $$\text{Si } p = F \text{ y } q = F \text{ resulta } p \supset q = V$$

[1] Las discusiones en Museo de Alejandría (siglo II) parecen haber llegado a molestar al gran poeta Calímaco que escribió el famoso epigrama: "Hasta los cuervos en los tejados graznan acerca de cuales implicaciones son verdaderas"

Queda por resolver el caso $p = F$ $q = V$. Esto se decide por la conveniencia de que la relación de implicación sea asimétrica como lo es en el lenguaje corriente ("Si llueve entonces está nublado" no equivale a que "Si está nublado entonces llueve").

Si se admitiera que en este caso $p \supset q$ fuera falsa entonces resultaría que $p \supset q$ equivaldría a $q \supset p$ lo cual no se quiere (ver tablas de verdad más adelante). Es por lo tanto adecuado que en este caso se defina:

$$\text{Si } p = F \text{ y } q = V \text{ resulta } p \supset q = V$$

En conclusión **la implicación sólo es falsa si el primer operando es verdadero y el segundo operando es falso**.

- Operador binario **equivalencia**: $\equiv$.

  Este equivale al conectivo **si y solamente si** del lenguaje corriente. También se expresa por **equivale a** . Es claro que la equivalencia es verdadera si y sólo si ambos operandos tienen el mismo valor de verdad.

  Las definiciones de los operadores anteriores quedan pues definidos por las siguientes tablas, llamadas **tablas de verdad** que dan los valores de verdad que resultan cuando aplicamos los operadores a dos proposiciones.

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \supset q$	$p \equiv q$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

  Así por ejemplo cuando p es V y q es F (segunda línea) se ve que $p \supset q$ es F y $p \vee q$ es V.

Ejercicio 1.2.3

Usando tablas de verdad probar la equivalencia de las fórmulas siguientes:

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

estas relaciones son conocidas como Leyes de De Morgan y son muy útiles para transformar conjunciones y disyunciones.

$$p \supset q \equiv \neg p \vee q$$

en algunos textos se toma esta relación como una definición de implicación.

$$(p \equiv q) \equiv (p \supset q) \wedge (q \supset p)$$

La implicación es importante en el razonamiento matemático. En él se usa la expresión:

"p es condición necesaria de q"

o mas detalladamente:

"para que q sea verdadera es necesario que p sea verdadera"

Esta afirmación equivale a que $q \supset p$ es verdadera, pues en el caso de que la implicación sea verdadera, para que sea verdadera q, debe necesariamente ser verdadera p.

p	q	$q \supset p$
V	V	V
V	F	V
F	V	F
F	F	V

Que $q \supset p$ es verdadera significa que estamos en los dos primeros casos, o en el último, y para que sea q=V debe ser necesariamente p=V. Pero esto último no es suficiente pues la segunda línea muestra que la implicación puede ser V y p puede ser V y sin embargo q ser F. Así cuando $q \supset p$, p es condición necesaria para la verdad de q, pero no es suficiente. Así cuando a partir de q deducimos, por algún razonamiento aceptado, la verdad de p sólo hemos demostrado que esta p es una condición necesaria de la verdad de q, pero no suficiente.

Otra expresión usada es:

"p es condición suficiente de q"

o mas detalladamente:

"para que q sea verdadera es suficiente (basta) que p sea verdadera"

Esta equivale a que $p \supset q$ es verdadera, pues en el caso de que así lo sea y p es verdadera también lo es q. Pero no es necesario que p sea verdad para que q lo sea pues también puede ser p falsa y q ser verdadera. Esto se ve en la tercera fila de la tabla:

p	q	$p \supset q$
V	V	V
V	F	F
F	V	V
F	F	V

Así cuando a partir de p deducimos, por algún razonamiento aceptado, la verdad de q sólo hemos demostrado que esta p es una condición suficiente de la verdad de q, pero no es una condición necesaria.

Por último:

"p es condición necesaria y suficiente de q"

o bien:

"q si y sólo si p"

equivale a $p \equiv q$. Esta equivalencia suele expresarse: $p \Leftrightarrow q$

Ejercicio 1.2.4

Demostrar que para que una condición p sea necesaria y suficiente para q basta demostrar:

que p implica q y que q implica p

o bien:

que p implica q y que $\neg p$ implica $\neg q$.

Ejercicio 1.2.5

Hallar ejemplos de condición necesaria pero no suficiente, de suficiente pero no necesaria y de necesaria y suficiente con casos de divisibilidad de enteros.
Un **literal** es una proposición simple p o una proposición precedida de una negación. Así p, q, $\neg r$, $\neg p$ son literales. Pero $p \wedge q$ no es un literal sino una fórmula

1.3 Fórmulas proposicionales

A partir de estas proposiciones formadas por la negación de una proposición y de la unión de dos proposiciones simples mediante los conectivos

$$\wedge, \vee, \supset, \equiv$$

se pueden construir fórmulas más complicadas mediante las siguientes reglas de construcción:

- Toda proposición es una fórmula.
- Si X e Y son fórmulas entonces:

  $$(\neg X), (X \wedge Y), (X \vee Y), (X \supset Y), (X \equiv Y), (X)$$

  son también fórmulas.
- Aplicando las reglas anteriores un número finito de veces se obtienen todas las fórmulas. En algunos textos las fórmulas así formadas se denominan **fórmulas bien formadas**.

En lo anterior se han denotado las fórmulas por letras mayúsculas y los paréntesis tienen el significado usual en Matemáticas.
Pueden ahorrarse paréntesis si las operaciones se hacen de derecha a izquierda y en el orden de precedencia: $\neg, \vee, \wedge, \supset, \equiv$ ($\vee$ y $\wedge$ son de igual precedencia. Así por ejemplo:
$r \wedge p \vee \neg q \supset q \supset \neg r$ se interpreta:

$$((((r \wedge p) \vee (\neg q)) \supset q) \supset (\neg r))$$

Las reglas de construcción anteriores sirven también para reconocer si una hilera de símbolos es o no una fórmula. Para ello se siguen las reglas siguientes:

- Se recorre la expresión de izquierda a derecha mientras no se pueda definir la última parte recorrida como fórmula
- Si lo último recorrido es una fórmula sustituir por "fórmula" y continuar el recorrido
- Si al terminar el recorrido el total se reduce a una fórmula se ha reconocido la expresión original como fórmula. Si no la expresión no vale como fórmula.

Ejemplo 1.3.1

$$((((\neg(A \supset B)) \supset C) \supset (R \supset (\neg S)))$$

en los pasos sucesivos se identifican como "fórmulas" las siguientes expresiones :

$$A, B, (A \supset B)$$

$$(\neg(A \supset B))$$

$$C, (((\neg(A \supset B)) \supset C)$$

$$R, S, (\neg S)$$

$$(R \supset (\neg S))$$

$$((((\neg(A \supset B)) \supset C) \supset (R \supset (\neg S)))$$

Ejercicio 1.3.1

Ver por el procedimiento anterior que:

$$((((\neg(A \supset B)) \supset C)\neg(R \supset (\neg S)))$$

no es una fórmula.

Ejercicio 1.3.2

Diseñar un algoritmo que lea una hilera de símbolos y decida si es o no una fórmula.
Cuando no hay posibilidad de confusión pueden omitirse los paréntesis. Se hace la convención de que el signo $\neg$ se considera antes que el $\supset$.
La fórmula del ejemplo 1.3.1 puede escribirse:

$$(\neg(A \supset B) \supset C) \supset (R \supset \neg S)$$

Puede verse que los operadores $\wedge, \vee, \equiv$ pueden ponerse en función de $\neg$ y de $\supset$. Por lo tanto con las reglas de construcción vistas se pueden reducir a :

$$(\neg X), (X \supset Y), (X)$$

y todavía construir todas las fórmulas del cálculo proposicional.

Ejercicio 1.3.3

Demostrar la afirmación anterior. Usar los resultados del ejercicio 1.2.3.
Por último todos los operadores pueden deducirse del operador binario "y negativo" (nand) que se representa $\uparrow$. Aplicado a dos operandos da F si y sólo si ambos son verdaderos.

Ejercicio 1.3.4

Ver que $\neg p = p \uparrow p$
Hallar la expresión de $\supset$ en función de $\uparrow$

1.4 Fórmulas consistentes. Tautologías. Deducción

Un problema de gran importancia en la aplicación de la lógica es el de verificar si una cierta proposición, o en general cierta fórmula proposicional es o no deductible de un conjunto dado de fórmulas. Tal problema equivale a saber si una demostración ha sido correctamente hecha o no, o dicho de otra forma, a saber si es lícito concluir algo de un conjunto de proposiciones o fórmulas aceptadas. Para atacar este problema es necesario introducir una serie de conceptos nuevos.
Una **interpretación** o **modelo** de una fórmula es una asignación de un valor de verdad a cada una de sus componentes.

Ejemplo 1.4.1

$p \supset \neg q$ tiene una interpretación si ponemos $p = V$ $q = V$. En esta interpretación la fórmula toma el valor F, en otras interpretaciones podría tomar otro valor.
Una fórmula es **consistente** si puede ser verdadera para alguna interpretación.

Ejemplo 1.4.2

La fórmula del ejemplo anterior es consistente. La fórmula $p \wedge \neg p$ no es consistente.
Una fórmula que no es verdadera para ninguna interpretación se llama **inconsistente**.
La consistencia o inconsistencia se puede averiguar por tablas de verdad o transformando la fórmula en otra en que sea obvia la consistencia o inconsistencia.

Ejercicio 1.4.1

Averiguar la consistencia de $p \wedge \neg q$
Una fórmula se dice **válida** si es verdadera para toda interpretación posible.

Ejemplo 1.4.3

$p \vee \neg p$ es válida.
Las fórmulas válidas se llaman también **tautologías**.
Una fórmula consistente pero no válida se denomina **contingente**. Para alguna interpretación es V, para alguna otra es F.
Para explicar el **principio de deducción** comenzamos con la definición de consecuencia lógica de un conjunto de fórmulas (hipótesis).
Si S es un conjunto de fórmulas y A una fórmula entonces:

$$S \models A$$

significa que todas las interpretaciones que hacen verdaderas **todas** las fórmulas que pertenecen a S hacen también verdadera a A. Se dice también que A es **consecuencia lógica** de las fórmulas de S. Para que no lo sea se requiere que exista una interpretación que haga verdaderas todas las fórmulas de S y haga falsa A.
Nótese que esta definición no coincide totalmente con lo que se entiende comúnmente por **consecuencia**. Es por ejemplo claro que, según el lenguaje corriente, tal denominación se aplica a:

$$\{p \supset q, q \supset r\} \models p \supset r$$

donde la consecuencia se debe a la propiedad transitiva de $\supset$ que se prueba con tablas de verdad. Pero según la definición también es:

$$\{p \supset q, q \wedge r, a \vee r\} \models p \supset r$$

o bien:

$$\{s\} \models (s \wedge (p \vee \neg p))$$

Ejercicio 1.4.3

Probar que estas dos últimas afirmaciones se ajustan a la definición de consecuencia lógica. Si el conjunto S es cualquiera y A es una tautología es siempre $S \models A$ pues ninguna interpretación hace falsa A. En particular si el conjunto S es vacío y A es una tautología es:

$$S \models A$$

Por eso para indicar que A es una tautología se indica abreviadamente:

$$\models A$$

Veremos en lo que sigue como puede verse si una fórmula es o no consecuencia lógica de otras. Una forma es usar la importante relación entre consecuencia lógica e implicación. Si H y C son fórmulas entonces C es consecuencia lógica de H (es decir: $\{H\} \models C$) si y sólo si $H \supset C$ es una tautología.
En efecto, si $H \supset C$ es una tautología (verdadera para cualquier valor de las componentes) nunca se daría $H = V, C = F$, por lo tanto siempre que $H = V$ debe ser $C = V$. Así que C es consecuencia lógica de H. Por otra parte si $\{H\} \models C$ siempre que $H = V$ es $C = V$ así que $H \supset C$ tiene siempre valor V.
Esto se generaliza fácilmente. Si $H_1, H_2, H_3, ..., H_n$, y C son fórmulas, entonces:

$$\{H_1, H_2, ..., H_n\} \models C \Leftrightarrow \models ((H_1 \wedge H_2 \wedge ... \wedge H_n) \supset C)$$

La demostración es análoga al caso de una sola fórmula H.
En los razonamientos deductivos las H_i se suelen llamar **hipótesis** y C la **conclusión**. Lo usual es suponer las hipótesis válidas y combinar unas fórmulas H_i con otras agregando las definiciones y otras fórmulas cuya validez se ha demostrado o aceptado y llegar a que la fórmula C es verdadera. No hay reglas para hacer esto sin tanteos.
La forma de deducción indicada en la equivalencia anterior exigiría analizar los valores de verdad de todas las H_i y de C, verificando que la implicación es una tautología. Sólo así sabríamos que la deducción es correcta.

Ejercicio 1.4.4

Considerar el conjunto de fórmulas:

$$\{p \vee (q \wedge r), \neg p, p \supset q\}$$

y ver si $\neg p \wedge q$ es consecuencia lógica de aquel conjunto.
1) Usar deducción basada en las definiciones de los conectivos.
2) usar la equivalencia anterior viendo por tablas de verdad si la implicación es una tautología.
Más eficiente es el llamado **principio de deducción**. Para enunciarlo se conviene en decir que un **conjunto** S de fórmulas es **consistente** si **todos** sus componentes se hacen verdaderos para **alguna interpretación**.
Para esa interpretación se dice que el conjunto es **consistente**.

Ejercicio 1.4.5

Ver si el conjunto del ejercicio anterior:
$\{p \vee (q \wedge r), \neg p, p \supset q\}$ es consistente.
Nótese que la consistencia del conjunto equivale a que sea consistente la fórmula constituida por la conjunción de todas sus fórmulas.
Viendo la definición de consecuencia lógica se ve que si un conjunto de fórmulas $\{S_1, S_2, .., S_n\}$ es inconsistente, cualquier fórmula Q es su consecuencia lógica pues la implicación $(S_1 \wedge S_2 \wedge .. \wedge S_n) \supset Q$ es una tautología.
Para indicar que una fórmula inconsistente es consecuencia lógica del conjunto de fórmulas S se indica $S \models F$.
Con esta definición el **principio de deducción** se enuncia:

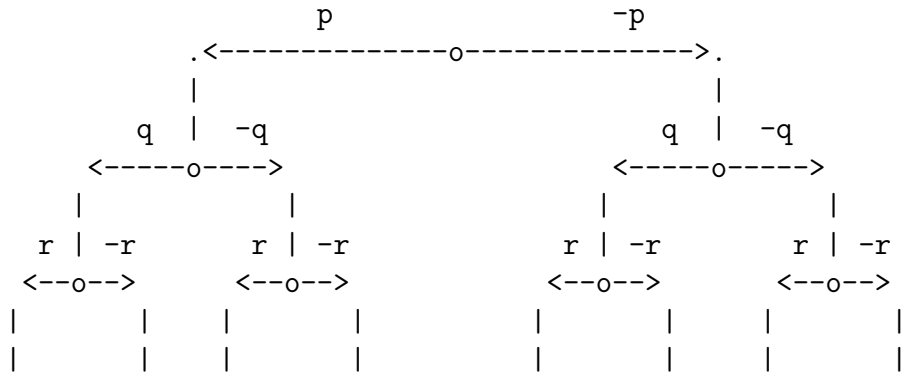
> C es consecuencia lógica del conjunto de fórmulas S si y sólo si el conjunto de fórmulas $H = S \cup \{\neg C\}$ es inconsistente, es decir F para toda interpretación.

Esto se ve notando que, o bien S es inconsistente con lo cual C (o cualquier otra fórmula) es su consecuencia lógica, o bien S es consistente por lo cual, para toda interpretación que hace S verdadero, para hacer H falso debe ser $\neg C$ falso o sea C verdadero.
Este principio de deducción es el que en la Matemática se llama de **demostración por el absurdo**: De una colección consistente de hipótesis y de la negación de la tesis a demostrar se deduce una proposición inconsistente. Esto prueba la tesis. En particular se puede deducir el propio C. Como se supone $\neg C$ verdadero se ha llegado a la proposición inconsistente $C \wedge \neg C$.

1.5 Algoritmos. Reducción

Ya hemos visto como se puede saber, mediante un algoritmo, si una hilera de símbolos es una fórmula. Se trata ahora de ver procedimientos para saber si una fórmula es consistente (verdadera para alguna interpretación) o válida (verdadera para toda interpretación) o inconsistente (falsa para toda interpretación). El método de las tablas de verdad que

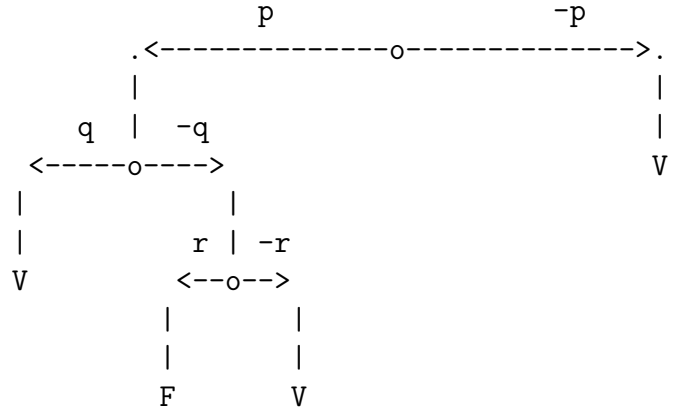
examina todos las combinaciones posibles de valores de verdad de la fórmula es siempre posible pero se hace inmanejable para fórmulas de muchas variables. Una expresión con 30 variables tendría 2^{30} o sea más de 1000 millones de líneas. La tabla de verdad se puede representar también como un árbol semántico binario. Si p,q,r son las variables o proposiciones literales:



En cada nodo se transforma la expresión poniendo el valor de verdad que resulta de reemplazar en la fórmula los valores de verdad asignados en el camino para llegar al nodo. Una trayectoria de la raíz a la hoja es una interpretación.
Hacer el árbol completo no es necesario si se ve que para una rama que sale de un nodo ya no se modificará el valor de verdad obtenido. Este procedimiento se llama algoritmo de Quine.

Ejemplo 1.5.1

Sea la proposición $(p \supset q) \vee (r \supset q)$



Se ve que no hace falta ver las alternativas de q y r cuando p=F.

Ejercicio 1.5.1

Demostrar usando árboles incompletos que:
$(((p \wedge q) \supset r) \wedge (p \supset q)) \supset (p \supset q)$ es válida.
$(((p \vee q) \supset (p \wedge r)) \supset (r \wedge q)) \supset (p \supset q)$ es consistente pero no válida.
Decir para que conjuntos de valores de p, q, r es verdadera.
Otro algoritmo que puede resultar útil es el llamado de **reducción al absurdo**. Consiste en suponer la fórmula falsa y de allí llegar a través de interpretaciones de subfórmulas, a la interpretación de sus componentes. Si esta interpretación es incompatible con alguna de las de las subfórmulas nuestra hipótesis inicial es falsa y por lo tanto la fórmula es válida.

Ejemplo 1.5.2

Sea la primera fórmula del ejercicio 1.5.1 igualada a F :

$$(((p \wedge q) \supset r) \wedge (p \supset q)) \supset (p \supset r) = F \tag{1}$$

De aquí resultan:

$$((p \wedge q) \supset r) \wedge (p \supset q) = V \tag{2}$$

$$(p \supset r) = F \tag{3}$$

de la (2) resultan :

$$(p \wedge q) \supset r) = V \tag{4}$$

$$(p \supset q) = V \tag{5}$$

de la (3) resultan:

$$p = V \tag{6}$$

$$r = F \tag{7}$$

de la (4) y de la (7) resulta:

$$(p \wedge q) = F$$

de esta y la (6) resulta $q = F$. Pero ésta y la (6) contradicen la (5). Luego la fórmula original no puede ser F. Luego es válida.
El algoritmo de reducción es particularmente efectivo cuando la fórmula contiene muchos conectivos de implicación.

Ejercicio 1.5.3

Averiguar, por reducción la validez de: $((p \wedge q) \supset r) \supset (p \supset (q \supset r))$

1.6 Equivalencia de fórmulas

Dos fórmulas se llaman **equivalentes** si su tabla de verdad es idéntica. Pero a menudo es tedioso construir las tablas de verdad.
En el álgebra común dos fórmulas son equivalentes si tienen el mismo valor para todos los valores posibles de sus variables. Como en este caso las "tablas de verdad" serían infinitas, la identidad se establece mediante transformaciones algebraicas típicas (propiedades asociativa, conmutativa, distributiva, productos notables y otras identidades conocidas) que se aplican a las fórmulas hasta llevarlas a fórmulas equivalentes idénticas. En general se trata de reducirlas, mediante tales identidades, a ciertas **formas canónicas** como son, para el caso de los polinomios, los productos de binomios de primer grado y trinomios de segundo, o bien a sumas de monomios. En el caso de fórmulas lógicas aplicaremos principios semejantes cuando el método de las tablas de verdad resultara muy tedioso.
Tenemos en primer lugar las siguientes identidades donde hemos indicado la equivalencia lógica por $A \approx B$ es decir, $\models (A \equiv B)$:

$$\begin{array}{ll}
(A \wedge A) \approx (A) \approx (A \vee A) & \text{(idempotencia)} \\
(A \wedge B) \approx (B \wedge A) & \text{(conmutatividad)} \\
(A \vee B) \approx (B \vee A) & \\
((A \wedge B) \wedge C) \approx (A \wedge (B \wedge C)) & \text{(asociatividad)} \\
((A \vee B) \vee C) \approx ((A \vee B) \vee C) &
\end{array}$$

Para tratar con el conectivo $\supset$ consideraremos el orden parcial de las fórmulas que, con las identidades anteriores, definen un **reticulado**. Las fórmulas son entidades entre las cuales están definidas la conjunción y la disyunción con las propiedades indicadas.
Entre dos fórmulas puede haber la relación $A \leq B$ que significa que la fórmula A implica la B para cualquier interpretación, es decir $\models (A \supset B)$. Tal relación tiene las propiedades de una **relación de orden** tal como se puede ver de la relación de implicación:

$$\begin{array}{l}
A \leq A \\
A \leq B \, y \, B \leq A \rightarrow A \approx B \\
A \leq B \, y \, B \leq C \rightarrow A \leq C \\
A \leq B \Leftrightarrow ((A \wedge B) \approx A) \\
A \leq B \Leftrightarrow ((A \vee B) \approx B)
\end{array}$$

donde $X \approx Y$ significa equivalencia lógica, para toda interpretación de X y Y.
Las tres primeras expresan las propiedades usuales de reflexibilidad, antisimetría y transitividad de una relación de orden. Las dos últimas nos dicen que la conjunción de dos elementos cualesquiera del reticulado nos da el "menor" y la disyunción da el "mayor". Dicho en otros términos la conjunción nos da la cota inferior más alta (extremo inferior) y la disyunción la cota superior mas baja (extremo superior). Las 10 propiedades anteriores sirven para definir las propiedades de un reticulado sobre un conjunto cualquiera. Si los elementos del conjunto son las fórmulas lógicas tales relaciones se pueden demostrar a partir de las definiciones de los conectivos vistas en la sección de fórmulas proposicionales.

Ejercicio 1.6.1

. Demostrar, usando tablas de verdad, las 10 propiedades de reticulado de las fórmulas lógicas.
Las fórmulas equivalentes para toda interpretación, como se puede ver fácilmente, corresponden al mismo nodo del reticulado, así que el reticulado en realidad está formado por el conjunto cociente de las fórmulas por la relación de equivalencia lógica $\approx$
En el reticulado de las fórmulas lógicas en particular los elementos tienen las siguientes propiedades que hacen que sea un **Algebra de Boole** :

$$\begin{array}{ll} ((A \wedge B) \vee C) \approx ((A \vee C) \wedge (B \vee C)) & \text{(distributiva)} \\ ((A \vee B) \wedge C) \approx ((A \wedge C) \vee (B \wedge C)) & \\ (A \vee \neg A) \approx V & \text{(complementariedad)} \\ (A \wedge \neg A) \approx F & \text{(involución)} \end{array}$$

Ejercicio 1.6.2

Establecer la analogía entre esta estructura y la definida sobre los conjuntos con las relaciones de unión, intersección, complementación, inclusión y equivalencia. ¿A que son análogos V y F?
Nótese que en todas estas propiedades algebraicas no interviene directamente la implicación. Ella puede ser eliminada de las fórmulas mediante la relación: $(A \supset B) \approx (\neg A \vee B)$.
Principios de dualidad Si en una fórmula lógica sin conectivos $\supset$ intercambiamos los signos $\vee$ y $\wedge$ y las constantes V y F, se obtiene una fórmula lógicamente equivalente. La nueva fórmula se dice dual de la original.
Si además sustituimos cada literal por su negación también la fórmula que resulta es lógicamente equivalente. Es otro tipo de dualidad.

Ejercicio 1.6.3

Demostrar que A es una consecuencia lógica de $H_1, H_2, ..., H_n$ si y sólo si:

$$H_1 \wedge H_2 \wedge ... \wedge H_n \wedge \neg A \approx F$$

o la fórmula dual:

$$\neg H_1 \vee \neg H_2 \vee ... \vee \neg H_n \vee A \approx V$$

La idea de la primera fórmula es que decir que la tesis A es consecuencia lógica de las hipótesis equivale a decir (entre otras maneras posibles):
1) Que suponer todas las hipótesis y la negación de la tesis verdaderas es falso.
2) Que suponer la tesis falsa y todas las hipótesis verdaderas no es verdadero.
Las fórmulas anteriores corresponden a dos formas de deducción utilizadas en Inteligencia Artificial. En la primera se supone A falso y se va combinando ésta con la hipótesis hasta llegar por reducción a una fórmula falsa (deducción hacia adelante). En la segunda se supone A verdadera y se combina con la negación de las hipótesis por reducción hasta llegar a una fórmula verdadera (deducción hacia atrás).

Ejercicio 1.6.4

Si tenemos un conjunto de n proposiciones demostrar que hay 2^n interpretaciones diferentes. Con esto ver que hay 2^{2^n} fórmulas lógicamente diferentes.
Nótese que aunque el número de fórmulas posibles con n proposiciones es infinito se pueden dividir en un número finito de clases de fórmulas equivalentes.

1.7 Cláusulas. Reducción a Formas Normales

En lo que sigue veremos una manera de representar conjuntos de proposiciones en una forma que facilita el cálculo de su consistencia, Para ello hay que introducir los conceptos de cláusula y forma normal conjuntiva.
Llamamos **cláusula** a la disyunción de un número finito de literales.

Ejemplo 1.7.1

$p \vee \neg s \vee r \vee \neg q \vee t$ es una cláusula.
Es obvio que para que una cláusula sea verdadera es necesario y suficiente que contenga al menos un literal verdadero. Para mantener esta proposición en el caso de cláusulas vacías se admite que **una cláusula vacía es falsa**.
Nótese que si en una cláusula eliminamos o agregamos un literal falso su valor de verdad no cambia.
Se llama **forma normal conjuntiva** a la conjunción de un número finito de cláusulas.

Ejemplo 1.7.2

$(p \vee \neg r) \wedge (\neg p \vee s \vee r) \wedge t$
$\emptyset$
Puesto que una conjunción de fórmulas es falsa si y sólo si contiene una fórmula falsa se admite que la conjunción vacía es verdadera. Por lo tanto: **una forma normal conjuntiva vacía es verdadera**. Por otra parte una forma normal conjuntiva que contiene una cláusula vacía es falsa.

Teorema 1.7.1 Toda fórmula puede transformarse en una forma normal conjuntiva equivalente

.
La demostración procede mediante los siguientes reemplazos donde A, B, C son fórmulas:

1. $A \equiv B$ por $(A \supset B) \wedge (B \supset A)$ que elimina los $\equiv$.
2. $A \supset B$ por $\neg A \vee B$ que elimina las implicaciones.
3. $\neg(A \wedge B)$ por $(\neg A \vee \neg B)$ y
   $\neg(A \vee B)$ por $(\neg A \wedge \neg B)$
   que eliminan la negación de conjunciones o disyunciones.

4. $\neg\neg A$ por A.

5. Después de estos reemplazos sólo quedan conectivos $\vee\wedge$ y los $\neg$ quedan delante de proposiciones. Es decir quedan literales unidos por tales conectivos posiblemente con paréntesis que incluyen $\wedge$. Estos se transforman aplicando la propiedad distributiva, por ejemplo:
$$A \vee (B \wedge C) \text{ sustituido por } (A \vee B) \wedge (A \wedge C)$$

La expresión resultante es pues una conjunción de disyunciones de literales.
Todavía pueden hacerse las simplificaciones siguientes:

1. Eliminar literales repetidos dentro de una cláusula.

2. Eliminar cláusulas conteniendo literales opuestos, ya que tales cláusulas son verdaderas y su eliminación no altera el valor de la forma.

   Ver que la forma que resulta, llamada **forma normal conjuntiva pura**, sigue siendo equivalente a la original, pues se han sustituido fórmulas por fórmulas equivalentes.

3. Además se pueden eliminar cláusulas que contengan otras cláusulas que aparezcan en otras partes de la forma.

Ejercicio 1.7.1

Ver que esta última eliminación da una forma equivalente. Dar un ejemplo.

Ejercicio 1.7.2

Ver que si una forma normal contiene la cláusula vacía puede reducirse a la cláusula vacía.

Ejercicio 1.7.3

Una cláusula es válida si y sólo si contiene el opuesto de alguno de sus literales.

Ejercicio 1.7.4

Toda cláusula no vacía es consistente.

Ejercicio 1.7.5

Una forma normal es válida si y sólo si contiene sólo cláusulas válidas

Ejercicio 1.7.6

Demostrar que si en una forma normal se elimina una cláusula válida se obtiene una forma equivalente.
Mientras se hacen las transformaciones que llevan a la forma normal siempre se puede reemplazar una cláusula válida por V, y eliminar cláusulas repetidas y literales repetidos en una cláusula. De todos modos la transformación puede ser muy trabajosa.

Ejercicio 1.7.7

Reducir a forma normal conjuntiva:
$((p \supset q) \wedge r) \supset (q \equiv r)$
decir si es válida o consistente. Verificar haciendo su tabla de verdad.
En lo que sigue representaremos la forma normal por la disyunción de sus cláusulas o por el conjunto cuyos elementos son las cláusulas. Así la conjunción:

$$(p \vee q) \wedge (p \vee r) \wedge (\neg p \vee r) \wedge (\neg r) \wedge (q \vee r)$$

puede representarse por el conjunto:

$$\{p \vee q, p \vee r, \neg p \vee r, \neg r, q \vee r\}.$$

Ejercicio 1.7.8

Decir cuales serían los conceptos duales de cláusula (se llama **cubo**) y de forma normal conjuntiva (se llama **forma normal disyuntiva**).

1.8 Algoritmos de Quine y de Davis-Putnam para formas normales

El algoritmo de Quine para ver la validez o consistencia de una fórmula se hace muy simple si se aplica a formas normales. Para ver la **validez** habría demostrar que cada cláusula es una tautología lo cual es trivial (ver Ejercicio 1.7.3). Para ver la **consistencia** de una forma normal **pura** formada por la conjunción del conjunto S de cláusulas, se introducen las siguientes notaciones:

> Sea un conjunto S de cláusulas.
>
> Dada una proposición p llamamos:
>
> S_p el conjunto de las cláusulas que contienen el literal p;
> $S_{\neg p}$ el de las que contienen $\neg p$
> $S'' = S \setminus (S_p \cup S_{\neg p})$ el de las cláusulas que no contienen p ni $\neg p$.

El algoritmo de Quine consiste en lo siguiente:

- Si $S = \emptyset$ entonces S es consistente.
- Si $S = \{F\}$ entonces S es inconsistente.
  Si no:
  - Tomar una proposición p de alguna cláusula de S.
  - Determinar los conjuntos S_p , $S_{\neg p}$ y $S'' = S \setminus (S_p \cup S_{\neg p})$
  - Quitar de las cláusulas de S_p el literal p, obteniendo el conjunto de cláusulas S'_p .

– Quitar de las cláusulas de $S_{\neg p}$ el literal $\neg p$, obteniendo el conjunto de cláusulas $S'_{\neg p}$.

Nota: Si S'' es vacío, no hay que agregarlo ya que vale V. Al quitar el literal p de la cláusula p, queda la cláusula vacía o sea F. Por otra parte recordar que un conjunto vacío equivale a V
Se verá enseguida que el conjunto original S (y por tanto la forma normal) es inconsistente si y sólo si lo son los dos conjuntos o formas derivadas: $S'_p \cup S''$ y $S'_{\neg p} \cup S''$.
Cada uno de ellos se puede tratar con el mismo algoritmo, formando un árbol binario.
Nótese que estos dos conjuntos no contienen el literal p ni el $\neg p$. Por aplicación recursiva del algoritmo se llega a algún conjunto inconsistente (por contener la cláusula vacía que es F) y no hace falta analizarlo más. El algoritmo termina cuando todas las ramas llevan a un conjunto inconsistente y entonces el S original es inconsistente. Una rama termina si se llega a un conjunto en que en no hay un par de cláusulas con literales opuestos. Tal conjunto es consistente. Un modelo se obtiene dando el valor V o F a un literal en cada cláusula, para hacerla verdadera,lo cual no puede alterar el valor de verdad de las otras cláusulas pues no pueden contener el opuesto de tal literal.
La justificación del procedimiento se basa en el teorema siguiente, en el cual se usa la notación anterior:

Teorema 1.8.1 Si $p = V$**,** S **es equivalente a** $S'_{\neg p} \cup S''$ **. Si** $p = F$ **entonces** S **es equivalente a** $S'_p \cup S''$

Supongamos $p = V$, entonces podemos eliminar todas las cláusulas que contienen p, ya que son verdaderas y al eliminarlas no cambia el valor de la forma. Además, por ser $\neg p$ falsa este literal puede eliminarse de todas las cláusulas sin alterar sus valores (y por ello sin cambiar el valor de la forma). Con esto el conjunto de cláusulas original $S = (S_p \cup S_{\neg p} \cup S'')$ se transforma en el $S'_{\neg p} \cup S''$. Si $p = F$ se ve, por un razonamiento análogo, que el S original equivale a $S'_p \cup S''$. Si S es inconsistente es falso para todo valor de las proposiciones, en particular de p, así que ambos conjuntos equivalentes también lo son. Recíprocamente, si alguno de estos no es inconsistente para el correspondiente valor de p y los valores de las otras variables que lo hacen verdadero el S es verdadero.

Ejercicio 1.8.1

Reducir mediante el algoritmo de Quine y ver la consistencia de: $(p \vee q) \wedge (p \vee r) \wedge (\neg p \vee r) \wedge (\neg r) \wedge (q \vee r)$.
Ponerlo antes en forma de conjunto.
El algoritmo anterior se puede representar como un árbol tal como se vió en 1.5 poniendo en el extremo de la flecha p la forma derivada $S'_{\neg p} \cup S''$ y en el de la flecha $\neg p$ la forma derivada $S'_p \cup S''$.

Ejercicio 1.8.2

Ver la consistencia de:
$(p \vee q) \wedge (p \vee r) \wedge (\neg q \vee \neg r) \wedge (r)$.

$(p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee \neg r) \wedge (r)$
Poner el desarrollo en forma de árbol.
Puede abreviarse el proceso seleccionando adecuadamente el literal para formar las dos formas derivadas. Son útiles las reglas siguientes:
Elegir el literal p :

1. Si S contiene p o contiene a $\neg p$ como cláusula.

2. En S aparece sólo uno de los dos literales p o $\neg p$.

Tal forma abreviada del algoritmo de Quine se debe a Davis y Putnam.

Ejercicio 1.8.3

Ver que en el primer caso la cláusula derivada es inconsistente y no hace falta analizarla más. Hay que ver sólo la otra. En el segundo caso ver que si ocurre por ejemplo p (pero no $\neg p$) sólo hay que ver la consistencia de $S'_p \cup S''$ pues la consistencia de la otra forma depende de la de S''.

Ejercicio 1.8.4

Considere la forma de razonamiento:
Adolfo está en el edificio (e). Si Adolfo está en el edificio entonces entró ayer(a) o bien entró hoy (h) o ambas cosas. Si entró ayer (a) entonces tuvo que ver al nuevo portero (p). Si entró hoy (h) también tuvo que verlo (p). Luego es consecuencia lógica de las anteriores que Adolfo vió al nuevo portero (p). Expresar el razonamiento en notación lógica usando los conectivos $\supset, \vee, \models$ y las proposiciones e,a,h,p,$\neg p$. Reducir la expresión lógica a forma normal conjuntiva. Hallar si ese tipo de razonamiento es consistente mediante el algoritmo de Davis y Putnam.
Tal forma de razonamiento se llama de **de prueba por casos** y se expresa en general:
$\{h, h \supset (p \vee q), p \supset c, q \supset c\} \models c$

1.9 Principio de Resolución. Resolventes.

Es inmediato que se cumple que $\{A \vee X, B \vee \neg X\} \models A \vee B$.
La consecuencia lógica se obtiene mecánicamente eliminando X de la primera cláusula y $\neg X$ de la segunda y formando una cláusula con ambas. En esto se basa el siguiente:

Teorema 1.9.1 Sean C_1 y C_2 cláusulas de la forma normal S y t un literal tal que $t \epsilon C_1$ y $\neg t \epsilon C_2$, entonces la cláusula $R = (C_1 \backslash t) \cup (C_2 \backslash \neg t)$ es una consecuencia lógica de S

.
La cláusula R se llama **resolvente** de las cláusulas C_1, C_2 respecto de t.

Corolario 1.9.1

El nuevo conjunto $S \cup R$ es equivalente al anterior pues se le agrega una cláusula que es siempre verdadera si lo son C_1 y C_2 y si alguna de ellas es falsa el S es inconsistente cualquiera sea R.

Ejercicio 1.9.1

Demostrar el teorema anterior.
La aplicación de este teorema a la prueba de consistencia de una forma normal pura se lleva a cabo por el procedimiento siguiente:

- Repetir mientras S no contenga la cláusula vacía:
  - Encontrar C_1 y C_2 t tales que $t \epsilon C_1$ y $\neg t \epsilon C_2$.
  - Calcular la resolvente $R = (C_1 \backslash t) \cup (C_2 \backslash t)$
  - Reemplazar S por $S \cup R$

Nótese que en cada paso S va creciendo proporcionando más y más cláusulas para hacer nuevas combinaciones. Si en un punto no hay más resolventes nuevos que hacer, entonces la S original es consistente. Si alguno de los resolventes obtenidos es F, entonces la S original es inconsistente.

Ejercicio 1.9.2

Justificar los criterios anteriores de terminación.

Ejercicio 1.9.3

Hallar la consistencia mediante resolventes de las siguientes formas normales:
$(p \vee q) \wedge (p \vee r) \wedge (\neg q \vee \neg r) \wedge (\neg p)$
$(p \vee \neg q) \wedge (p \vee r) \wedge (\neg p \vee q) \wedge (\neg q)$

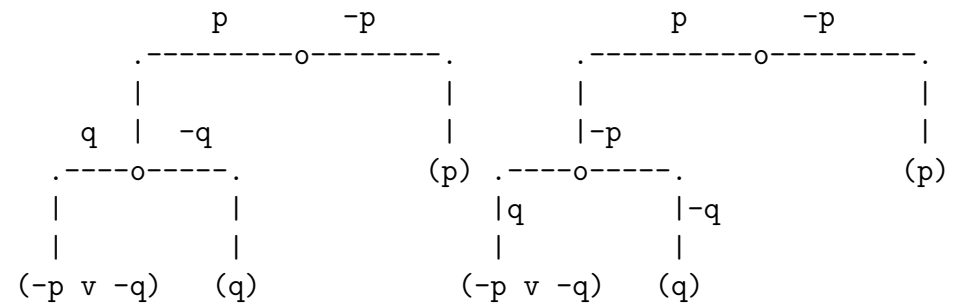
Ejercicio 1.9.4

Demostrar por resolución el principio de deducción por casos.
$\{h, h \supset (p \vee q), p \supset c, q \supset c\} \models c$ (Eliminar los $\supset$).Poner hipótesis y negación de conclusión numeradas de 1 a 5. Resolventes de 1 y 2, 3 y 5, 4 y 5, 6 y 7, 8 y 9).
En el proceso se pueden eliminar cláusulas de la forma $p \vee \neg p$ que son siempre V. Puede evitarse generar las resolventes repetidas. Con todo puede ser difícil agotar todos los casos y por una mala elección de los componentes de los resolventes el proceso se puede prolongar indefinidamente.
Puede demostrarse que si el conjunto de cláusulas S es inconsistente y contiene los resolventes de sus elementos entonces S contiene la cláusula vacía. La demostración resulta de generalizar la representación en árbol de una construcción de las resolventes en la forma siguiente:

- Generar el árbol semántico con las proposiciones de S deteniéndose en una rama hasta que se pueda poner en la hoja una de las cláusulas de S que resulte falsa al seguir la rama.
- Proseguir el proceso anterior hasta que se hayan puesto todas las cláusulas de S en alguna hoja (puede haber cláusulas repetidas). Si no se pueden generar todas S es consistente.
- subiendo de cada par de arcos a partir de las hojas, poner en el nodo de donde parten una cláusula formada así:
  - Si los arcos corresponden a las proposiciones s y $\neg s$ y las cláusulas contienen estos literales poner en el nodo el resolvente.
  - Si alguna de las cláusulas no contiene ni el literal del arco ni su negación ponerla en el nodo.

Ejemplo 1.9.1

Árbol semántico del conjunto de proposiciones:
$S = \{p, \neg p \vee \neg q, \neg p \vee r, q\}$



Vemos pues que si S es inconsistente el árbol siempre puede construirse pues hay un camino (interpretación) que tiene los opuestos de los literales de cualquier cláusula de S. Tal interpretación (con las del camino y cualquier valor en las otras proposiciones) hace S falsa. Si se puede construir el árbol el S es inconsistente. Las resolventes obtenidas en los nodos son falsas pues son consecuencia lógica de los sucesores o son idénticos a un sucesor. Se llega así al nodo raíz que resulta falso pues sus arcos son p y $\neg p$ de modo que sus antecesores son $(\neg p)$ y p como es una resolvente y es consecuencia lógica de las hojas el conjunto de éstas es inconsistente. Vemos pues que **el método de resolución siempre permite verificar si un conjunto S de fórmulas es inconsistente**.

1.10 Cláusulas de Horn.

Se denominan **cláusulas de Horn** las que tienen a lo más un literal positivo.

Ejemplo 1.10.1

1. $\neg p \vee \neg q \vee \neg r$
2. $\neg p \vee \neg q \vee r$
3. p
4. $\neg r$
5. $\neg p \vee q \vee r$

En los ejemplos: La cláusula 3 se llama unidad positiva y expresa un **hecho**.
La 2, con un solo literal positivo se llama **definida**.
La 1 y la 4 son **negativas**.
La 5 no es una cláusula de Horn.
Las cláusulas de Horn, en especial las definidas, son muy importantes en el procesamiento de expresiones que representan inferencias desde un conjunto de hipótesis a una conclusión. En efecto, las expresiones de tal tipo son de la forma:
$h_1 \wedge h_2 \wedge h_3 \supset c$ o sea:
$\neg h_1 \vee \neg h_2 \vee \neg h_3 \vee c$.

Si las hipótesis en vez de ser proposiciones son ellas mismas inferencias y la conclusión es una cláusula o en particular un hecho el problema consiste en ver la consistencia de un conjunto (o forma conjuntiva) de cláusulas de Horn. Si la inferencia de hipótesis a conclusión es **válida** , entonces la conclusión se deduce de las hipótesis; si no es válida no se puede hacer tal inferencia. Como los algoritmos prueban mejor la inconsistencia de un conjunto de cláusulas de Horn, es más usual formar un conjunto de cláusulas de Horn con las **hipótesis** y la **negación de la conclusión**

Ejemplo 1.10.2

Sean las siguientes afirmaciones sobre el diagnóstico de una enfermedad:

g=“la garganta está inflamada”
n=“hay congestión nasal”
r=“hay resfrío”
f=“hay fiebre”

Supongamos que se aceptan las hipótesis siguientes (traducirlas al lenguaje común):

1. $g \wedge n \supset r$
2. $r \supset f$
   y para un cierto paciente se observa que:
3. $f \wedge n$
   queremos saber si es válida la conclusión:

4. r

Esto significa demostrar la inconsistencia del conjunto de cláusulas de Horn:

$$\neg g \vee \neg n \vee r, \neg r \vee f, f, n, \neg r$$

Es decir, las reglas 1 y 2 y el hecho 3 y la conclusión (o pregunta que se hace) puesta como el hecho 4 se transforman en un conjunto de cláusulas de Horn.
En este caso, una simple inspección muestra que el conjunto es consistente, de modo que la conclusión r no se puede extraer de las hipótesis. En caso de tener muchas hipótesis es necesario tener un algoritmo que muestre la consistencia o inconsistencia.

1.11 Método de resolución de conjuntos de cláusulas de Horn

El método de la resolvente adopta, en el caso de cláusulas de Horn, la siguiente forma:
Sea S_0 un conjunto de cláusulas de Horn sin tautologías (no contienen formas tipo $r \vee \neg r$).

Ponemos: $S \leftarrow S_0$
Repetir mientras F $\notin S$ y se pueda seleccionar:
Seleccionar p y C tales que:
p es una unidad positiva de S (un hecho)
C es una cláusula que contiene $\neg p$
Calcular la cláusula resolvente $R = C \setminus \neg p$
Reemplazar S por $(S \setminus C) \cup R$

Nótese que en cada paso se suprime un literal de una cláusula pues se quita C y se agrega C sin $\neg p$.
El algoritmo termina cuando se llega a una cláusula vacía (proveniente de dos literales opuestos), en cuyo caso el sistema es inconsistente, o hasta que no quedan unidades positivas tales que tengan un literal opuesto en otras cláusulas. En este último caso el sistema es consistente.

Ejercicio 1.10.1

Probar, usando el algoritmo de resolución que el sistema del Ejemplo 1.10.2 es consistente. Ver que si se le agrega la cláusula g ("hay garganta inflamada") se vuelve inconsistente y la conclusión: "hay resfrío" es verdadera.
La demostración de la validez del algoritmo anterior es muy sencilla.
Si no se llega a una cláusula vacía y el algoritmo se detiene por no encontrar una unidad positiva que tenga un literal opuesto en otras cláusulas podemos siempre dar una interpretación del conjunto que resulte verdadera. Basta poner V en las unidades positivas (lo cual hace V otras cláusulas pero no puede hacer F ninguna cláusula pues en ninguna hay sus negaciones). Además las que no son unidades positivas se ponen F. Si aparecen

en otras cláusulas en forma de literal negativo, esto hace tales cláusulas V. Si aparecen positivas deben estar acompañadas por literales negativos (por ser cláusulas de Horn) los cuales no tienen unidades positivas correspondientes (si no ya las hubiéramos reducido) y por lo tanto se les asignó valor F, lo cual hace las cláusulas V.
Por otra parte si el algoritmo termina porque aparece una resolvente falsa, como la resolvente es consecuencia lógica de S, S es inconsistente.
Así que las condiciones de terminación deciden sobre la consistencia en la forma dicha anteriormente. En general se ve que en cada paso del algoritmo el conjunto de cláusulas que resulta S' es equivalente al anterior S. Vimos en efecto, en el método de resolución (sección 1.9) que S y $S \cup R$ son equivalentes en general. Luego lo son también en este caso particular de calcular R (usando las unidades positivas). Por otra parte $S \cup R$ es equivalente al conjunto que queda en el paso siguiente: $(S \setminus C) \cup R$ ya que C es una consecuencia lógica de R (si R es V también lo es C que es $R \cup \neg p$. Entonces: - Si $S \cup R$ es V también lo es R. Al quitarle a S la cláusula verdadera C queda $S \cup R$ verdadera y $(S \setminus C) \cup R$ también es V. - Si $S \cup R$ es F es o R falsa o S falsa (o ambas). Si R es falsa también lo es $(S \setminus C) \cup R$. Si es S falsa y R verdadera, entonces como C es verdadera $S \setminus C$ es F y por lo tanto $(S \setminus C) \cup R$ es F.
Queda pues probada la equivalencia entre un conjunto de cláusulas de Horn S y el que resulta después de un paso en el algoritmo.
Por último observamos que si al final queda un conjunto S_f no vacío de cláusulas (o lo que es lo mismo que no contiene la cláusula vacía) entonces es consistente (y por ello lo es el S_0 original. Tal S_f tiene pues un modelo. Se lo llama **modelo canónico** de S_0.
Tal modelo que como vimos sólo tiene como verdaderas las cláusulas que son unidades positivas y no tienen su negativo en otras cláusulas es el que asigna el menor número de valores V que cualquier otro modelo. En efecto si hacemos F cualquiera de las unidades positivas el S_0 se hace falso.
Se puede definir **modelo mínimo** de un conjunto S de cláusulas de Horn en la forma siguiente:
A toda interpretación I de S le corresponde un conjunto de proposiciones I_p formado por aquellas proposiciones de S que se toman como verdaderas. Se puede definir entonces la intersección de las interpretaciones como la interpretación que toma como V las proposiciones que están en la intersección de los correspondientes conjuntos. La intersección de dos modelos no es, en general, un modelo. Si lo es, el número de cláusulas verdaderas en la intersección de los modelos es menor (o igual) que en los que se intersectan. Puede definirse el **modelo minimal** como el que resulta de intersectar todos los modelos posibles de S, siempre que tal intersección sea un modelo. No está garantizado que esto último ocurra. Pero si ocurre el modelo minimal es único. Por otra parte el algoritmo de reducción nos dice que si S es consistente se puede construir una interpretación que asigna valor V a un número mínimo de proposiciones. Este modelo debe coincidir con el minimal. Luego:

Teorema 1.11.1 Todo conjunto finito consistente de cláusulas de Horn admite un modelo mínimo.

Véase que este teorema nos permite asegurar que, en un conjunto de conocimientos expresado por cláusulas de Horn, hay un conjunto de hechos cuya verdad asegura la verdad de cualquier conclusión extraída del conjunto. El algoritmo de reducción permite encontrar tales hechos.

1.12 Consistencia de conjuntos infinitos de fórmulas

Es importante ver las propiedades de consistencia de un conjunto de fórmulas cuando se quita la restricción de que sea finito.
Sea P un conjunto con un número finito o numerable de proposiciones. Por ejemplo: $P = \{r, p, h, ...\}$.
Sea H el conjunto de todas las fórmulas que se pueden formar a partir de las proposiciones en P.
Sea $S \subset H$ un conjunto de fórmulas de H (puede ser infinito). Se dice que S es **finitamente consistente** si todos los subconjuntos finitos de S son consistentes. $S \subset H$ se dice **maximal** si además de ser finitamente consistente para toda fórmula $A \varepsilon H$ contiene A o $\neg A$.
Nótese que si H es infinito y S es maximal, también S es infinito y contiene todas las fórmulas de H afirmadas o negadas. Pero, a diferencia de H, S no puede contener una fórmula y su negación.
Veamos como se relaciona esta maximalidad con la interpretación.
Toda interpretación (modelo) I sobre el conjunto P de proposiciones determina un conjunto maximal S_I. Basta poner en S_I las fórmulas de H que se hacen verdaderas por la interpretación I.

Ejercicio 1.12.1

Demostrar que el S_I así definido es maximal.
Se puede demostrar el inverso , es decir, el siguiente:

Teorema 1.12.1 Todo conjunto maximal S tiene un modelo único.

Veamos primeramente que si hay un modelo I, tal modelo es único.
Por ser S maximal contiene, en particular, cada proposición o su negación. Es decir que para toda p, S debe contener a p o a $\neg p$. Para que I sea un modelo debe ser, si S contiene p entonces $I(p) = V$ y si no lo contiene debe contener $\neg p$ y ser $I(\neg p) = V$ o sea $I(p) = F$. Todo otro modelo debe asignar esos mismos valores a las proposiciones, así que es idéntico a I. Falta ver que tal I es realmente un modelo, es decir, que no sólo hace verdaderos los literales sino también las fórmulas de S.
Sea A una fórmula cualquiera de S. Veremos que $I(A) = V$ con la I antes definida.
Sea Q^+ el conjunto de las proposiciones que aparecen en A y Q^- el conjunto de las negaciones de tales proposiciones. Formemos los conjuntos $R^+ = Q^+ \cap S$ de las proposiciones positivas de la fórmula A que están en S y el $R^- = Q^- \cap S$ de las negativas.

Por ejemplo si fuera $A = \neg p \vee q \vee r$ se tendría:
$Q^+ = \{p, q, r\}$ $Q^- = \{\neg p, \neg q, \neg r\}$, R^+ y R^- dependen de cuales literales haya en S.
Se ve que I asigna valores V a los literales de R^+ y R^- Por otra parte el conjunto de fórmulas $R^+ \cup R^- \cup A$ es consistente pues es un subconjunto finito de S que es finitamente consistente. A debe pues ser consistente, o sea verdadero para alguna interpretación. Pero A tiene las mismas proposiciones que $R^+ \cup R^-$ y la I considerada es tal que hace verdadera $R^+ \cup R^-$. Luego tal I debe hacer verdadera A.

Teorema 1.12.2 Un conjunto S de fórmulas es consistente si y sólo si todos sus subconjuntos finitos son consistentes.

La condición de que todos los subconjuntos finitos sean consistentes es obviamente necesaria pues si alguno no lo es, tampoco lo sería S.
Para ver que es suficiente, es decir, para ver que S, finitamente consistente, es consistente (lo cual no está asegurado si S es infinito) habría que construir un modelo de S o un modelo de S_m que contenga a S y que sea maximal (lo cual significa que sea finitamente consistente y que para toda fórmula $A \varepsilon H$ contenga A o $\neg A$.) con lo cual admitiría un modelo por el Teorema 1.12.1, que sería también modelo de $S \subset S_m$.
Para construir tal S_m consideramos todas las fórmulas generadas a partir de un conjunto numerable P de proposiciones. Tal conjunto H de fórmulas es numerable (pueden ordenarse por ejemplo en orden lexicográfico) $A_1, A_2, .., A_n, ..$
Formemos los subconjuntos S_n de H en la siguiente forma:
$S_0 = S$
$S_{n+1} = S_n \cup A_n$ si todos los subconjuntos de $S_n \cup A_n$ son consistentes.
$S_{n+1} = S_n \cup \neg A_n$ si no todos los subconjuntos de $S_n \cup A_n$ son consistentes.

Puede verse que todos los S_n así formados son finitamente consistentes. Se demuestra por inducción:
Esto es cierto para S_0 pues $S_0 = S$ que es, por hipótesis, finitamente consistente.
Supongamos que S_n es finitamente consistente. Hay que ver que lo es S_{n+1} .
Cuando se hace $S_{n+1} = S_n \cup A_n$ evidentemente lo es S_{n+1} pues así hemos definido S_{n+1}
Cuando se hace $S_{n+1} = S_n \cup \neg A_n$ esto se hace porque existe un subconjunto finito $S' \subset S_n$ tal que $S' \cup A_n$ (contenido en $S_n \cup A_n$) es inconsistente.
Sea S'' un subconjunto finito cualquiera de $S_{n+1} = S_n \cup \neg A_n$. Entonces el conjunto $(S' \cup S'') \setminus \neg A_n$ contenido en $S_n \cup A_n$ es consistente. En efecto S'' es un subconjunto finito de S_n tal vez unido con la fórmula $\neg A_n$ (pues $S'' \subset S_{n+1} = S_n \cup \neg A_n$) al extraerle $\{\neg A_n\}$ queda un subconjunto de S_n.
Por otra parte $S' \subset S_n$. Entonces S' y S'' son consistentes pues son finitos y están contenidos en S_n que por hipótesis de la inducción es finitamente consistente. Por lo tanto lo es $(S' \cup S'') \setminus \neg A_n$.
Sea I un modelo de $(S' \cup S'') \setminus \neg A_n$ que vimos era consistente. La $I(A_n)$ no puede ser verdadera pues entonces lo sería la disyunción $I(S' \cup A_n)$ que supusimos inconsistente. Luego es $I(A_n) = F$ $I(\neg A_n) = V$. Pero I es un modelo de $(S' \cup S'')$ pues este es un subconjunto que resulta del $(S' \cup S'') \setminus \neg A_n$, modelo de I al que se agrega $\neg A_n$ siendo

$I(\neg A_n) = V$. Luego I es un modelo de S' el cual era un conjunto finito cualquiera de S_{n+1}. Luego S_{n+1} es finitamente consistente.
La unión S_m de todos los S_n es maximal, pues es finitamente consistente (lo es cada S_n) y, por la forma en que se construyen los S_n por cada fórmula A_n de H contiene A_n o $\neg A_n$. Todo subconjunto finito de esta unión es también un subconjunto finito de algún S_n (finitamente consistente) así que es consistente. S_m es pues finitamente consistente.
Por ser $S_m = \cup S_m$ maximal tiene por el teorema 1.12.1 un modelo el cual es también modelo de $S \subset S_m$

Capítulo 2

Lógica de predicados

2.1 Partes de la proposición

Consideremos las proposiciones siguientes:

p="Todo mamífero es vertebrado"
q="Algún mamífero es vertebrado"
h="Ningún mamífero es vertebrado"
r="Ningún mamífero es no vertebrado"
s="Algún no mamífero es vertebrado"
t="Existe algún mamífero no vertebrado"
u="Todo mamífero es no vertebrado"
v="Todo vertebrado es mamífero"
w="Algún vertebrado es no mamífero"

Ejercicio 2.1.1

Suponga que las clases "mamífero" "vertebrado" y sus complementos son no vacías y que p=V. Decir cuales de las proposiciones anteriores resultan:

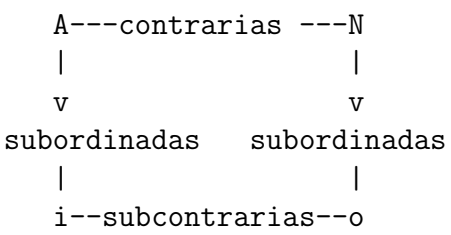
- falsas
- verdaderas
- no decidibles con la información dada

Nótese que cuando deducimos, en el ejemplo anterior, que si p es verdadera entonces q es verdadera podemos expresar esto usando la notación del cálculo proposicional diciendo que $p \supset q$. Pero es claro que esta fórmula no es válida sino contingente, mientras que vemos que nuestra deducción es absolutamente válida. Es decir, el tipo de deducción que hemos hecho no se puede expresar mediante el cálculo proposicional. Observando un poco nuestra inferencia vemos que ella no sólo depende de que sea p=V, sino de la relación entre las partes de las proposiciones p y q. Al decir que el predicado "vertebrado" se aplica a "todo", "mamífero" lo aplicamos con seguridad a "algún" "mamífero". La parte de la Lógica que trata deducciones de este tipo debe pues tratar las proposiciones no

como átomos, como en el cálculo proposicional, sino que debe entrar en sus partes (sujeto y predicado) y en las relaciones de operadores tales como "todo" y "algún" o "existe". Entonces debe determinar que relaciones de estos elementos debe haber en proposiciones tales como p, q, t para que se pueda concluir que de la verdad de una se puede inferir la verdad o falsedad de otra. Aristóteles, en el siglo IV a.C. encontró este tipo de relaciones determinando que hay cuatro tipos de proposiciones:

- A universal afirmativa: Todo S es P.
- i particular afirmativa: Algún S es P. O bien: Existe un S que es P.
- N universal negativa: Todo S es no P. O bien: Ningún S es P.
- o particular negativa: Algún S es no P. O bien: Existe un S que no es P.

Las letras A i N o fueron introducidas por los lógicos medievales con la regla mnemotécnica deducida de las palabras latinas: **A**ff**i**rmo y **N**eg**o**. Se hizo el cuadro siguiente para ver sus relaciones:



Las que están en diagonal son contradictorias. Las contrarias no pueden ser ambas verdaderas, pero sí ambas falsas.
Si una A o N es verdadera su subordinada también lo es.
Las subcontrarias pueden ser ambas verdaderas pero no ambas falsas.
La contradictorias (A y o o bien N y i) no pueden tener el mismo valor de verdad, cuando una es V la contradictoria es F y viceversa.

Ejercicio 2.1.2

Las reglas anteriores especifican las deducciones posibles. Ver que en el ejemplo anterior hemos usado estas reglas.

Ejercicio 2.1.3

Representar las clases "mamíferos" y "vertebrados" por diagramas de conjuntos y ver en ellos las reglas de deducción.
Otro ejemplo de deducción basado en la partición sujeto-predicado y en los conectivos "todo" "alguno" (o "existe") son los silogismos, también analizados por Aristóteles. Durante la Edad Media fueron considerados como la principal forma de argumentación.
Un ejemplo de silogismo es:

Todos los mamíferos son vertebrados.
Todos los gatos son mamíferos.
Todos los gatos son vertebrados.

Las dos primeras proposiciones se llaman **premisas** la tercera es la **conclusión**. Si cambiamos los dos últimos "todos los" por "algunos", también la inferencia es correcta. También si cambiamos el primer y el último "todos los gatos..son" por "ningún gato..es" (es decir la tipo A por la tipo N) y ponemos la segunda premisa como conclusión la inferencia sigue siendo válida. Aristóteles encontró todos los casos verdaderos y dedujo unos de otros usando las reglas de inferencia dadas en el cuadro anterior y algunas reglas de inversión: cambio de sujeto por predicado. Por ejemplo sustituir la última por "Algunos vertebrados son gatos".
Los silogismos se pueden representar por conjuntos ya que el verbo "es" significa pertenecer a la clase descrita por el predicado.

Ejercicio 2.1.4

representar los conjuntos "vertebrado", "mamífero" y "gato" y ver cuales inferencias por silogismo pueden deducirse.
Otras inferencias pueden hacerse cuando se han definido ciertas propiedades de algunas funciones o relaciones. Así la inferencia:

"París está en Francia"
"Francia está en Europa"
"París está en Europa"

la inferencia resulta correcta cuando definimos "está en" como una relación que tiene la propiedad transitiva.
Otro ejemplo:

"Juan es padre de Adolfo"
"Adolfo es hijo de Juan"

La inferencia resulta correcta cuando especificamos la naturaleza de las relaciones "es padre de" y "es hijo de" diciendo que una es inversa de la otra.
Otro ejemplo:

"Todos los perros ladran"
"Fido es un perro"
"Fido ladra"

La regla de inferencia para concluir la tercera proposición dice que: Si "todos" se aplica a una implicación de predicados aplicables a individuos, esto permite inferir que si a un individuo particular se le aplica el antecedente de la implicación se le aplicará también el consecuente. Si para todo animal "todos" se aplica a "si es perro entonces ladra" y a "Fido se aplica "perro" entonces también se le aplica "ladra"
Para representar conocimientos mediante proposiciones y manejarlos extrayendo inferencias de ellos, sería pues deseable un tipo de cálculo que pudiera representar los tipos de

proposiciones e inferencias indicados antes. Y vimos que para ello no basta el cálculo proposicional, pues las inferencias dependen de las relaciones entre sujetos y predicados de las proposiciones. Tal parte de la lógica se llama **cálculo de predicados**.

2.2 Elementos de representación del cálculo de predicados

El cálculo de predicados se construye con los elementos siguientes:

1. **Constantes Individuales**. Designan objetos particulares: París, Europa, Juan, Fido. Los representaremos por las primeras letras del abecedario: a, b, c,..,j. En computación de conocimientos se usan los nombres completos o con un número: mesa, libro_2, perro_3, etc..

2. **Variables o Clases**. Designan clases universales o conjuntos de individuos: ciudad, perro, hombre. Se denotan con las últimas letras del abecedario: v,x,y,z. En computación de conocimientos se usan los nombres completos.

3. **Funciones o Relaciones**. Se expresan como en Matemáticas. Por ejemplo "es padre de a" se denota p(a). En el ejemplo anterior si a es Adolfo y j es Juan se tiene: j=p(a). En computación puede ponerse Juan=Padre(Adolfo).

   Puede haber funciones de varias variables. Por ejemplo en una organización jerárquica en árbol dos individuos cualesquiera c y d tienen un "jefe común más próximo" que es el valor de la función: j(c,d).

   Así, por ejemplo:

   jefe_común_más_próximo(Blas,Juan) puede valer José si suponemos que José es jefe inmediato de Blas y Pedro, y que Pedro lo es de Juan.

   Las funciones de cero variables se denotan por su sólo nombre. Son símbolos con un valor fijo. Por lo tanto son constantes individuales como las descritas en 1.

   Las funciones se dicen 0-arias, uni-arias, bi-narias,.., n-arias, según el número de variables. Una expresión $f(x)$ o $p(x, y)$ de una o más variables se denomina **forma funcional**. La f o p solas se denominan **constantes funcionales**.

4. **Funciones Predicativas**. Expresan características de los objetos, o, lo que es lo mismo, pertenencia a un conjunto. Pueden tener solamente los valores V o F. Las escribiremos con mayúsculas.

   Ejemplo: G(x) puede expresar "x es un gato". Si x vale Misu (cierto gato) es G(Misu)=V mientras que si Juan es una persona G(Juan)=F. En computación se pueden usar nombres comenzados por mayúsculas: como Gato(x) o Gato(Misu).

   Las funciones predicativas pueden tener más de una variable. Así, por ejemplo, si M es "más joven que", entonces M(x,y) expresa que "x es más joven que y". Así si M(Carlos, María) es F si Carlos es de igual edad o más viejo que María.

Las funciones predicativas 0-arias son constantes lógicas que pueden tomar el valor F o V. Son pues las proposiciones del Cálculo Proposicional. Este resulta así un caso restringido del Cálculo de Predicados.

Los nombres G, M, Gato se llaman **constantes predicativas**. Las formas como G(x) o M(x,y) se llaman **formas funcionales predicativas**

No hay variables cuyos **valores** sean funciones. El cálculo con esta restricción se llama **Cálculo de predicados de primer orden**. Los argumentos de una función, sea ella común o predicativa pueden ser también formas funcionales. Así si H expresa "es un hombre" y p es la función "padre" entonces H(p(j)) expresa que el padre de j es un hombre.

Se introducen por conveniencia y abreviatura las siguientes definiciones:

1. **Término** es una variable o una función (común o predicativa). En particular, si no hay argumentos, puede ser una constante o una proposición común.

2. **Igualdades** son expresiones del tipo a=m donde a y m son términos. Mas adelante definiremos rigurosamente la igualdad. Por ahora diremos que son nombres diferentes para la misma variable o función (en particular función 0-aria o constante individual). Si x="animales que amamantan a sus crías" y z="mamíferos" es x=z.

   Si c="Cervantes" y a="autor de El Quijote" es c=a. Por último definimos:

3. **átomos**. Son funciones predicativas o igualdades. Un átomo tiene un valor V o F.

   M(x,y), c=a, El_pájaro_es_gris son átomos.

   La última es una función predicativa 0-aria y equivale a una proposición.

2.2.1 Construcción de fórmulas

Para construir las fórmulas del cálculo de predicados usamos los operadores $\vee, \wedge, \neg, \supset, \equiv$ del cálculo proposicional, e introducimos dos nuevos que llamamos **cuantificadores**:

- $\forall$ seguido de una variable, por ejemplo x :

  $\forall x$ se lee"para todo x."; usualmente lo sigue una fórmula. Si esta tiene la variable x se dice que x es una variable **ligada** y por cada valor que le demos a la x del $\forall x$ debemos darle el mismo a la x de la fórmula. El $\forall$ se llama **cuantificador universal**.

- $\exists$ seguido de una variable, por ejemplo x :

  $\exists x$ se lee"existe algún x."; usualmente lo sigue una fórmula en la cual puede aparecer la x como variable ligada. El $\exists$ se llama **cuantificador existencial**

Con todos estos elementos se construyen todas las fórmulas del cálculo de predicados usando las reglas siguientes:

1. Un átomo es una fórmula.

2. Si x es una fórmula, también lo es $\neg x$
3. Si X e Y son fórmulas también lo son:
   $X \vee Y, X \wedge Y, X \supset Y, X \equiv Y$
4. Si A es una fórmula también lo son:
   $\forall A$ y $\exists A$

De la definición resulta que toda fórmula debe ser verdadera o falsa. Volveremos sobre ésto en la semántica de la lógica de predicados.

Ejemplo 2.2.1.1

Veamos una fórmula que exprese que dos números enteros x e y son primos entre sí (no tienen divisor común mayor que 1).
Se usa la función predicativa $E(u, v)$ que es V si y sólo si $u = v$ de argumentos enteros y la función común $M(r, s)$ que da como valor el producto de los enteros r y s. La fórmula es:

$$\forall k \forall k'(\neg \exists z(E(x, M(k, z)) \wedge E(y, M(k', z)) \wedge \neg E(z, 1)))$$

Ejercicio 2.2.1.1

Traducir la fórmula anterior al lenguaje corriente.

Ejercicio 2.2.1.2

Expresar en una fórmula "Tom tiene algún hermano al cual le agradan todos los automóviles"

2.2.2 Variables libres y ligadas

La idea de variable ligada es usual en las fórmulas matemáticas. Cuando se escribe:

$$\sum_{n=1}^{k} x^n/n$$

se ve que n recorre un conjunto de valores enteros $1, 2, ...k$ el cual queda determinado por la fórmula. Cuando se da a n un valor este debe ponerse en toda la fórmula (exponente y divisor). Se dice que tal variable está ligada. No podemos al evaluar la fórmula asignarle un valor cualquiera de su dominio. Por otra parte x y k pueden tomar valores cualesquiera asignados desde afuera siempre que estén en el campo de valores posibles de esas variables (x real y k entero).
En las expresiones de lógica de predicados $\forall$ y $\exists$ están seguidos de variables que si aparecen en la fórmula que las sigue se consideran ligadas a las del cuantificador. En este sentido son análogas a las del operador $\sum_{n=1}^{k}$.
En la fórmula del ejemplo 2.2.1.1 k, k', z son variables ligadas mientras que x, y son libres. Si les damos los valores $x = 100$ y $y = 75$ la fórmula resulta falsa, como se ve al poner $z = 25, k = 4, k' = 3$. Si le damos los valores $x = 100, y = 27$ la fórmula es verdadera.

Ejercicio 2.2.1.3

Expresar las siguientes proposiciones:

> "Existen personas a las cuales les disgusta algo".
> "Hay personas a las cuales les disgustan todos los helados".

Indicar, en cada caso cuales son variables libres y cuales ligadas. Usar las funciones predicativas $P(x)$ x es una persona; $H(y)$ y es un helado; $G(r, z)$ a r le gusta Z.
Notamos que, en el cálculo de predicados tal como ha sido definido no hay variables cuyos valores sean funciones (comunes o predicativas). Tal cálculo, como se dijo se llama de primer orden.
Se puede desarrollar un cálculo de segundo orden, en el cual hay variables cuyos valores son funciones. Su poder expresivo es mayor.
Por ejemplo para definir la igualdad se pone simplemente:
a=b por definición $\forall P(P(a) = P(b))$, es decir todo lo que se dice de a (P es cualquier predicado) puede decirse de b y viceversa. En el cálculo de primer orden la definición es más complicada. Pero se ve que el poder deductivo de la lógica de segundo orden es menor.
Para terminar con la sintaxis del cálculo de predicados hay que notar que el **alcance** de un cuantificador indica hasta qué aparición de la variable cuantificada debe aplicarse la cuantificación. Pueden presentarse algunos casos dudosos que se ven en los ejemplos siguientes.

$$\forall x M(x) \supset P(x)$$

La fórmula se divide por el conectivo $\supset$ así que la cuantificacón afecta $M(x)$. La x de $P(x)$ es una variable libre. Aunque no hay confusión conviene tener nombres diferentes para variables ligadas y libres. Esto se logra renombrando algunas variables.

$$\forall x[P(x, y) \vee \exists x Q(x, z)]$$

En este caso dentro del alcance de $\forall x$ (la fórmula entre [ ] se redefine la cuantificación mediante el cuantificador $\exists x$. Para evitar complicaciones en la interpretación conviene renombrar la variable de cuantificación de $\exists$. Esto se hace indispensable cuando se quiere, en las transformaciones que veremos más adelante, poner todos los cuantificadores al comienzo de la expresión.
En general en el procesamiento de una fórmula de izquierda a derecha se considera a los cuantificadores $\forall, \exists$ como de más precedencia que $\supset, \vee, \wedge$ pero con menos que el $\neg$.
Así en $\forall x \neg P(x) \vee R(y)$ el $R(y)$ debe procesarse cuando ya se ha tenido en cuenta todo el alcance del $\forall x$ hasta el $\vee$.

2.3 Semántica del cálculo de predicados

Como en el cálculo proposicional, en el de predicados deben darse criterios para dar valor de verdad a las fórmulas.

Para los operadores $\vee, \wedge, \supset, \equiv$, entre fórmulas y el $\neg$ aplicado a una fórmula, los criterios son como los del cálculo proposicional. Es necesario dar criterios para los cuantificadores y las funciones, comunes o predicativas.
Tenemos las reglas siguientes:

1. Las **proposiciones** se refieren a **objetos** de un cierto conjunto D que llamaremos **dominio de interpretación** que suponemos no vacío. Si fuera D vacío entonces $\forall x P(x)$ sería válida para cualquier P ya que $x \epsilon D \supset P(x)$ es siempre verdadero pues $x \epsilon D$ es falso para todo x.

2. Para interpretar las **funciones** definimos una función interpretativa I_c que a cada constante funcional f n-aria (n=0,1,...) le hace corresponder una función n-aria de $D^n \rightarrow D$ (si n=0 es una constante) que denotaremos por $I_C(f)$.

3. Para las constantes predicativas P n-arias les hacemos corresponder una función lógica $I_C(P)$ de D^N en el conjunto { V,F }.

4. A cada variable la interpretación le hace corresponder un objeto de D mediante la función interpretativa I_v. Es decir, en una interpretación debe darse valor a todas las variables.

Una interpretación I es pues una terna $I = (D, I_c, I_v)$ Con tal interpretación se pueden expresar proposiciones acerca del dominio D.

Ejemplo 2.3.1

. Sea D el conjunto de objetos en una habitación.
$D = \{silla\_1, silla\_2, mesa, televisor, piso, estante, sofa, pared\}$
y las funciones:

$s(x)$:	común unaria;	$I_c(s)$	sostiene a x
$M(x)$:	predicativa unaria;	$I_c(M)$	x es de madera
$T(x,y)$:	predicativa binaria;	$I_c(T)$	x toca y

La función s aplica D en D y se define así:

Para silla_1, silla_2, mesa, sofá, pared, vale: piso;
Para televisor vale mesa;
Para estante vale pared;
La función predicativa M :
Para silla_1,silla_2,mesa,estante vale V;
Para los otros vale F
La función T requiere, para ser definida una tabla de 64 entradas Para (silla_1,piso), (silla_2,piso), (mesa,piso), (sofa,piso), (pared,piso), (pared,estante), (pared,sofa), (mesa,televisor), y los pares simétricos vale V; para los demás pares vale F.

2.3.1 Reglas de interpretación

Para definir la verdad de una fórmula hay que dar reglas que permitan calcular los valores de las variables y las funciones. Las reglas son:

- Si x es una variable libre, $I(x) = I_v(x)$
- Si f es una constante funcional n-aria y $t_1, t_2, ..., t_n$ son términos (variables o formas funcionales) la interpretación de la fórmula f es el valor de la función correspondiente en la interpretación cuyos argumentos son las interpretaciones de los términos, es decir:
  $I(f(t_1, t_2, ..., t_n)) = I_c(f)(I_v(t_1), I_v(t_2), ..., I_v(t_n))$
  Así en el ejemplo 2.3.1, si $I_v(x) =$ mesa, $s(x)$ se interpreta por sostiene_a(mesa) que según su tabla de valores vale piso.
- Si P es una constante predicativa n-aria y $t_1, t_2, ..., t_n$ son términos la interpretación es: $I(P(t_1, t_2, ..., t_n)) = I_c(P)(I_v(t_1), I_v(t_2), ..., I_v(t_n))$
  En el ejemplo si la constante es T y es: $I_v(x) = silla\_1$ y $I_v(y) = mesa$, entonces: $T(x, S(y))$ se interpreta:
  Toca(silla_1, sostiene a(mesa)) = Toca(silla_1,piso) $\bar{\text{V}}$
- Si t_1, t_2 son términos $I(t_1 = t_2)$ es V si $I(t_1) = I(t_2)$. Si no es falso.
  Por ejemplo si: $I_v(x)$=mesa, $I_v(y)$ = televisor, entonces la interpretación de la igualdad:
  $x = S(y)$ es mesa=sostiene a (televisor), que es V dado que sostiene a (televisor) vale mesa. En cambio $x = y$ es falsa.
- Si A y B son fórmulas entonces los valores de
  $\neg A, A \vee B, A \wedge B, A \supset B, A \equiv B$
  se definen como en álgebra de proposiciones.
- Para interpretar los cuantificadores introducimos la notación:
  $I_{x/d}$ es una interpretación como la $I = (D, I_c, I_v)$ pero que asigna a la variable x la interpretación d siendo $d \epsilon D$. Entonces:
  - Si A es una fórmula y x la variable ligada se interpreta:
    $I(\forall A) = V$ si $I_{x/d}$ para **todo** $d \epsilon D$
    $= F$ en los demás casos.
  - Si A es una fórmula y x la variable ligada se interpreta:
    $I(\exists A) = V$ si $I_{x/d}$ para algún $d \epsilon D$
    $= F$ si no hay tal d.

  Así en el ejemplo 2.3.1 $\forall x M(x) = F$ como se ve al hacer $I_{x/sofa}$.
  Pero es: $\exists x M(x) = V$ como se ve al hacer $I_{x/mesa}$.

Como en el caso del cálculo proposicional, una interpretación I de la fórmula A tal que $I(A) = V$ se llama **modelo** de A

Una fórmula se dice:

- **válida** si es V para toda interpretación
- **consistente** si es V para alguna interpretación
- **inconsistente** si es F para toda interpretación
- **contingente** si es consistente pero no válida.

El problema de decidir la contingencia de una fórmula es aquí más complicado que en el cálculo proposicional ya que D puede ser muy grande o ser infinito (por ejemplo si incluye el conjunto de los números naturales). No hay un algoritmo general para demostrar que una fórmula de cálculo de predicados es válida, consistente o contingente.

Ejercicio 2.3.1.1

Basado en el ejemplo 2.3.1 interpretar y expresar en lenguaje ordinario las fórmulas siguientes:

1. $\exists x M(x)$
2. $\forall x(M(x) \supset (piso = S(x)))$
3. $\forall x \forall y(T(x, y) = T(y, x))$

Ejercicio 2.3.1.2

Expresar fórmulas cuya interpretación corresponda con:

1. "Si un objeto sostiene a otro lo toca"
2. "Algunos objetos de madera se tocan"

¿Como demostraría estas afirmaciones? ¿Cómo se demostraría que si un objeto sostiene a otro, es tocado por éste?

2.3.2 Repertorio de fórmulas válidas

Las siguientes fórmulas del cálculo de predicados son válidas y se pueden usar para transformar expresiones. En ellas A y B son fórmulas que no contienen cuantificadores.

$(\forall x A) \wedge (\forall x B) \equiv \forall x(A \wedge B)$
$(\forall x A) \vee (\forall x B) \supset \forall x(A \vee B)$
$\forall x(A \supset B) \supset (\forall A \supset \exists A)$
$\forall x(A \equiv B) \supset (\forall A \equiv \exists A)$
$\exists x(A \supset B) \equiv (\exists A \supset \exists B)$
$\exists x(A \vee B) \equiv (\exists A \vee \exists B)$
$\exists x(A \wedge B) \supset (\exists A \wedge \exists B)$
$\forall x \neg A \equiv \neg \exists x A$
$\exists x \exists y A \equiv \exists x \exists y A$
$\exists x \forall y A \supset \forall y \exists x A$

En los casos en que se ha puesto $\supset$ es porque la fórmula no es válida con $\equiv$.

El cálculo de predicados es una extensión del cálculo proposicional. Todo esquema de deducción que sea válido en el cálculo proposicional lo es también en el cálculo de predicados.

Ejercicio 2.3.2.1

Demostrar la validez de las fórmulas anteriores.

2.4 Formas Prenex

Como en el cálculo proposicional, en el de predicados ciertas formas de las fórmulas facilitan la determinación de la consistencia y validez de conjuntos de fórmulas. Como allí consistencia o validez de un conjunto de fórmulas significa la consistencia o validez de la conjunción de esas fórmulas.

Llamamos **forma prenex** a una **matriz** (fórmula sin cuantificadores) precedida de un **prefijo** que es una sucesión finita de cuantificaciones.

Ejemplo 2.4.1

$\exists x \exists y \forall z((x \vee z) \supset (y \vee t))$

En general, al cambiar el orden de los cuantificadores puede cambiar el valor de verdad de la fórmula en una cierta interpretación. Hemos visto en 2.3.2 que $\exists x \forall y A$ no es equivalente a $\forall y \exists x A$. Es claro que si una fórmula A no contiene x entonces $A, \exists x A, \forall x A$ son equivalentes.
Se puede ver el siguiente teorema que se demuestra por construcción:

Teorema 2.4.1 Para cada fórmula existe una fórmula prenex equivalente.

El algoritmo de conversión es el siguiente:

1. Renombrar las variables para que todas las ligadas tengan diferente nombre.

2. Eliminar los símbolos $\supset$ y $\equiv$ como en el caso de la reducción a forma normal conjuntiva (ver Teorema 1.7.1).

3. Renombrar las variables de manera que no queden variables libres y ligadas con igual nombre.

4. Eliminar cuantificadores cuya variable no ocurra en su alcance.

5. Pasar las negaciones inmediatamente delante de los átomos de la expresión. Para ello se usan las siguientes reglas de sustitución:
   $\neg\forall xA \to \exists x\neg A$
   $\neg\exists xA \to \forall x\neg A$
   $\neg(A \wedge B) \to \neg A \vee \neg B$
   $\neg\neg A \to A$

6. Transferir las cuantificaciones al comienzo de la fórmula. para ello se usan las siguientes reglas de sustitución:
   $\forall xA \wedge \forall xB \to \forall x(A \wedge B)$
   $\forall xA \wedge B \to \forall x(A \wedge B)$
   $A \wedge \forall xB \to \forall x(A \wedge B)$
   $\exists xA \wedge B \to \exists x(A \wedge B)$
   $A \wedge \exists xB \to \exists x(A \wedge B)$
   y las análogas para la disyunción:
   $\forall xA \vee \forall xB \to \forall x(A \vee B)$
   $\forall xA \vee B \to \forall x(A \vee B)$
   $A \vee \forall xB \to \forall x(A \vee B)$
   $\exists xA \vee B \to \exists x(A \vee B)$
   $A \vee \exists xB \to \exists x(A \vee B)$

7. Pueden usarse las propiedades algebraicas de $\vee$ y $\wedge$ (asociatividad, conmutatividad e idempotencia) para simplificar las fórmulas.

Con esto queda separado el prefijo de la matriz. ésta es como una fórmula de cálculo proposicional. Puede ponerse como una forma normal conjuntiva de **cláusulas** en las cuales los **literales** son **fórmulas o negaciones de átomos**. Un **conjunto** de cláusulas cada una de las cuales se reduce a la forma prenex puede ponerse como una conjunción de todas las cláusulas y se puede llevar a una sola forma prenex.

Ejercicio 2.4.1

Reducir a la forma prenex:

$$\forall x[P(x) \wedge \neg\forall y\exists x(\neg Q(x,y) \supset \forall z S(a,y,x))]$$

2.5 Formas de Skolem

Una vez llegados a la forma prenex con su matriz transformada en una forma normal conjuntiva A, se puede hacer todavía otra transformación que facilita la determinación de la consistencia. Tal transformación elimina los cuantificadores existenciales. Llamamos S_A a tal fórmula. Veremos que no es equivalente a A pero será consistente si y sólo si lo es A. Claro que puede haber interpretaciones que hagan S_A verdadera y A falsa. Pero si A es consistente siempre habrá una interpretación que haga verdadera S_A , y si es A inconsistente no habrá ninguna que haga verdadera S_A. Es decir S_A permite decidir sobre la consistencia de A.
Esto es lo que en realidad interesa pues si se desea saber si la fórmula C es consecuencia lógica de las hipótesis H_1, H_2 hemos visto que:
$\{H_1, H_2\} \models C \Leftrightarrow (H_1 \wedge H_2 \wedge \neg C) = F$, es decir:
$A = H_1 \wedge H_2 \wedge \neg C$ es inconsistente.
La prueba de inconsistencia se aplica a S_A y permite decidir sobre la consecuencia lógica.

Supongamos que A tiene la forma prenex y no tiene variables libres. Si las tiene y son $x_1, x_2, .., x_n$ se hace la cláusula existencial $\exists x_1 \exists x_2 ... \exists x_n A$ que es consistente si y sólo si lo es A. Con esto se eliminan las variables libres.

Ejercicio 2.5.1

Justificar esta observación.

Ahora se procede a eliminar los cuantificadores existenciales. Para ello en la matriz cada variable que esté afectada por un cuantificador existencial se sustituye por una forma funcional cuyos argumentos son las variables que preceden a esta en el prefijo y son afectadas por un cuantificador universal. Luego se quitan todos los cuantificadores existenciales. La forma prenex con matiz normal conjuntiva y sin cuantificadores existenciales es la S_A mencionada.

Ejemplo 2.5.1

Sea la forma prenex:
$\forall x \forall y \exists z M(z, x, y)$
donde x,y,z son enteros y M significa "z excede al producto de x por y". Se puede dar una forma sin el cuantificador $\exists$ introduciendo una función $f(x, y)$ tal que sus valores sean siempre mayores que el producto $x \times y$. Por ejemplo $x \times y + 3$. Tendríamos:
$\forall x \forall y M(f(x, y), x, y)$
M significa "$f(x, y)$ excede al producto de x por y". Si la primera fórmula es consistente (tiene valores de x, y, z que la hacen verdadera, entonces es posible construir tal función $f(x, y)$. Si es inconsistente no es posible. Cualquier función que se ponga hace esta última forma inconsistente.

Ejemplo 2.5.2

Sea la forma prenex de la matriz M:
$A: \quad \exists u \forall x \exists v \forall y \forall z \exists w M(u, v, w, x, y, z)$
viendo cuantos cuantificadores $\forall$ preceden a cada $\exists$ de las u, v, w resulta la forma de Skolem:
$S_A: \quad \forall x \forall y \forall z M(a, f(x), g(x, y, z), x, y, z)$
Para que la relación entre A y S_A sea la deseada (S_A es consistente si y sólo si lo es A) la clave es la forma de construir las funciones $a, f(x), g(x, y, z)$. La $g(x, y, z)$ que sustituye a w por ejemplo fue construida de modo que si para una cierta interpretación es $x = x_1, y = y_1, z = z_1, w = w_1$, es $M = V$, entonces interpretamos $g(x_1, y_1, z_1)$ de modo que dé el valor w_1. De esta manera w sustituida en la fórmula original por una función que calcula un valor aceptable en tal interpretación de M. Como los valores a calcular se asignan en el orden indicado por los cuantificadores resulta que a no depende de ningún valor que se dé a x, y, z. Es pues una constante. La f depende sólo de x; g depende de x, y, z. Entonces tal interpretación que hace verdadera a A hace verdadera a S_A. Es decir si A es consistente, lo es S_A. Por otra parte si A es inconsistente no hay interpretación posible que la haga verdadera así que S_A que resulta de poner en A ciertos valores para u, v, w no puede hacer S_A verdadera. Así que S_A es inconsistente.
Se llama **forma clausal** a una forma de Skolem cuya matriz es una forma normal conjuntiva.
Resulta pues que a toda forma predicativa le podemos asociar una forma clausal que es consistente si y sólo si lo es la forma original.

2.6 Interpretación de Herbrand

No hay un algoritmo general que permita decidir si una forma clausal es o no consistente. Esta dificultad en el cálculo de predicados se debe a la infinitud posible de las interpretaciones.
Una idea simplificativa es buscar un subconjunto de esa infinitud donde sea posible decidir sobre la consistencia y que sea tal que la forma clausal sea inconsistente en ese dominio si y sólo si es inconsistente en general. Esto reduce el problema de explorar todos los dominios posibles. Un dominio con tales características es el llamado **dominio de Herbrand** que se introduce a continuación.

Sea G un conjunto de formas clausales. Definimos:

1. Dominio de Herbrand es el dominio mínimo H_G tal que:
   - Para toda constante individual $a \epsilon G$ el dominio tiene una constante individual $A \epsilon H_G$.
   - Si G no tiene ninguna constante individual introducimos una cualquiera $C \epsilon H_G$.
   - Para toda constante funcional (no predicativa) n-aria $e \epsilon G$ introduciremos en H_G la constante funcional E y además los términos $E(h_1, h_2, .., h_n)$ donde los

h_i son elementos de H_G.

Nótese que si G no contiene constantes funcionales es simplemente $H_G = \{C\}$
Si G contiene alguna constante funcional entonces H_G es infinito.

Ejemplo 2.6.1

Sea $G = \{R(x) \wedge P(e(x))\}$. El H_G asociado resulta $H_G = \{C, E(C), E(E(C)), ...\}$

2. Interpretación de Herbrand de G es una $I(G)$ donde:

   - El dominio de interpretación es H_G.
   - La interpretación de la constante individual $a \epsilon G$ es A, es decir $I(a) = A$.
   - La interpretación de la forma funcional $E(x_1, x_2, .., x_n)$ es una función de $H_G^n \rightarrow H_G$ que hace corresponder a $(h_1, h_2, .., h_n) \epsilon H_G$ el término $E(h_1, h_2, .., h_n)$ siendo E la constante funcional de H_G correspondiente a $e \epsilon G$.

Teorema 2.6.1 Sea G un conjunto de formas clausales. Tal conjunto es inconsistente si y sólo si toda interpretación de Herbrand es falsa.

Es obvio que la condición es necesaria. Si alguna interpretación fuera verdadera G no sería inconsistente.
Para ver que es suficiente hay que ver que para cualquier $I(D, I_c, I_v)$ que haga $I(G) = V$ se halla una interpretación de Herbrand I^H tal que $I^H = V$ de modo que si no hay tal I^H verdadera es porque no hay ninguna $I(G)$ verdadera.
Para ver esto vamos a construir la I^H en tres etapas:

1. A cada término $T \epsilon H_G$ asociamos un $d_T \epsilon D$ de la interpretación I, en la siguiente forma:

   - Si la constante es individual $a \epsilon G$ definimos: $d_A = I_c(A)$
   - Si en G no hay ninguna constante individual, por lo cual en H_G hemos introducido una constante C entonces el correspondiente $d_C \epsilon D$ es un elemento cualquiera de D.
   - Para toda constante funcional $e \epsilon G$ n-aria (n$>$ 0) y para cualquier n-tupla $h_1, h_2, .., h_n$ de elementos de H_G definimos el elemento de D: $d_{E(h_1,h_2,..,h_n)} = I_c(e)(d_{h_1}, d_{h_2}, ..d_{h_n})$
     donde $E \epsilon H_G$ es el correspondiente a $e \epsilon G$.

2. Se define I_c^H de la siguiente forma:
   $I_c^H(a) = A$
   $I_c^H(e) =$ la E antes definida.

3. Se introducen interpretaciones que hace I_c^H para las constantes predicativas. Para una constante predicativa P m-aria la función $I_c^H(P)$ se define por:

$$I_c^H(P)(h_1, h_2, .., h_m) = I_c(P)(d_{h_1}, d_{h_2}, ..d_{h_n})$$

para toda m-tupla $h_1, h_2, .., h_m$ de H
Es decir se hace $I_c^H(P)$ para m elementos cualesquiera de H_G verdadero o falso según sea el $I_c(P)$ de los correspondientes elementos de D.

4. Queda ahora por ver que por cada forma clausal $g \epsilon G$ tal que $I(g) = v$ es $I^H(g) = V$. Sea

$$g = \forall x_1 \forall x_2 ... \forall x_n M(x_1, x_2, .., x_n)$$

donde $M(x_1, x_2, .., x_n)$ considerada sola es una matriz de las variables $x_1, x_2, .., x_n$. Por la forma en que se definió la $I^H(g)$ se ve que

$$I^H(g) = V \text{ si y sólo si } I^H(M)(h_1, h_2, .., h_n)$$

es V, donde $h_1, h_2, .., h_n$ son términos cualesquiera de H_G. Pero el valor de verdad de esta última ha sido definido como el de $I(M)(d_{h_1}, d_{h_2}, .., d_{h_n})$ con los d_{h_i}) correspondientes. Pero como $I(M) = V$ pues supusimos $I(g) = V$ entonces $I(M) = V$ para cualquier n-tupla de D, en particular: $d_{h_1}, d_{h_2}, .., d_{h_n}$. Así que $I^h(g) = V$.

Si tenemos una forma predicativa o una cláusula en el conjunto G de cláusulas, se denomina **forma base** o **cláusula base**, la que se obtiene de esa cláusula sustituyendo en ellas las variables por términos de H_G. Se dice que tales formas o cláusulas base son **instancias base** de las formas o cláusulas originales.
Una interpretación de Herbrand de G queda determinada por los valores de verdad de las instancias base de las cláusulas de G.
Si al sustituir en las cláusulas de G los términos de H_G las instancias base que resultan hacen G verdadero entonces G es consistente.
Si ninguna de las instancias base posibles hace G verdadera, entonces es G inconsistente y por el teorema visto no hace falta buscar otra interpretación, ninguna será verdadera.
Como H_G es usualmente infinito la demostración de inconsistencia debe hacerse indirectamente.
Lo importante es:

- que las formas base se manejan como proposiciones;
- que basta investigar la inconsistencia sobre una sola interpretación, cuyo dominio es finito numerable.

Esto establece una conexión entre el cálculo de predicados y el proposicional. En particular permite demostrar el teorema de compacidad:

Teorema 2.6.2 Un conjunto S de fórmulas del cálculo de predicados es consistente si y sólo si todo subconjunto finito es consistente.

La condición es necesaria pues si hubiera un subconjunto de S inconsistente, lo sería S. Sea el S un conjunto de cláusulas (ver 2.4.1). Supongamos que todo subconjunto finito de S es consistente. Sea S' el conjunto de todas las instancias base de las cláusulas de S (cada cláusula puede tener un número numerable de instancias). S sería consistente si y sólo si S' lo fuera (ver 2.6.1). Sea $A' \subset S'$ finito. Sea $A \subset S$ el conjunto de cláusulas con una instancia en A'. Por ser A' finito, A es finito y, por lo tanto (hipótesis) consistente. Por 2.6.1 el conjunto de sus instancias base es consistente. Tal conjunto de instancias base incluye A'. Luego A', que es un conjunto finito cualquiera de instancias de S' es consistente. Pero cada instancia es una fórmula proposicional. Por 1.12.2 S' es consistente. Entonces por 2.6.1 lo es S.

Teorema 2.6.3 Todo conjunto numerable consistente de fórmulas admite un modelo numerable.

Pues el dominio de su interpretación de Herbrand es numerable.
Este teorema debido a Löwenheim y Skolem es notable ya que pueden hacerse teorías consistentes, basadas en el cálculo de predicados, donde el dominio de interpretación es no numerable (teorías de los números reales por ejemplo). Sin embargo tales teorías admiten también una interpretación numerable.

2.7 Ejemplos de interpretaciones de Herbrand

Ejemplo 2.6.1.1

Tomemos como ejemplo el silogismo de la forma:

$$\begin{array}{ll} H_1 & \forall x(P(x) \supset Q(x)) \\ H_2 & \forall x(Q(x) \supset R(x)) \\ C & \forall x(P(x) \supset R(x)) \end{array}$$

Para probar que tal razonamiento es válido hay que probar que:
$h_1 \wedge h_2 \wedge \neg C$ es inconsistente. Lo cual equivale a la inconsistencia de:

$$\forall x((P(x) \supset Q(x)) \wedge (Q(x) \supset R(x))) \wedge \exists y(P(y) \wedge \not R(y)).$$

que en forma prenex es:

$$\exists y \forall x((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x))) \wedge P(y) \wedge \not R(y)).$$

cuya forma clausal de Skolem es:

$$\forall x((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x))) \wedge P(c) \wedge \not R(c)).$$

El dominio de Herbrand correspondiente, por no haber sino la constante c cuya representación en H es C se reduce a $\{C\}$. Hay una sola instancia base que asigna argumento

constante a cada forma predicativa $P(C), Q(C), R(C)$. Las interpretaciones de Herbrand son ocho según los valores V, F que puedan tomar estos predicados. De una inspección de la matriz (o de la tabla de verdad) resulta que la matriz de la forma es siempre falsa. Es decir no hay interpretación de Herbrand verdadera. Por lo tanto el conjunto de formas clausales (reducida a una sola) es inconsistente, con lo cual el esquema silogístico resulta válido.

Ejemplo 2.6.1.2

Sea el esquema de recurrencia:

B	$P(a)$	afirmación básica
I	$\forall x(P(x) \supset P(f(x)))$	inducción
G	$\forall x P(x)$	generalización

Un caso particular de este razonamiento es aquel en el cual el dominio de interpretación son los números naturales y f es la función sucesor.Tal esquema es el principio de inducción completa. La validez de $B \wedge I \wedge G$ equivale a la inconsistencia de $B \wedge I \wedge \neg G$ es decir de:

$$P(a) \wedge \forall x(P(x) \subset P(f(x))) \wedge \exists y \neg P(y)$$

cuya matriz es:

$$P(a) \wedge (\neg P(x) \vee P(f(x)) \wedge P(b)$$

pues al pasar a la forma clausal podemos pasar $\exists y \neg P(y)$ al comienzo y la función de elección al no haber cuantificadores $\forall$ precedentes se reduce a una constante b.
El dominio de Herbrand resulta:

$$h = \{a, b, f(a), f(b), f(f(a)), f(f(b)), ..., f^n(a), f^n(b), ...\}$$

donde f^n denota n aplicaciones de la función f a a.
Hay infinitas interpretaciones de Herbrand que resultan de sustituir en la matriz las variables por los elementos de H_G, es decir, las instancias base. Para hallar todas las interpretaciones hay que poner V o F de todas las formas posibles a los elementos del conjunto:

$$S = \{P(a), P(b), P(f(a)), P(f(b)), ..., P(f^n(a)), P(f^n(b)), ...\}$$

y ver si todas son inconsistentes. Pero en este caso se puede ver que hay una consistente. Sea la interpretación:
$P(f^n(a)) = V$ y $P(f^n(b)) = F$ para todo n.
Sustituyendo en la matriz:

$$P(f^{n+1}(a)) \wedge (\neg P(f^n(a)) \vee P(f^{n+1}(b))) \wedge \neg P(f^n(b))$$

y se ve que ésta es verdadera por lo menos en esta interpretación. Es decir que es consistente en ese dominio y por el teorema 2.6.1 lo sería en cualquier otro. El esquema de recurrencia no es pues válido en general.

En la fundamentación de la teoría de los números enteros el principio de inducción se acepta como postulado o se lo demuestra a partir de un postulado equivalente.
Hay tres dificultades para ver la consistencia de una forma en el Cálculo de Predicados.

1. Hay infinitos dominios de interpretación y la demostración de inconsistencia en uno no prueba que deba haberla en otro.

2. Si el dominio elegido es infinito hay una infinidad de instancias de la fórmula al sustituir en las formas funcionales distintos valores de los argumentos.

3. Si el dominio es infinito podemos aún definir las funciones predicativas de forma que sean verdaderas o falsas para unos valores determinados de los argumentos.

El método de Herbrand elimina la primera dificultad. Basta investigar el dominio H_G. Si H_G es infinito y A es una formula, las instancias base por ser la fórmula finita, son numerables. Sean $A_1, A_2, ..., A_n, ...$ Si tomamos i de ellas tenemos el conjunto de instancias $S_i = \{A_1, A_2, ..., A_i\}$. Como son en número finito se puede decidir la consistencia. Si son consistentes incrementamos i considerando $i+1$ instancias. La fórmula A es inconsistente si y sólo si para algún S_i lo es. Se tiene que descubrir la inconsistencia, si la hay, en un número finito de pasos, pero el procedimiento puede ser muy ineficiente.

2.8 Algoritmo de Quine, Davies y Putnam

El algoritmo visto en 1.8 se puede aplicar a las instancias base de las formas clausales para decidir sobre su consistencia.

Ejemplo 2.7.1

Veamos el mismo ejemplo 2.6.1, para el cual hay que investigar la consistencia del conjunto de instancias base:

$S = \{\neg P(c) \vee Q(c), \neg Q(c) \vee R(c), P(c), R(c)\}$.
$S_1 = S \setminus (S_{\neg P(C)} \vee S_{P(c)}) = \{Q(c), \neg Q(c) \vee R(c), P(c), R(c)\}$
$S_2 = S'_{\neg P(C)} \vee S'_{P(c)} \vee S'' = \{Q(C), \neg Q(C)\}$

Se ve que este último es inconsistente, luego lo es S y el conjunto original, lo cual prueba la consistencia de la forma silogística. Las extensiones a dominios infinitos no son muy útiles.
También puede usarse el algoritmo de resolución.

Ejemplo 2.7.2

Para el mismo ejemplo 2.6.1 tenemos las siguientes cláusulas en el conjunto base:

1. $\neg P(c) \vee Q(c)$

2. $\neg Q(c) \vee R(c)$

3. $P(c)$
4. $R(c)$
5. $Q(C)$ de 1 y 3
6. $\neg R(C)$ de 2 y 5
7. F de 6 y 4

lo cual prueba la inconsistencia del conjunto base.

Ejemplo 2.7.3

Para el caso 2.6.2 con dominio infinito se tiene:

$$S = \{P(a), P(b), P(f(a)), P(f(b)), ..., P(f^n(a)), P(f^n(b)), ...\}$$

La matriz es:

$$P(a) \wedge (\neg P(x) \vee P(f(x)) \wedge P(b)$$

Tenemos que sustituir en la matriz todos los casos:

$$x = a, x = b, f(a), f(b)$$

La sustitución procede así:

1	$P(a)$	original
2	$\neg P(x) \vee P(f(x))$	original
3	$P(b)$	original
4	$\neg P(a) \vee P(f(a))$	instancia de 2 con a
5	$P(f(a))$	resolvente de 1 y 4
6	$\neg P(f(a)) \vee P(f_2(a))$	instancia de 2 con f(a)
7	$P(f_2(a))$	resolvente de 5 y 6
8	$\neg P(f_2(a) \vee P(f_3(a))$	instancia de 4 con f(a)
9	$P(f_3(a))$	resolvente de 7 y 8

Se ve que el único proceso posible de reducción se repite con instancias que sólo difieren en el grado de $f_n(a)$. El proceso sigue infinitamente pero nunca obtenemos F, lo cual prueba que el conjunto es consistente.

2.9 Sustituciones en términos

Una sustitución es una función del conjunto de las variables en el de los términos. Por ejemplo a una variable x se la sustituye por otra y o bien por una forma funcional $f(x, y)$ que es también un término (ver 2.2).
Una sustitución se denota como un conjunto s de pares: (variable, término) que indica por cual término hay que sustituir la variable. La sustitución se aplica a un término y produce otro término.

Ejemplo 2.8.1

Sea la sustitución:

$s = \{(x, f(x,z)), (y,z), (u, g(h,v\}$ aplicada al término:
$r = f(f(x,z), g(y,u))$ resulta el nuevo término.
$s(r) = f(f(f(x,z),z), g(z, g(h,v)))$

Podemos considerar que, para los términos no sustituidos, cada uno se sustituye por sí mismo. Sólo un número finito de elementos de esta aplicación va de un término a otro diferente. La sustitución idéntica deja a cada término como está.
La aplicación sucesiva de dos sustituciones es una sustitución. Así si aplicamos s_1 y luego s_2 esto equivale a una sustitución S que es la **composición** de ambas. Se indica: $s = s_2 \circ s_1$.

Ejemplo 2.8.2

Consideremos la del ejemplo 2.8.1 como s_1 y definamos
$s_2 = \{(z, r(x,z)), (x, k(x))\}$. Tenemos:

$$s = s_2 \circ s_1(r) = f(f(k(x), r(x,z))g(r(x,z)), g(h,v)))$$

Ejercicio 2.8.1

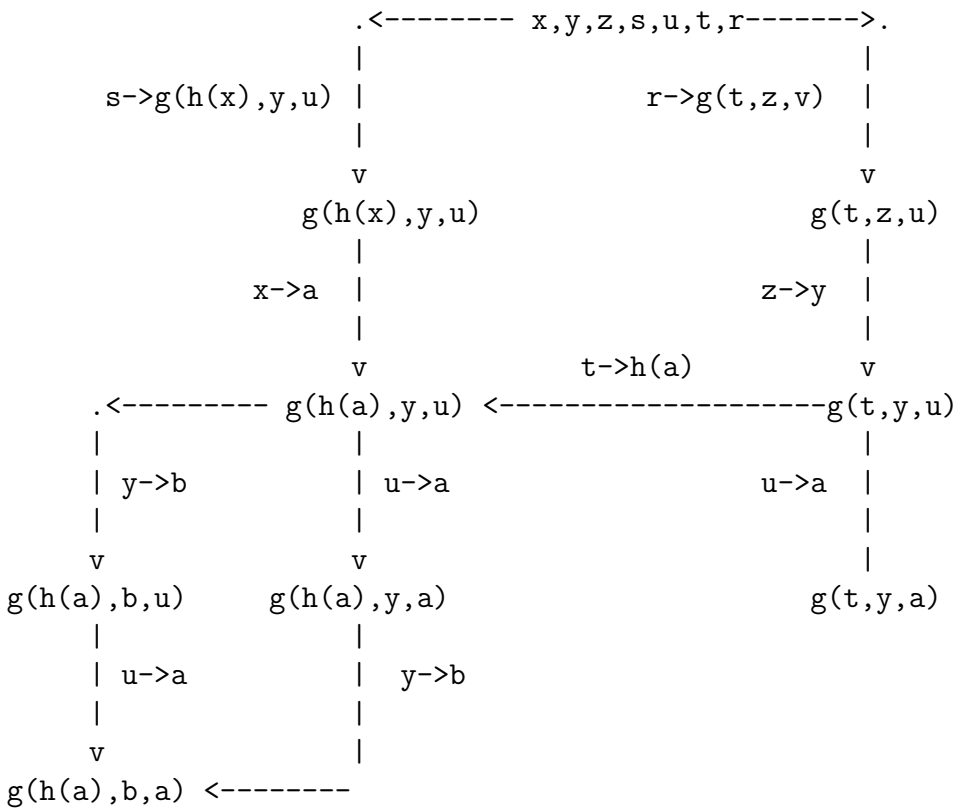
Escribir el conjunto de pares que forman la $s = s_2 \circ s_1$ del ejemplo 2.8.2.

Ejercicio 2.8.2

Ver que las sustituciones con la operación de composición $\circ$ vista es asociativa; existe la identidad y no es conmutativa.
Se define la **unión** de dos sustituciones $s_1 \cup s_2$ por la unión de los conjuntos de pares. Debe cumplirse la condición de que para cualquier variable x al menos uno de los dos s_1 o s_2 es igual a x. Si no ocurre así, en la expresión de la unión aparecerían dos versiones de la sustitución de x que serían incompatibles.
Si al aplicar s a t_1 obtenemos $t_2 = s(t_1)$ el t_2 se denomina **instancia** de t_1.
Se denota $var(t)$ al conjunto de variables de un término.
Una instanciación (aplicación de una sustitución) puede quitar variables a dicho conjunto, sustituyéndolas por términos. Cuando se tiene $var(t) = \emptyset$ se dice que el término está **totalmente instanciado**. Para indicar que t_2 es una instancia de t_1 por una sustitución se escribe $t_2 \prec t_1$. La relación $\prec$ reflexiva y transitiva pero no simétrica. No es tampoco antisimétrica pues en general no es cierto que si es $t_2 \prec t_1$ y $t_1 \prec t_2$ ello no implica $t_1 = t_2$. Pero si ambas relaciones se cumplen, ello significa que las sustituciones que llevan de t_1 a t_2 y de t_2 a t_1 sólo cambian el nombre de las variables. Los términos que sólo difieren en el nombre de las variables se pueden poner en una clase de equivalencia. En lo que siguen nos referiremos a estas clases o a un representante. La relación $\prec$ define un reticulado cuyo máximo es la clase de las variables y cuyos mínimos son los términos totalmente instanciados.

El **extremo superior** de un conjunto de términos siempre puede calcularse subiendo en el árbol hasta el máximo y llegando a un antecesor común. El extremo inferior no siempre existe.

Ejemplo 2.8.3 Reticulado de términos



El extremo superior del conjunto: `{g(h(a),b,u),g(t,y,a), g(h(a,b,a)}` es `g(t,y,u)`. El extremo inferior del conjunto `{g(h(a),b,u), g(t,y,u)}` es `g(h(a),b,u)`. El extremo inferior del conjunto `{g(h(x),y,u), g(t,z,u)}` es `g(h(a),y,u)`. El conjunto `{g(t,y,a),g(h(x),y,u}` no tiene extremo inferior.

2.10 Unificación

El método de resolución base puede resultar ineficiente debido a que las instancias base son infinitas.
Es posible usar las cláusulas mismas sin una instanciación explícita. Esto se puede hacer mediante el procedimiento siguiente llamado **unificación**. Sean las cláusulas de S del cual se quiere averiguar la consistencia, C_1 y C_2:
$C_1 = \{..., l_1, ...\}$ y $C_2 = \{..., \neg l_2, ...\}$
l_1 y l_2 son literales.

Suponemos además que C_1 y C_2 no tienen variables con igual nombre. ésto puede lograrse renombrando adecuadamente. Si l_1 y l_2 al ser interpretadas en H producen dos instancias iguales se dice que C_1 y C_2 son **unificables**. Con esta instanciación de C_1 y C_2 se producen:
$C_1' = \{..., l', ...\}$ y $C_2 = \{..., \neg l', ...\}$
Estas cláusulas admiten como resolvente la cláusula:
$R' = C_1' \setminus \{l'\} \vee C_2' \setminus \{\neg l'\}$
R' es una consecuencia lógica de C_1' y C_2' y estas, por ser instanciaciones de C_1 y C_2 son consecuencias lógicas de C_1 y C_2.
La instanciación puede hacerse de muchas formas siguiendo el reticulado de instanciaciones. Se trata de hacerla con el número menor de sustituciones logrando una resolvente R cuyas instanciaciones posteriores nos puedan dar todas las R' logradas con más sustituciones. Tal cláusula, cuando se la puede hallar, se llama **cláusula resolvente general** de los literales l_1 y l_2. Recordando el reticulado de sustituciones vemos que la cláusula lograda mediante sustituciones que hacen iguales las transformadas de l_1 y l_2 es $l' \prec l_i$ siendo l_i el extremo inferior de l_1 y l_2. Las sustituciones que llevan a las que tienen menos sustituciones son entonces s_1 y s_2 tales que:
$s_1[l_1] = l_i$ y $s_2[l_2] = l_i$. Por la condición de C_1 y C_2 de no tener variables de igual nombre, resulta que l_1 y l_2 no tienen variables comunes, así que s_1 y s_2 operan siempre sobre variables diferentes y $s_1 \cup s_2 = s_i$ está definida sin ambigüedad. Se tiene pues que s_i es el unificador más general. La cláusula resolvente general es:
$R = (s_i(C_1) \setminus \{l_i) \vee (s_i(C_2) \setminus \{l_i)$
Como las cláusulas l_1, l_2 y l_i son átomos, son formas predicativas.
Para que sean unificables deben tener la misma constante predicativa y aplicarse a términos (variables o formas funcionales) unificables. Los átomos se unifican unificando los términos. Si uno de los términos es una variable, la unificación es inmediata ($P(x, u)$ y $P(y, u)$ se unifican sustituyendo x por y. Si ambos son formas funcionales, sólo se pueden unificar si la constante funcional es igual y si se aplica a términos unificables. Así se pueden unificar $P(x, f(z))$ y $P(x, f(u))$ sustituyendo u por z.

2.11 Algoritmo de resolución en lógica de predicados

El algoritmo de resolución (1.9) toma ahora la forma:

Repetir mientras F $\not\in S$ y se pueda seleccionar:
Seleccionar l_1, l_2, C_1, C_2 tales que:
C_1, C_2 son clausulas o factores de clausulas;
$l_1 \varepsilon C_1, \neg l_2 \varepsilon C_2$ y l_1, l_2 son unificables;
Calcular la resolvente $R = (s_i(C_1) \setminus \{l_i) \vee (s_i(C_2) \setminus \{l_i)$;
Reemplazar S por $S \cup R$

2.12 Gramáticas y consistencia de conjuntos de cláusulas de Horn

Una gramática puede definirse por un conjunto de reglas llamadas **producciones** que permiten generar y reconocer estructuras gramaticales. Las producciones definen las diferentes estructuras gramaticales, las más generales (como oración, predicado, complemento) por medio de las más particulares hasta llegar a las más concretas o constantes (verbo, nombre, artículo).

Ejemplo 2.12.1

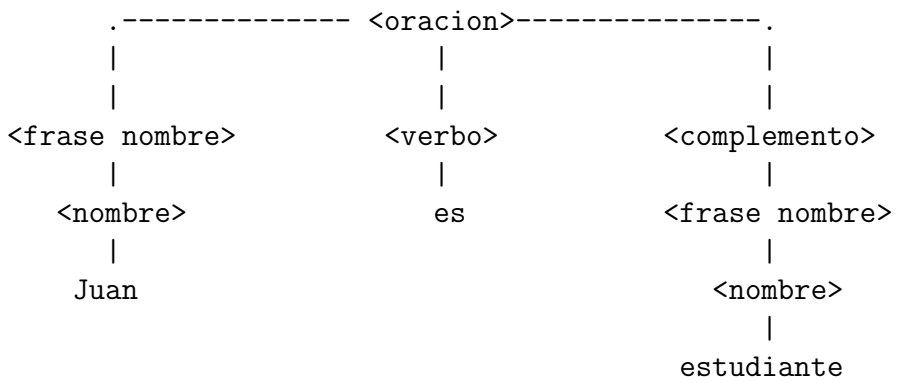
Una gramática muy simple sería la dada por las producciones siguientes:

<oración>:-	<frase nombre> <verbo> <complemento>
<frase nombre>: -	<nombre>
	<artículo> <nombre>
	<artículo> <nombre> <adjetivo>
<complemento>:-	<adverbio>
	<adverbio> <complemento>
	<frase nombre>
<verbo>:-	es
	lee
	busca
<artículo>:-	el
	un
<adjetivo>:-	difícil
	voluminoso
<adverbio>:-	rápidamente
<preposición>:-	de
<nombre>:-	libro
	madera
	Juan
	estudiante

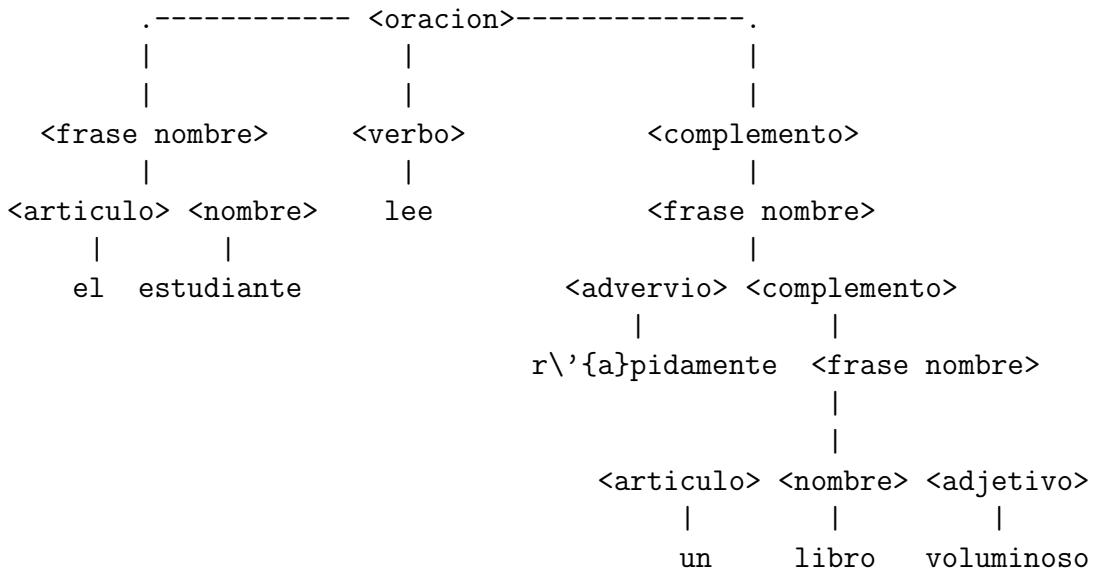
Ejercicio 2.12.1

Expresar la gramática del ejemplo mediante un árbol cuya raíz es <oración> y cuyas hojas son las constantes o átomos.
Una oración bien construida (aunque no significativa) se forma sustituyendo en <oración> cada elemento por su significado dado por las producciones hasta llegar a las constantes (verbos, nombres, etc.).

Ejemplos:



Véase que se puede obtener "Juan lee madera" que es sintácticamente correcta pero no lo es semánticamente.



Recíprocamente dada una hilera de palabras básicas, las producciones permiten saber, mediante un procedimiento o algoritmo, si la hilera es una oración bien formada.

Ejemplo

Sea la oración:
el estudiante lee rápidamente el libro.
El procedimiento descendente busca una <oración>. Por tanto busca una <frase nombre>. Por tanto busca un <nombre> o un <artículo>. Halla "el" que es <artículo> por lo cual

busca un <nombre>. Halla "estudiante" que es nombre. Si lo que sigue no es adjetivo, completó la búsqueda de frase nombre. Si en la búsqueda de una estructura agota todas las alternativas y no la encuentra, entonces tenemos un error. La hilera no es una oración. Prosiguiendo, según la producción de <oración> debe buscar <verbo>. Halla "lee" que es verbo. Pasa a buscar <complemento>. Por tanto busca <adverbio> o <frase nombre>. Como halla un <adverbio> ("rápidamente") busca <frase nombre>, así que busca <artículo> o <nombre>. Halla "el" que es <artículo>. Busca <nombre>. Halla "libro". El punto final detiene la búsqueda. Como todas las búsquedas han sido exitosas la hilera es una <oración>.
Véase que las producciones se pueden expresar en forma de un árbol y el algoritmo recorre el árbol a partir de la raíz.

Ejercicio 2.12.2

Diseñar un algoritmo para reconocer oraciones.
Es posible realizar un algoritmo ascendente que comience aislando "el" y al ver que es <artículo> se revisa si es <frase nombre> con lo cual se espera <nombre> y si no hay <adjetivo> se completa la búsqueda de <frase nombre>. Entonces se busca <verbo> y si se halla se busca <complemento>, etc.
Por supuesto pueden generarse gramáticas mucho más complejas con cientos de producciones y constantes, como las de los lenguajes algorítmicos.
Un compilador puede proceder reconociendo la sintaxis en la forma descrita. Al completar ciertas estructuras puede generar las instrucciones de un lenguaje de nivel más básico (ensamblador, p-código o lenguaje de máquina) que calcule la instancia de la estructura.
Hay una conexión muy importante entre la generación de estructuras mediante producciones y el análisis de consistencia de un conjunto de cláusulas de Horn por resolución para determinar su consistencia. Las cláusulas se pueden representar por producciones. El lado izquierdo es el literal positivo. El derecho (puede ser vacío) es el conjunto de las proposiciones que aparecen como literales negativos. Si no hay literal positivo se pone F (falso) en el lado izquierdo. Se procede a hacer las sustituciones como para generar oraciones. Si se pueden hacer todas resulta el conjunto inconsistente. Si no es consistente. El método puede justificarse comparando las operaciones con el método de resolución (1.11).

Ejemplo

Sea el conjunto :
$S = \{p \vee negr \vee \neg s, q \vee \neg p, r \vee \neg p, s\}$
En lo que sigue omitimos los símbolos $\vee$ por claridad.

	cláusulas	resolventes	producciones	sustituciones
C_1	$p\neg r\neg s$		P_1 $p :- r, s$	
C_2	$q\neg p$		P_2 $q :- p$	
C_3	$r\neg p$		P_3 $r :- p$	
C_4	s		P_4 $s :-$	
C_5	$q\neg r\neg s$	C_1, C_2, p	P_5 $q :- r, s$	P_2P_1
C_6	$p\neg r$	C_1, C_4, r	P_6 $p :- r$	P_4P_1
C_7	$q\neg r$	C_2, C_6, s	P_7 $q :- p$	P_2P_6

Tanto el método de resolución como en el de las producciones pueden continuar hasta que las expresiones se repiten sin dar F. Esto significa que la S original es consistente.
Un caso de inconsistencia es el siguiente:
$S = \{\neg p \vee \neg q \vee \neg r, p \vee \neg r \vee \neg t, r, t \vee \neg q, q, t \vee \neg p \vee \neg r, p \vee \neg q \vee \neg r \vee \neg s\}$

	cláusulas	resolventes	producciones	sustituciones
C_1	$\neg p \neg r \neg r$		P_1 $F :- p, q, r$	
C_2	$p \neg r \neg t$		P_2 $p :- r, t$	
C_3	r		P_3 $r :-$	
C_4	$t \neg q$		P_4 $t :- q$	
C_5	q		P_5 $q :-$	
C_6	$t \neg p \neg r$		P_6 $t :- p, r$	
C_7	$p \neg q \neg r \neg s$		P_7 $p :- q, r, s$	
C_8	$\neg q \neg r \neg r \neg t$	C_1, C_2, p	P_8 $F :- r, t, q, r$	$P_2 P_1$
C_9	$\neg q \neg r \neg t$	C_8, C_3, r	P_9 $F :- q, r, t$	$P_3 P_8$
C_{10}	$\neg q \neg r \neg q$	C_9, C_4, t	P_{10} $F :- q, r, q$	$P_4 P_9$
C_{11}	$\neg r \neg q$	C_{10}, C_5, q	P_{11} $F :- q, r$	$P_5 P_{10}$
C_{12}	$\neg r$	C_{11}, C_5, q	P_{12} $F :- r$	$P_5 P_{11}$
C_{13}	F	C_3, C_{12}, r	P_{13} $F :-$	$P_3 P_{12}$

Capítulo 3

Presentación axiomática de la Lógica

3.1 Sistemas Axiomáticos

Son una forma de presentar un cuerpo de conocimientos que consiste en:

- Un conjunto finito de **axiomas** que son proposiciones consideradas válidas.
- Un conjunto finito de **reglas de inferencia** que indican como producir nuevas proposiciones a partir de los axiomas y proposiciones ya demostradas (teoremas).

La **demostración** de una proposición es una sucesión ordenada de proposiciones: axiomas, teoremas, reglas de inferencia y proposiciones precedentes en la sucesión que termina en la proposición a demostrar, la cual queda así como teorema.
En nuestra notación de cálculo proposicional y de predicados denotaremos este proceso como:

$$S \vdash A$$

donde S es la citada sucesión de axiomas, reglas de inferencia y teoremas y A es el teorema que resulta. Los elementos de S se llaman **hipótesis** y A es la **conclusión**.
Se dice que A es **consecuencia lógica** de S. La expresión $\emptyset \vdash A$, que se abrevia $\vdash A$ significa que A es una **tautología**. Si es $S \vdash F$ se dice que S es inconsistente.
Nótese la semejanza con lo dicho para inferencia, consecuencia lógica, tautología e inconsistencia en 1.4, donde denotamos la consecuencia lógica por:

$$S \models A$$

Sin embargo los significados son totalmente diferentes. En el enfoque de 1.4, llamado **enfoque semántico** la verdad de una proposición o fórmula depende de la verdad de las premisas en S y la verdad de estas descansa en la **interpretación**. En el **enfoque sintáctico**, que aquí veremos, la verdad de una proposición o fórmula depende de los axiomas, de las reglas de inferencia y del proceso de demostración.
Todo sistema axiomático debe ser **consistente**. Es decir, el conjunto S de axiomas y reglas de inferencia no puede ser tal que $S \vdash F$.

Es conveniente que el sistema sea **mínimo** en el sentido de que no haya axiomas que puedan deducirse de otros, ya que serían teoremas y siempre sería posible quitarlos de la lista de axiomas sin que se perdiera ninguna de las conclusiones. Esto también se expresa diciendo que los axiomas deben ser **independientes**.
El sistema axiomático debe ser **sólido**, lo cual significa que toda proposición **derivada** o teorema debe ser una proposición **válida**. Esto se expresa por:

$$\vdash A \rightarrow \models A$$

Ciertos sistemas axiomáticos tienen la propiedad inversa: toda proposición válida es un teorema, es decir, se puede demostrar a partir de los axiomas y reglas de inferencia. Un sistema con tal propiedad se dice **completo**. Se expresa:

$$\models A \rightarrow \vdash A$$

Todo sistema axiomático debe ser sólido, si no es inaceptable como sistema axiomático. Por otra parte puede no ser completo. Puede haber proposiciones válidas indemostrables.

3.2 Sistema axiomático para el cálculo proposicional

Un sistema axiomático para el cálculo proposicional puede construirse con tres axiomas y una regla de inferencia:

A_1	$X \supset (Y \supset X)$
A_2	$(X \supset (Y \supset Z)) \supset ((X \supset Y) \supset (X \supset Z))$
A_3	$(\neg X \supset \neg Y) \supset ((\neg X \supset Y) \supset X)$
R_1	Si $X \supset Y$ y X son teoremas entonces Y es un teorema.

Donde X, Y, Z son fórmulas.
Se admite que A_1, A_2, A_3, R_1 son teoremas. Si un sistema es sólido podemos sustituir la palabra teorema por **válido** o por **tautología**. Al aplicar la regla de inferencia a cualquier teorema resulta un teorema. Por lo tanto el sistema así definido es sólido.
Todo lo que se deriva de A_1, A_2, A_3 usando R_1 es válido, así que también el sistema es consistente.

Ejercicios 3.2.1

Demostrar los teoremas siguientes:

1. $p \supset p$
2. $p \supset \neg\neg p$
3. $\neg\neg p \supset p$
4. $(\neg p \supset \neg q) \supset (q \supset p$
5. $(p \supset q) \supset ((\neg p \supset q) \supset q$

Ejemplo 3.2.1

La demostración del primero sería:

1	$(p \supset (p \supset p) \supset p)$	por A_1
2	$(p \supset ((p \supset p) \supset p)) \supset ((p \supset (p \supset p) \supset (p \supset p))$	por 1 y A_2
3	$((p \supset (p \supset p) \supset (p \supset p))$	por 1,2 y R_1
4	$(p \supset (p \supset p)$	por A_1
5	$(p \supset p)$	por 3,4 y R_1

Se pueden introducir como definiciones los demás conectivos del cálculo proposicional definiendo:

1. $X \vee Y \equiv \neg X \supset Y$
2. $X \wedge Y \equiv \neg(X \supset \neg Y$
3. $(X \equiv Y) \equiv (X \supset Y) \wedge (Y \supset X)$

Si se demuestran, a partir de estas definiciones, las propiedades de $\vee, \wedge, \equiv$ éstas se pueden usar en las demostraciones.

Ejercicios 3.2.2

Demostrar que $X \vee Y \equiv Y \vee X$.
Ver que $X \vee Y$ es un teorema si lo es X o Y. Usar 3.2.1 1,2,5.

3.2.1 Metateoremas

Véase que no hay reglas fijas para hacer demostraciones (equivalentes a los algoritmos de mostrar consistencia en el enfoque semántico).
Son muy útiles para hacer demostraciones los **teoremas que se refieren a propiedades de los teoremas**. Se los llama **metateoremas**. Mencionaremos los siguientes:

1. **Principio de deducción de Tarsky-Herbrand**

   $S, B \vdash A \leftrightarrow S \vdash (B \supset A)$

   Es decir, A es deducible del conjunto S y la fórmula B si y sólo si de S se deduce $B \supset A$. Se ve que la condición $S \vdash (B \supset A)$ es necesaria para que valga $S, B \vdash A$. Sea $S, B \vdash A$ válida. Entonces si es válida B lo es A por R_1, así que $S, B \vdash A$ es válida. Que la condición es necesaria se prueba por inducción sobre el número de pasos de la prueba $S, B \vdash A$. Si ésta se desarrolla en los pasos $A_1, A_2, ..., A_n$ se ve que en el paso A_1 es $S \vdash (B \supset A_1)$. Esto se supone válido para el paso A_k y se prueba para el A_{k+1}. La demostración detallada puede verse en Mendelson p.32.

   El teorema de deducción es el más usado en Matemáticas. Para probar que una implicación es cierta $B \supset A$ si valen ciertas premisas, es decir $S \vdash (B \supset A)$ se supone B verdadera y se demuestra que de S, B resulta A.

Ejemplo 3.2.1.1

Supongamos (S)válidas las reglas de suma de exponentes: $r^m r^n = r^{m+n}$ y supongamos que $a^{logx} = x$ de esto resulta la implicación: Si (B) $z = x_1 x_2$ entonces (A) $logz = logx_1 + logx_2$. Para demostrar esto hay que ver que de S y B resulta A. Es la demostración conocida $z = x_1 x_2 = a^{logx_1} a^{logx_2} = a^{logx_1 + logx_2} = z = a^{logz}$ Es decir $logz = logx_1 + logx_2$.

2. **Regla de intercambio**. Permite cambiar una fórmula o subfórmula por otra equivalente:

   Si X,$Y \supset Z$ y $Z \supset Y$ son teoremas y

   si Y es X o una subfórmula de X,

   entonces si reemplazamos en X una Y por la Z resulta un teorema. Por ejemplo en cualquier teorema en que aparezca $p \supset q$ lo podemos reemplazar por $\neg p \vee q$

3. **Sustitución uniforme**. Permite reemplazar una fórmula en un teorema:

   Si $X(p)$ es un teorema que contiene la proposición p y reemplazamos en él la proposición p por la fórmula A, la expresión que resulta es un teorema.

Si en los metateoremas anteriores reemplazamos la palabra **teorema** por **tautología**, es obvio que son válidos, pues la tautología vale cualquiera sea el valor de sus elementos. Luego los metateoremas son válidos en todo sistema axiomático sólido y completo donde cada teorema es una tautología y viceversa.

Ejemplo 3.2.1.1

Como ejemplo de aplicación de las reglas de deducción y sustitución veamos la demostración de la validez del esquema llamado **prueba por casos**:
Si $S, B \vdash A$ y $S, \neg B \vdash A$ entonces $S \vdash A$

1	$S, B \vdash A$	hipótesis
2	$S \vdash (B \supset A$	1, deducción
3	$((p \supset q) \supset ((\neg p \supset q) \supset q))$	teorema
4	$((B \supset A) \supset ((\neg B \supset A) \supset A))$	3, sustitución
5	$S \vdash ((\neg B \supset A) \supset A))$	2,4,R_1
6	$S, \neg B \vdash A$	hipótesis
7	$S \vdash (\neg B \supset A)$	6, deducción
8	$S \vdash A$	7,5,R_1

3.3 Completitud del cálculo proposicional

Daremos una idea de la demostración de que el sistema axiomático A_1, A_2, A_3, R_1 para el cálculo proposicional es completo.
Supondremos que toda fórmula tiene solamente los conectivos $\neg, \supset$. Todo otro conectivo puede reducirse a estos.

Teorema 3.3.1 En toda fórmula A construida con las proposiciones $p_1, p_2, .., p_n$ en la cual I es una interpretación, si p'_k es p_k cuando $I(p_k) = V$ y p'_k es $\neg p_k$ cuando $I(p_k) = F$ y además A' es A cuando $I(A) = V$ y por $\neg A$ cuando $I(A) = F$, entonces $\{p'_1, p'_2, .., p'_n\} \vdash A'$

Véase que se trata de hacer un cambio de proposiciones p por p' de manera que sean todas verdaderas en la I dada. Con ellas es directo construir cualquier proposición A' que es A o su negación. De esa construcción se puede pasar directamente a A o $\neg A$ a partir de las p mediante aplicación de los operadores $\neg$ y $\supset$ a través de fórmulas válidas en I.
Se puede demostrar el teorema por inducción sobre el número de conectivos. Si no hay ninguno A es una de las p_k y el resultado es inmediato. Si hay varios conectivos se puede ver que A tiene la fórmula $B \supset C$ o bien $\neg D$ (donde B, C, D son fórmulas con menos conectivos que A. Por la hipótesis de inducción el resultado es válido para B', C', D' transformadas como se dice en el teorema. Luego:
$\{p'_1, p'_2, .., p'_n\} \vdash B'$;
$\{p'_1, p'_2, .., p'_n\} \vdash C'$;
$\{p'_1, p'_2, .., p'_n\} \vdash D'$; así que:
$\{p'_1, p'_2, .., p'_n\} \vdash A'$ Hay que considerar los diferentes casos (Ver Mendelson p.36).

Teorema 3.3.2 El sistema axiomático A_1, A_2, A_3, R_1 para el cálculo proposicional es completo

Sea A una tautología constituida por las proposiciones $\{p'_1, p'_2, .., p'_n\}$. Se pueden construir 2^n expresiones del tipo $\{p'_1, p'_2, .., p'_n\} \vdash A$ que son verdaderas ya que una tautología es válida para todas las interpretaciones. Por el principio de deducción:
$\{p'_1, p'_2, .., p'_{n-1}\} \vdash (p'_n \supset A)$
$\{p'_1, p'_2, .., p'_{n-1}\} \vdash (\neg p'_n \supset A)$
y éstas son verdaderas. Por lo tanto son verdaderas
$\{p'_1, p'_2, .., p'_{n-1}\} \vdash A$ (por el principio de prueba por casos).
Prosiguiendo la reducción se llega a: $\vdash A$.
Es decir A es deducible de los postulados (por el teorema 3.3.1, el principio de deducción y la prueba por casos).

3.4 Uso de los sistemas axiomáticos

En cálculo proposicional, donde hay algoritmos para probar la consistencia, el uso de sistemas axiomáticos no es muy necesario. Es como una introducción para usarlo en otras áreas como el cálculo de predicados o la aritmética.
Las deducciones basadas en axiomas y la regla R_1 (modus ponens) se llaman pruebas de tipo Hilbert.
Pueden darse sistemas sin axiomas y con varias reglas de deducción (pruebas tipo Gentzen).
Las reglas pueden dar la semántica de cada conectivo. Estos sistemas se llaman de **deducción natural** . Damos a continuación un sistema de deducción natural para el cálculo proposicional. En lo que sigue la expresión de la forma:

$$\frac{S_1, S_2, .., S_n}{S}$$

significa que del conjunto de fórmulas $\{S_1, S_2, .., S_n\}$ se deduce S. Escribamos $H \to C$ válido si H tiene como consecuencia C.
Regla de monotonicidad:

$$\frac{}{S, A \to A} \qquad \frac{S \to A}{S, B \to A}$$

Reglas de introducción:

$(\wedge)$ $\dfrac{S \to A \; S \to B}{S \to A \wedge B}$

$(\vee)$ $\dfrac{S \to A}{S \to A \vee B}$ $\qquad$ $\dfrac{S \to B}{S \to A \vee B}$

$(\supset)$ $\dfrac{S, A \to B}{S \to A \supset B}$

$(\neg)$ $\dfrac{S, A \to F}{S \to \neg A}$

$(\equiv)$ $\dfrac{S, A \to B \; S, B \to A}{S \to A \equiv B}$

Reglas de eliminación:

$(\wedge)$ $\dfrac{S \to A \wedge B}{S \to A}$ $\qquad$ $\dfrac{S \to A \wedge B}{S \to B}$

$(\vee)$ $\dfrac{S \to A \vee B \; S, A \to C \; S, B \to C}{S \to C}$

$(\supset)$ $\dfrac{S \to A \supset B}{S, A \to B}$

$(\neg)$ $\dfrac{S \to A \; S \to \neg A}{S \to F}$ $\qquad$ $\dfrac{S \to \neg\neg A}{S \to A}$

$(\equiv)$ $\dfrac{S \to A \equiv B}{S, A \to B}$ $\qquad$ $\dfrac{S \to A \equiv B}{S, B \to A}$

Ejemplo 3.4.1

Apliquemos esta forma de deducción para demostrar la validez del esquema de prueba por casos:
$\{h, h \supset (p \vee q), p \supset c, q \supset c\} \to c$

Llamemos S al conjunto de las hipótesis. Una prueba sería:

1	$S, h \supset (p \vee q) \rightarrow h \supset (p \vee q)$	regla básica
2	$S \rightarrow h \supset (p \vee q)$	monotonía
3	$S, h \rightarrow p \vee q$	2 y elim. $\supset$
4	$S \rightarrow p \vee q$	monotonía
5	$S, p \supset c \rightarrow p \supset c$	regla básica
6	$S \rightarrow p \supset c$	monotonía
7	$S, p \rightarrow c$	6 y elim. $\supset$
8	$S, q \supset c \rightarrow q \supset c$	regla básica
9	$S \rightarrow q \supset c$	monotonía
10	$S, q \rightarrow c$	9 y elim. $\supset$
11	$S \rightarrow c$	4,7,10 y elim. $\vee$

3.5 Axiomas del cálculo de predicados

El cálculo de predicados puede axiomatizarse sin dificultad si se excluye la igualdad. Un sistema puede ser:

A_1	$X \supset (Y \supset X)$
A_2	$(X \supset (Y \supset Z)) \supset ((X \supset Y) \supset (X \supset Z))$
A_3	$(\neg X \supset \neg Y) \supset ((\neg X \supset Y) \supset X)$
R_1	Si $X \supset Y$ y X son teoremas entonces Y es un teorema.
R_2	Si x es libre en el teorema P, entonces $\forall x P$ es un teorema.

Donde X, Y, Z son fórmulas.
El $\exists x A$ se define como $\neg\forall\neg A$
El metateorema de deducción (ver 3.2.1): $S, B \vdash A \leftrightarrow S \vdash (B \supset A)$
no es válido en general. Por ejemplo: $P \vdash \forall x P$ es válido, pero $P \supset \forall x P$ no lo es, a menos que P sea un teorema.
Sistema de deducción natural. Se puede construir un sistema de deducción natural para el cálculo de predicados agregando a las reglas vistas en 3.4 las siguientes:

$(\forall)$ $\dfrac{S \vdash A(x)}{S \vdash \forall x A(x)}$ x no libre en S

$(\exists)$ $\dfrac{S \vdash A(t)}{S \vdash \exists x A(x)}$

$(\forall)$ $\dfrac{S \vdash \forall x A(x)}{S \vdash A(t)}$ t no libre en A(t)

$(\exists)$ $\dfrac{S \vdash \exists x A(x)}{S \vdash A(c)}$ c no en S ni en A(x)

La igualdad puede ser definida como una función predicativa E con las propiedades siguientes (reflexiva, recíproca y transitiva):

$\forall x E(x, x)$

$\forall x \forall y[E(x,y) \supset E(y,x)]$
$\forall x \forall y \forall z[E(x,y) \wedge E(y,z) \supset E(x,z)]$

The equality can also be characterized by the following inference rules:

$$\frac{E(s_1,t_1) \wedge E(s_2,t_2) \wedge ... \wedge E(s_m,t_m)}{P(s_1,s_2,..,s_m) \equiv P(t_1,t_2,..,t_m)}$$

$$\frac{E(s_1,t_1) \wedge E(s_2,t_2) \wedge ... \wedge E(s_n,t_n)}{E(f(s_1,s_2,..,s_n), f(t_1,t_2,..,t_n))}$$

P denota una variable predicativa m-aria, f una función n-aria.

3.6 Teorías de primer orden

El cálculo de predicados puede utilizarse en la formalización de teorías mediante interpretaciones especiales dependientes de la materia que trate la teoría a formalizar. De hecho la formalización de la Aritmética, la Geometría, la Física, incorporan el cálculo de predicados de primer orden, aunque ésto no se declare explícitamente en las exposiciones.
Una propiedad importante de la teoría es la **decibilidad**. Una teoría es decidible si toda fórmula puede ser demostrada o refutada, esto es, puede demostrarse su contraria a partir de los axiomas y reglas de inferencia de la teoría.
Por otra parte se dice que una fórmula es **computable** si existe un algoritmo tal que a partir de axiomas y teoremas antes demostrados y usando las reglas de inferencia se puede demostrar tal fórmula. Veremos que ambos conceptos están relacionados.
Existen teorías no formalizadas que se refieren a un campo del universo (real, imaginario, pasado, presente o futuro) pero en ellas siempre existe un lenguaje, proposiciones que afirman o niegan, ciertas proposiciones que se toman como verdaderas (axiomas) y ciertos métodos de razonar o inferir afirmaciones o negaciones a partir de los axiomas.
Hemos visto que el cálculo de predicados contiene estos elementos. Cuando están declarados explícitamente y se cumplen ciertas condiciones de solidez decimos que hay una teoría **formal**.
En una teoría formal que incorpore el cálculo de predicados se tienen los elementos siguientes.

- Una **simbología** formada por un conjunto de **constantes funcionales o predicativas** que nos indican cuales serán los tipos de afirmaciones y relaciones que aparecerán en la teoría.
- Un **lenguaje** o conjunto de fórmulas obtenidas mediante el **cálculo de predicados** a partir de la simbología. Estas fórmulas son en principio vacías de contenido.
- Una **estructura** que consiste en una interpretación del lenguaje. Se da un **dominio de interpretación** y a cada constante funcional o predicativa se le asigna una función o predicado del número adecuado de variables con valores de los argumentos en el dominio de interpretación. Las funciones comunes tienen valores en el dominio de interpretación y las predicativas en el dominio $\{V, F\}$. Esto permite dar una interpretación a los términos y fórmulas del lenguaje.

- Una teoría es un conjunto de fórmulas en las cuales se ha distinguido un subconjunto finito especial (axiomas) de cuyos elementos, mediante reglas especificadas se obtienen **teoremas** que son proposiciones válidas de la teoría. Los axiomas son los del cálculo de predicados más los que se agregan como axiomas específicos de la teoría.

Las teorías basadas en el cálculo de predicados de primer orden se llaman **teorías de primer orden**. Las teorías de orden superior, por ejemplo las que admiten funciones comunes o predicaticas como variables, tienen en general mayor poder expresivo, pero su poder deductivo es menor. Por ejemplo no se pueden definir en ellas sistemas axiomáticos completos y sólidos. Para las aplicaciones de la Ciencia y las Matemáticas suelen ser suficientes las teorías de primer orden.

Ejercicio 3.6.1

Señalar en el ejemplo 2.3.1 cuales son la simbología, el lenguaje y la estructura. ¿ Como se podría introducir una teoría?
La utilidad de las teorías es que al ser establecidas permiten:

1. una exposición clara del tema al que se refieren,
2. un manejo algorítmico del conocimiento del tema,
3. una jerarquización de las afirmaciones que facilita la aplicación y la verificación de las afirmaciones.

Por otra parte existen otros métodos además del axiomático para establecer la verdad de una fórmula dada. Un ejemplo es el uso de tablas de verdad en el cálculo proposicional. El establecimiento del valor de verdad a partir de axiomas y reglas de deducción debe usarse cuando los métodos semánticos (como las tablas de verdad) o algorítmicos (como el cálculo basado en la resolución o en el análisis algorítmico de una fórmula) no son posibles. Se han hecho intentos para crear algoritmos que busquen demostraciones a partir de axiomas.
El trabajo de crear un sistema axiomático para un campo del conocimiento no es simple. Debe definirse el dominio de interpretación que debe contener los conceptos básicos (en el caso de la Física serían masa, fuerza, velocidad,etc.) la traducción de los hechos (por ejemplo observaciones) al lenguaje antes definido y la elección de cuales afirmaciones se elegirán como axiomas de manera que se puedan definir todas las observaciones consideradas correctas.
Debe evitarse la inconsistencia (falta de solidez) e investigarse la incompletitud (existencia de verdades que no pueden deducirse de la teoría) pues esto indica que la teoría podría ser ampliada. Por último puede encontrarse una teoría que abarque teorías anteriores como casos particulares.

3.6.1 Modelos de una teoría

Una teoría T con un lenguaje L puede tener **modelos** es decir, interpretaciones que hacen verdaderos los axiomas de la teoría. Los teoremas son consecuencia lógica de la teoría,

así que todo modelo hace verdaderos los teoremas. Luego:

Teorema 3.7.1 Los teoremas de la teoría T son las fórmulas de L que son verdaderas para todo modelo de T

Si definimos teorema como en los sistemas axiomáticos, como una proposición o fórmula obtenida a partir de los postulados mediante las reglas de inferencia, la definición será equivalente a la dada aquí, siempre que el sistema axiomático sea obtenido de los axiomas propios de la teoría y un sistema axiomático sólido y completo del cálculo de predicados.
La teoría, o más exactamente metateoría, de modelos tiene por objeto construir modelos de teorías dadas consistentes. Vimos un ejemplo en el método de Herbrand para construir un modelo numerable en la lógica de predicados. Puede extenderse a toda teoría con un número numerable de axiomas. Se han dado otros métodos dentro de la teoría de modelos con diferentes propiedades.
La cuestión inversa consistiría en tener una estructura matemática como la de los números naturales o los grafos y de allí construir una teoría que tenga aquella estructura como modelo.
En el caso de los números naturales la simbología puede ser $\{0, s, +, *\}$ donde 0 es una constante, s la función que a cada elemento le hace corresponder otro llamado **siguiente** y $+, \times$ son las operaciones suma y multiplicación (funciones binarias). Así 2 se representa $s(s(0))$. Con esto se pueden representar todos los elementos de la Aritmética. La axiomatización puede hacerse por los conocidos axiomas de Peano.
Otros problemas sobre las teorías y sus modelos llevan a las siguientes distinciones :

- Una teoría es **decidible** si existe un algoritmo de un número finito de pasos que decida si una fórmula es o no un teorema. Para probar la decidibilidad basta demostrar que tal algoritmo existe, sin que sea necesario construirlo.

- Una teoría es **completa** si toda fórmula es un teorema o su negación lo es. Es decir, si puede decirse con seguridad si la fórmula es o no un teorema. Nótese que una teoría puede ser decidible y no completa. Más adelante veremos que si es completa es decidible.

- Una teoría es **categórica** si todos sus modelos son isomorfos, es decir son en realidad el mismo modelo con diferentes nombres de sus elementos de interpretación.

Ejemplo 3.7.1

En la teoría del orden parcial hay algunos modelos para los cuales dados dos elementos x, y se cumple siempre que $x \leq y$ o $y \leq x$. ésto es válido si el modelo son los números reales, pero por ejemplo si el modelo es el de los conjuntos con la relación de inclusión la proposición dicha (expresada por la fórmula: $\forall x \forall y(x \leq y \vee y \leq x)$ no se cumple. La teoría no es categórica.

La **cardinalidad** de un modelo es el número cardinal de sus elementos (finito, numerable, continuo, etc.). Si una teoría admite modelos de cardinalidad diferente

no pueden ser isomorfos, así que la teoría no es categórica. Una exigencia más débil es la α-categoricidad:

Una teoría es α-categórica si todos sus modelos de cardinalidad α son isomorfos.

El siguiente teorema relaciona los conceptos vistos. Es consecuencia inmediata de las definiciones.

Teorema 3.7.2 Una teoría categórica es α-categórica para todo número cardinal α.

También se verifica el siguiente:

Teorema 3.7.3 Una teoría consistente que no tiene un modelo finito y es α-categórica para un cardinal α infinito es completa

.

Sea T sin un modelo finito. Supongamos que es no completa. Debe tener una fórmula A tal que las dos teorías $T \vee A$ y $T \vee \neg A$ sean ambas consistentes (si no hubiera tal A sería completa). Los modelos correspondientes son infinitos pues T no admite modelos finitos. Según el teorema de Löwenheim Skolem cada teoría admite un modelo de cardinalidad α (hemos visto sólo el caso restringido del teorema en que α es $\aleph_0$ pero el teorema es general). Como A debe ser V en el primer modelo y F en el segundo ambos modelos no pueden ser iguales, pues una interpretación hace la fórmula A verdadera y la otra la hace falsa. Luego hay dos modelos diferentes para T, ambos de cardinalidad α.

Teorema 3.7.4 Una teoría completa es semi-decidible

Sea T una teoría completa y A una fórmula del lenguaje T. Entonces uno de los conjuntos de fórmulas $T \vee A$, $T \vee \neg A$ es inconsistente. Esto puede determinarse, por ejemplo, mediante el método de reducción. Si el proceso termina, con esto se decide si A es un teorema o lo es $\neg A$. Pero para asegurar la decibilidad estricta hay que demostrar que el proceso siempre termina.

Si una teoría es incompleta puede completarse agregando nuevos axiomas. Pero, si estos son muy fuertes, el número de axiomas se puede reducir de modo que no haya ningún modelo interesante.

Se puede demostrar que no puede obtenerse ninguna teoría completa (ni siquiera decidible) de la aritmética de los números enteros ni de los grupos finitos.

Se puede demostrar que el cálculo de predicados no es, en general, decidible. Un caso especial en que las funciones predicativas son de una variable y las funciones comunes son constantes (cero variables) conocido como **cálculo de predicados monádico** es decidible. El cálculo de predicados no es, en general, decidible.

3.7 Algoritmos y máquina de Turing

Una máquina de Turing (1936) es un modelo simple de un computador de algoritmos.

La MT consiste en:

1. Una memoria, representada por una cinta de celdas. En cada celda puede haber un símbolo de un cierto alfabeto S. Uno de los símbolos debe ser un blanco B. Debe haber por lo menos dos símbolos diferentes.
2. Una cabeza lectora-escritora que puede revisar sucesivamente las celdas y:
   - revisar (leer) el contenido de celda dejando disponible el símbolo que contiene.
   - escribir un símbolo en la celda revisada, borrando el que había.
   - mover la cinta un lugar a la izquierda o a la derecha para ir revisando las celdas contiguas.
3. Un registro que guarda uno de un conjunto finito Q de símbolos que representan estados diferentes de la máquina. Se distinguen:
   - un estado inicial $q_0 \epsilon Q$
   - uno o más estados finales $q_f \epsilon Q$
4. Por último existe un conjunto de instrucciones. Para cada estado hay una y sólo una instrucción. La instrucción es una función que para cada valor leído posible le hace corresponder una terna de los elementos siguientes:
   - un símbolo que indica el nuevo valor a escribir en la cinta.
   - el nuevo estado a guardar en el registro (el cual indicará la proxima instrucción a ejecutar)
   - un indicador $+$ o $-$ que indica si la cinta, después de ejecutar la instrucción irá a la derecha o a la izquierda.

Una computación comienza en el estado q_0 con la cabeza apuntando a la celda 0 de la cinta.

El q_0 y el contenido de esa celda determinan la primera terna (instrucción) a ejecutar. La ejecución consiste en escribir en la celda actual el primer elemento de la terna, poner en el registro de estados el segundo elemento y mover la cinta una celda a la izquierda o a la derecha.

La ejecución avanza hasta llegar a un estado final o prosigue indefinidamente. Si termina lo que queda en la cinta es el resultado de la computación.

Ejemplo 3.8.1 Un ejemplo muy simple es la máquina que decide si un número expresado en binario tiene un número par o impar de unos.

Sea: $S = \{0, 1, B\}$; $Q = \{q_0, q_1, q_f\}$.

Las instrucciones son:

estado	0	1	B
q_0	$(0, q_0, +)$	$(0, q_1, +)$	$(0, q_f, \perp)$
q_1	$(0, q_1, +)$	$(0, q_0, +)$	$(1, q_f, \perp)$

donde $\perp$ significa indefinido.

Ejercicio 3.8.1

Aplicar la máquina anterior a los números: 1011 y 0110

Tesis de Turing: Toda computación que puede llamarse un algoritmo puede ser realizada por una máquina de Turing

La proposición no es un teorema, ya que el término **algoritmo** no está bien definido. Usualmente se entiende por algoritmo una receta o procedimiento expresado en cierto lenguaje que describe una sucesión de operaciones sobre datos tomados de un cierto conjunto. En cada operación se especifica la siguiente y ciertas transformaciones en los datos. Eventualmente se terminan las operaciones y dejan un resultado expresado en forma de datos del mismo conjunto.

Los resultados son una función de los datos iniciales y se dice que tal función es computable por la máquina de Turing.

Es importante saber si dada una máquina de Turing es posible saber si la computación que realiza termina o prosigue indefinidamente. Esto es conocido como el **problema de la parada**. La función a calcular tiene como argumento cualquier máquina de Turing y como resultado uno de los dos valores V o F según la máquina se detenga o prosiga indefinidamente. La imposibilidad de construir una máquina de Turing que resuelva este problema se afirma en el siguiente:

Teorema El problema de la parada en las máquinas de Turing es no computable

Sea T una máquina de Turing con datos t (estado inicial de la cinta). Sea D una supuesta MT que tomando (T,t) como datos decide si al correr la T con datos cualesquiera hay parada (V) o no la hay (F). Supongamos, para simplificar D que t=T, es decir la T corre con datos que son precisamente la descripción binaria de T. Podemos entonces definir una máquina E tal que lee la expresión de T y la copia en la cinta para ser considerada como los datos de T. Así T constituye todos los datos de E. Supongamos que E procesa los datos T y puede dar como salida V (si la T pararía al procesar los datos T) o F (si no pararía). Hagamos una pequeña modificación a E de tal forma que si halla V (habría parada) entre en un lazo. Sea U la MT así modificada. Tal U al procesar el código de cualquier MT (que use como datos su propio código) para si tal MT no para y no para si tal MT para. Pero al aplicar tal U a sí misma (a su propio código) vemos que tal U para cuando no para y viceversa. Pero tal U no puede existir. Luego la E y la D con la cual la hemos construido no pueden existir.

Índice General