

Representación del Conocimiento

Lógica Proposicional

Dr. Alejandro Guerra-Hernández

Universidad Veracruzana

Centro de Investigación en Inteligencia Artificial
Sebastián Camacho No. 5, Xalapa, Ver., México 91000

<mailto:aguerra@uv.mx>

<http://www.uv.mx/personal/aguerra/rc>

Maestría en Inteligencia Artificial 2020



Universidad Veracruzana

Organización

- 1 Sintaxis
- 2 Semántica
- 3 Inducción matemática
- 4 Robustez
- 5 Completitud
- 6 Formas Normales



Metavariables y fórmulas

- ▶ Las reglas de derivación que se han introducido en el capítulo precedente, son **válidas** para **cualquier** fórmula que podamos formar con el lenguaje de la proposicional.
- ▶ A partir de los **átomos proposicionales** podemos usar **conectivos** lógicos para crear fórmulas lógicas más complejas.
- ▶ El objetivo del capítulo anterior era entender la **mecánica** de las reglas de la deducción natural.



Ejemplo

- Consideren este caso, una aplicación de la regla de prueba ($\rightarrow e$):

1. $p \rightarrow q$ *premisa*
2. p *premisa*
3. q ($\rightarrow e$) 1,2



Continuación del ejemplo

- ▶ Su aplicación es válida aún si sustituimos $p \vee \neg r$ por p y q con $r \rightarrow p$:

$$1. \quad p \vee \neg r \rightarrow (r \rightarrow p) \quad \textit{premisa}$$

$$2. \quad p \vee \neg r \quad \textit{premisa}$$

$$3. \quad r \rightarrow p$$

- ▶ Escribimos las reglas de prueba como esquemas de razonamiento que usan símbolos griegos que pueden ser **substituidos** por fórmulas genéricas.



Átomos proposicionales

- ▶ Lo primero que necesitamos es una fuente **no acotada** de átomos proposicionales: p, q, r, \dots o p_1, p_2, p_3, \dots
- ▶ La cuestión del si tal conjunto es infinito no debería preocuparnos.
- ▶ El carácter no acotado de la fuente es una forma de confrontar que si bien, normalmente necesitaremos una gran cantidad finita de proposiciones para describir un programa de computadora, no sabemos de antemano cuantos vamos a necesitar.



Primer aproximación

- ▶ Que las fórmulas de nuestra lógica proposicional deben ser cadenas de caracteres formadas a partir del **alfabeto**

$$\{p, q, r, \dots\} \cup \{p_1, p_2, p_3, \dots\} \cup \{\neg, \wedge, \vee, \rightarrow, (,)\}$$

es una observación trivial, que **no provee** la información necesaria.

- ▶ **Ejemplo:** (\neg) y $pq \rightarrow$ son cadenas construídas a partir de este alfabeto, aunque no tienen sentido en la lógica proposicional.



Fórmulas bien formadas (definición inductiva)

1. Todo átomo proposicional p, q, r, \dots y p_1, p_2, p_3, \dots es una fbf.
2. Si ϕ es una fórmula bien formada, también lo es $(\neg\phi)$.
3. Si ϕ y ψ son fbf, también lo es $(\phi \wedge \psi)$.
4. Si ϕ y ψ son fbf, también lo es $(\phi \vee \psi)$.
5. Si ϕ y ψ son fbf, también lo es $(\phi \rightarrow \psi)$.



Fórmula bien formada (BNF)

- ▶ Adaptada de Huth y Ryan [3]:

$$\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \quad (1)$$

donde p denota cualquier átomo proposicional y cada ocurrencia de ϕ a la derecha del símbolo $::=$ denota una fórmula bien formada previamente construída.



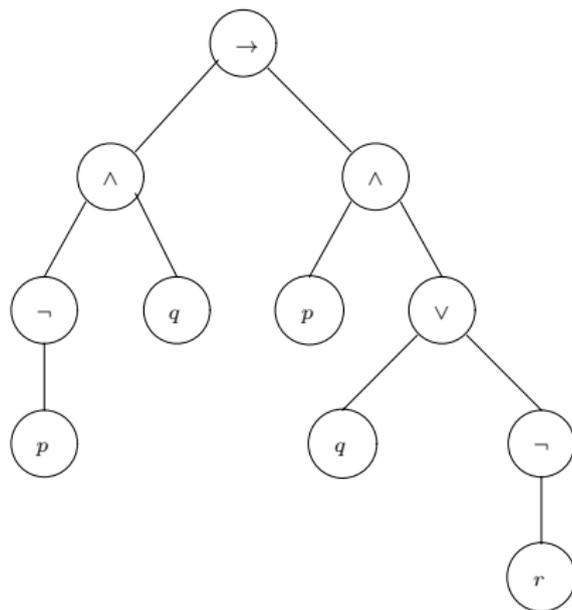
Principio de inversión

- ▶ Solo una regla se aplica a la vez.
- ▶ **Ejemplo:** ¿Es $((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r)))$ una fbf?
- ▶ La última regla aplicada fue (5) dado que la fbf es una implicación con $((\neg p) \wedge q)$ como antecedente y $(p \wedge (q \vee (\neg r)))$ como conclusión.
- ▶ El antecedente es una conjunción (3) entre $(\neg p)$ y q , donde el primer operando es una negación (2) de p que es un átomo (1); igual que q .
- ▶ De igual forma podemos proceder con la conclusión de la expresión original y demostrar que ésta es una fbf.



Arbol sintáctico

- ▶ Las fórmulas tienen una **estructura de árbol**:



Sintaxis y árboles sintácticos

- ▶ La raíz del árbol es una implicación, de manera que la expresión en cuestión es, a su nivel más alto una implicación.
- ▶ Ahora basta probar **recursivamente** que los sub-árboles izquierdo y derecho son también fórmulas bien formadas.
- ▶ Observen que las fórmulas bien formadas en el árbol o bien tienen como raíz un átomo proposicional; o un operador con el número adecuado de operandos.
- ▶ Otro ejemplo de definición **inductiva**.



Sub-fórmulas y Sub-árboles

- ▶ Las sub-fórmulas se corresponden a los sub-árboles de un árbol sintáctico.
- ▶ Siguiendo el ejemplo, esto incluye las hojas p y q que ocurren dos veces; así como r .
- ▶ Luego $(\neg p)$ y $((\neg p) \wedge q)$ en el sub-árbol izquierdo de la implicación.
- ▶ Así como $(\neg r)$, $(p \vee (\neg r))$ y $((p \wedge (q \vee (\neg p))))$ en el sub-árbol derecho de la implicación.
- ▶ El árbol entero es un sub-árbol de si mismo.



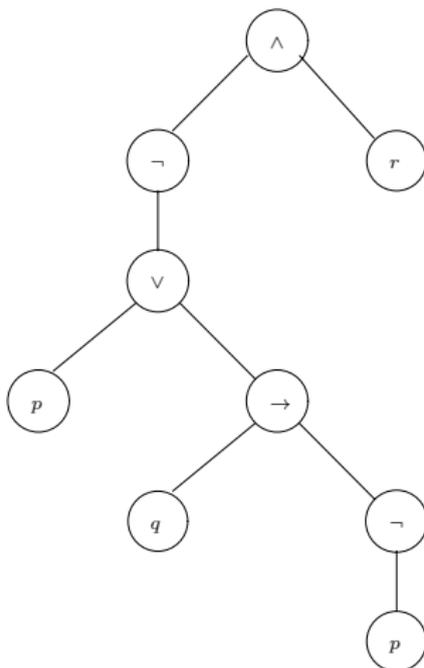
Ejemplo: sub-fórmulas

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



Ejemplo: validación sintáctica

► ¿Porqué representa una fbf?



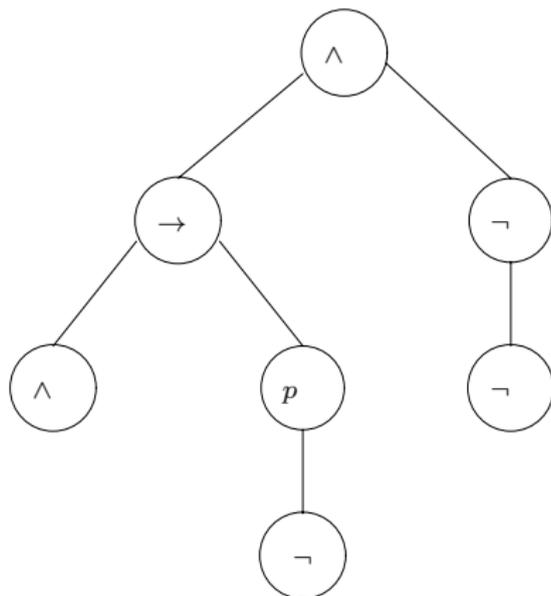
Continuación del ejemplo

- ▶ Todas sus hojas son átomos proposicionales: p dos veces; q y r una.
- ▶ Todos sus nodos internos son operadores lógicos (\neg dos veces, \wedge , \vee y \rightarrow) y
- ▶ El número de sus sub-árboles es el correcto en todos los casos (un sub-árbol para la negación, dos para los demás operadores).
- ▶ La expresión linearizada del árbol puede obtenerse recorriendo el árbol de manera en orden: $((\neg(p \vee (q \rightarrow (\neg p)))) \vee r)$.



Ejemplo de fórmula no bien formada

- ▶ ¿Cómo sabemos que el árbol de la figura no representa una fbf?



Continuación del ejemplo

- ▶ Hay dos razones para ello: primero, las hojas \wedge y \neg , lo cual puede arreglarse diciendo que el árbol está parcialmente construido;
- ▶ Segundo, y definitivo, el nodo para el átomo proposicional p no es una hoja, es un nodo interno.



Validez y Consecuencia Lógica

- ▶ Desarrollamos un cálculo del razonamiento que nos permite verificar si las inferencias de la forma $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ son **válidas**, lo cual quiere decir que a partir de las premisas $\phi_1, \phi_2, \dots, \phi_n$ podemos demostrar ψ usando las reglas de prueba.
- ▶ Desarrollaremos una nueva relación de **consecuencia lógica** entre las premisas y la consecuencia de las inferencias.

$$\phi_1, \phi_2, \dots, \phi_n \models \psi$$



Consecuencia Lógica y Valores de Verdad

- ▶ Esta relación se basa en los valores de verdad de las **proposiciones atómicas** que ocurren en las premisas y la conclusión;
- ▶ así como la forma en que los **operadores lógicos** manipulan estos valores de verdad.
- ▶ ¿Qué es un valor de verdad? Los enunciados **declarativos** que representan las proposiciones atómicas se corresponden con la realidad (verdaderos); o no (falsos).



Ejemplo: Conjunción

- ▶ El valor de verdad de $p \wedge q$ está determinado por tres aspectos: el valor de verdad de p , el valor de verdad de q y el significado de \wedge .
- ▶ El significado de la conjunción captura la observación de que $p \wedge q$ es verdadera ssi p y q son ambas verdaderas; de otra forma $p \wedge q$ es falsa.
- ▶ De forma que, desde la perspectiva de \wedge todo lo que debemos saber es si p y q son verdaderos.
- ▶ Lo mismo para los demás operadores!



Modelos y Valores de Verdad

- ▶ El conjunto de **valores de verdad** contiene dos elementos T y F donde T representa verdadero, y F falso.
- ▶ Un **modelo** de una fórmula ϕ es una asignación de valores de verdad a las proposiciones atómicas que ocurren en ϕ .
- ▶ **Ejemplo:** La función que asigna $T \mapsto q$ y $F \mapsto p$ es un modelo de la fórmula $p \vee \neg q$.



Tabla de verdad: Conjunción

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F



Observaciones

- ▶ Como cada argumento puede tener dos valores de verdad, el número de combinaciones posibles es 2^n donde n es el número de argumentos del operador.
- ▶ **Ejemplo:** $\phi \wedge \psi$ tiene $2^2 = 4$ casos a considerar, los que se listan en la tabla de verdad anterior.
- ▶ Pero $\phi \wedge \psi \wedge \chi$ tendría $2^3 = 8$ casos a considerar.



Tabla de verdad: Disyunción

ϕ	ψ	$\phi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F



Tabla de verdad: Implicación

ϕ	ψ	$\phi \rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T



Semántica de la implicación

- ▶ Se puede pensar en ella como una relación que **preserva** la verdad.
- ▶ Es evidente que $T \rightarrow F$ no preserva la verdad puesto que inferimos algo que es falso a partir de algo que es verdadero. Por ello, la entrada correspondiente en la tabla de verdad de la implicación da como salida falso.
- ▶ También es evidente que $T \rightarrow T$ preserva la verdad;
- ▶ Pero los casos donde el primer argumento tiene valor de verdad F también lo hacen, porque no tienen verdad a preservar dado que el supuesto de la implicación es falso.



La implicación como abreviatura

- ▶ Al menos en cuanto a los valores de verdad se refiere, la expresión $\phi \rightarrow \psi$ es una abreviatura de $\neg\phi \vee \psi$.
- ▶ Las tablas de verdad pueden usarse para probar que cuando la primer expresión mapea a verdadero, también lo hace la segunda.
- ▶ Esto quiere decir que ambas expresiones son **semánticamente equivalentes**.
- ▶ Aunque claro, las reglas de la deducción natural tratan a ambas fórmulas de manera muy diferente, dado que su sintaxis es bien diferente.



Tabla de verdad: Negación y Contradicción

- ▶ Observen que en este caso tenemos $2^1 = 2$ casos que considerar.

ϕ	$\neg\phi$
T	F
F	T

- ▶ También pueden definirse tablas de verdad para la contradicción y su negación: $\perp \mapsto F$ y $\neg\perp \mapsto T$.



Valor de verdad de una expresión

- ▶ **Ejemplo:** ¿Cual es el valor de verdad de la expresión $\neg p \wedge q \rightarrow p \wedge (q \vee \neg r)$, si el modelo es $F \mapsto q$, $T \mapsto p$ y $T \mapsto r$?
- ▶ Una de las ventajas de nuestra semántica es que es **composicional**.
- ▶ Si sabemos los valores de verdad de $\neg p \wedge q$ y $p \wedge (q \vee \neg r)$, entonces podemos usar la tabla de verdad de la implicación para saber el valor de verdad de toda la expresión.
- ▶ De forma que podemos ascender el árbol sintáctico de la expresión propagando los valores de verdad.



Arboles sintácticos y evaluación de una expresión

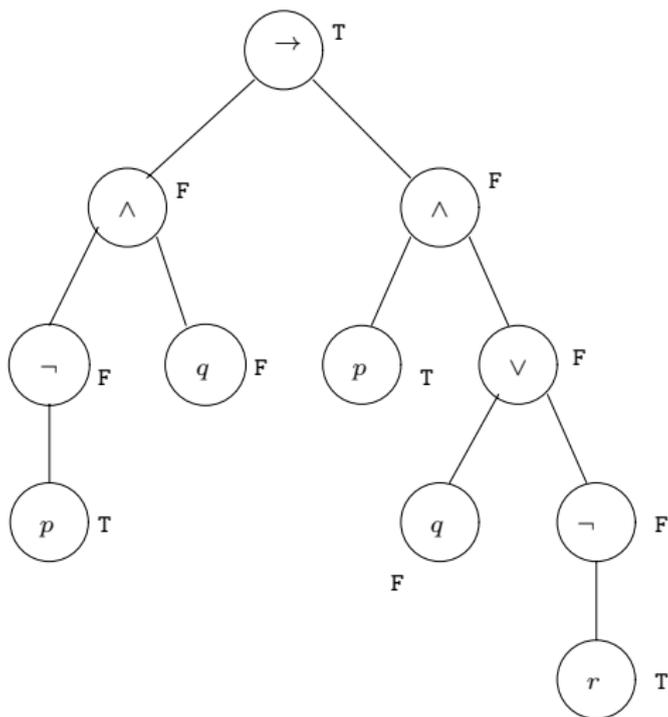


Tabla de verdad y evaluación de una expresión

- Para la expresión $(p \rightarrow \neg q) \rightarrow (q \vee \neg p)$:

p	q	$\neg p$	$\neg q$	$p \rightarrow \neg q$	$q \vee \neg p$	$(p \rightarrow \neg q) \rightarrow (q \vee \neg p)$
T	T	F	F	F	T	T
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T



El castigo de Gauss

- ▶ ¿Cual es la suma de los números del 1 al 100? El rumor dice que Gauss respondió 5050 a los pocos segundos de haber sido castigado, para sorpresa de su profesor ¿Cómo logró Gauss librarse tan fácilmente del castigo? Quizá sabía que:

$$1 + 2 + 3 + 4 + \cdots + n = \frac{n(n+1)}{2} \quad (2)$$

de forma que:

$$\begin{aligned} 1 + 2 + 3 + 4 + \cdots + 100 &= \frac{100(101)}{2} \\ &= 5050 \end{aligned}$$



La inducción matemática

- Nos permite probar que **todo** número natural satisface cierta propiedad. Supongamos que tenemos una propiedad M que creemos es verdadera para todo número natural. Supongamos ahora que sabemos lo siguiente de M :
1. **Caso base:** El número natural 1 tiene la propiedad M , es decir, tenemos una prueba de que $M(1)$.
 2. **Paso inductivo:** Si n es un número natural que asumimos tiene la propiedad $M(n)$, entonces podemos probar que $n + 1$ tiene la propiedad $M(n + 1)$; es decir, tenemos una prueba de que $M(n) \rightarrow M(n + 1)$.



Formalmente

- ▶ El principio de inducción matemática dice que, basados en estas dos piezas de información, todo número natural n tiene la propiedad $M(n)$. Al hecho de suponer $M(n)$ en el paso inductivo, se le conoce como **hipótesis de la inducción**.



Ejemplo: El castigo de Gauss

- ▶ La suma de $1 + 2 + 3 + 4 + \cdots + n$ es igual a $n(n + 1)/2$ para todo número natural n .
- ▶ Denotamos por LIE_n a $1 + 2 + 3 + 4 + \cdots + n$; y por LDE_n a $n(n + 1)/2$ para todo $n \geq 1$.
- ▶ **Caso base:** Si $n = 1$ entonces $LIE_1 = 1$ (solo hay un sumando), que es igual a $LDE_1 = 1(1 + 1)/2 = 1$.



Continuación del ejemplo: Paso inductivo

- ▶ Asumamos que $LIE_n = LDE_n$ (hipótesis inductiva).
- ▶ Necesitamos probar que $LIE_{n+1} = LDE_{n+1}$, esto es, que $1 + 2 + 3 + 4 \cdots + n + (n + 1)$ es igual que $(n + 1)((n + 1) + 1)/2$.
- ▶ De forma que...



Continuación...

$$\begin{aligned}LIE_{n+1} &= 1 + 2 + 3 + 4 + \cdots + n + (n + 1) \\&= LIE_n + (n + 1) \quad \text{reagrupando la suma} \\&= LDE_n + (n + 1) \quad \text{por la hipótesis inductiva} \\&= \frac{n(n + 1)}{2} + (n + 1) \\&= \frac{n(n + 1)}{2} + \frac{2(n + 1)}{2} \\&= \frac{(n + 2)(n + 1)}{2} \\&= \frac{(n + 1)(n + 2)}{2} \\&= \frac{(n + 1)((n + 1) + 1)}{2} \\&= LDE_{n+1}\end{aligned}$$



Final

- ▶ De forma que, como hemos probado el caso base y el paso inductivo, podemos inferir matemáticamente que todo número natural n satisface el teorema anterior. □



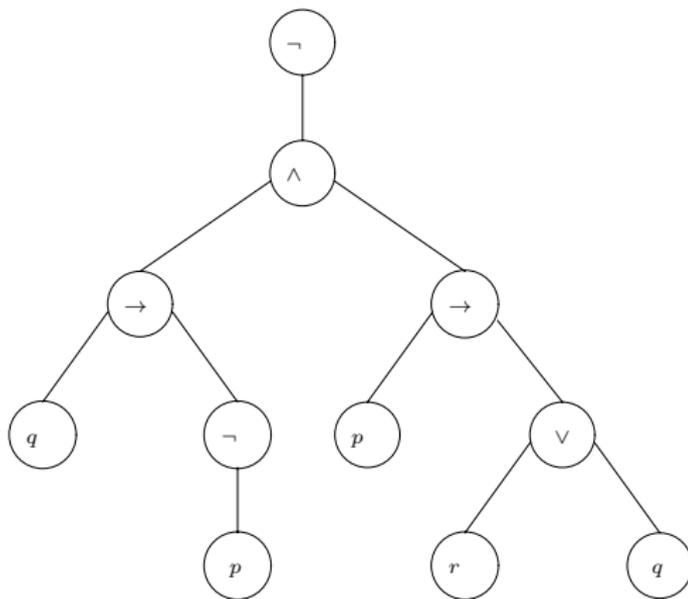
Cursos-de-valores

- ▶ Existe una variante de inducción matemática en la que la hipótesis inductiva para probar $M(n + 1)$ no es solo $M(n)$, sino la conjunción $M(1) \wedge M(2) \wedge \dots \wedge M(n)$.
- ▶ En esta variante, llamada **cursos-de-valores**, no es necesario tener un caso base explícito, todo puede hacerse en el paso inductivo.



Altura de un árbol sintáctico

- ▶ Dada una fbf ϕ , definimos su **altura** como 1 más la longitud de la rama más larga del árbol.



Inducción estructural

- ▶ Observen que la altura de un átomo proposicional es $1 + 0 = 1$.
- ▶ Puesto que toda fbf tiene una altura finita, podemos demostrar enunciados sobre las fbf haciendo inducción matemática sobre su altura.
- ▶ Este truco suele conocerse como **inducción estructural**, una importante técnica de razonamiento de Ciencias de la Computación.
- ▶ Es un caso especial de la inducción por cursos-de-valores.



Ejemplo

- ▶ Toda fbf proposicional tiene sus paréntesis **balanceados**.
- ▶ Procederemos por inducción por cursos-de-valores sobre la altura del árbol sintáctico de las fbfs ϕ .
- ▶ Denotemos por $M(n)$ que “todas las fórmulas de altura n tienen el mismo número de paréntesis que abren y cierran.” Asumimos $M(k)$ para cada $k < n$ y tratamos de probar $M(n)$. Tomemos una fórmula ϕ de altura n .



Continuación

- ▶ **Caso base:** Cuando $n = 1$, ϕ es un átomo proposicional por lo que no hay paréntesis en la expresión y $M(1)$ se satisface: $0 = 0$.
- ▶ **Paso inductivo por curso-de-valores:** Para $n > 1$ la raíz del árbol sintáctico de ϕ debe ser \neg , \rightarrow , \vee o \wedge . Supongamos que es \rightarrow , ϕ tiene la forma $(\phi_1 \rightarrow \phi_2)$. Usando la hipótesis de inducción asumimos que ϕ_1 , cuya altura es menor que n , tiene el mismo número de paréntesis que abren y cierran; lo mismo que ϕ_2 . Como en $(\phi_1 \rightarrow \phi_2)$ agregamos un paréntesis que abre y uno que cierra, ϕ está balanceada. □
- ▶ Eventualmente ϕ tendrá la forma del caso base.



Planteamiento

- ▶ Las reglas de la **deducción natural** permiten desarrollar rigurosos hilos de argumentación a través de los cuales **probamos** que ψ es el caso, asumiendo otras proposiciones como $\phi_1, \phi_2, \dots, \phi_n$.
- ▶ En ese caso decimos que el consecuente $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es **válido**.
- ▶ ¿Tenemos evidencia de que las reglas de prueba son todas **correctas** en el sentido de que preserven la verdad computada con nuestra semántica basada en tablas de verdad?
- ▶ Si ese es el caso la lógica proposicional es **robusta** (*sound*).



Consecuencia Lógica

- ▶ Si, para **todas las evaluaciones** donde $\phi_1, \phi_2, \dots, \phi_n$ son **verdaderas**, ψ también lo es, entonces decimos que:

$$\phi_1, \phi_2, \dots, \phi_n \models \psi$$

- ▶ El símbolo \models se llama relación de **consecuencia lógica** (*logical entailment*).



Ejemplos I

- ▶ ¿Es el caso que $p \wedge q \models p$? Para responder debemos inspeccionar todas las asignaciones de verdad para p y q . Cuando la asignación de valores compute T para $p \wedge q$ debemos asegurarnos de que ese también es el caso para p . Pero $p \wedge q$ solo computa T cuando p y q son verdaderas, por lo que p es consecuencia lógica de $p \wedge q$.
- ▶ ¿Qué hay acerca de $p \vee q \models p$? Hay tres asignaciones de verdad donde $p \vee q$ es T , de forma que p debería ser T en todas ellas. Sin embargo, si asignamos T a q y F a p la disyunción computa T pero p es falsa. De forma que la relación $p \vee q \models p$ no se sostiene.



Ejemplos II

- ▶ ¿Qué sucede si modificamos la inferencia anterior para que sea $\neg q, p \vee q \models p$. Observe que esto obliga a focalizar en evaluaciones donde q es falsa, lo cual obliga a que p sea verdadera si queremos que la disyunción lo sea. Por tanto, es el caso que $\neg q, p \vee q \models p$
- ▶ Observen que $p \models q \vee \neg q$ se da, aún cuando no existen ocurrencias de las proposiciones atómicas del antecedente en el consecuente.



Robustez, definición formal

- ▶ Sean $\phi_1, \phi_2, \dots, \phi_n$ y ψ fórmulas lógicas proposicionales.
- ▶ Si la inferencia $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es válida, entonces es el caso que $\phi_1, \phi_2, \dots, \phi_n \models \psi$.



Prueba

- ▶ Puesto que $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es **válida**, conocemos una **prueba** de ψ a partir de las **premisas** $\phi_1, \phi_2, \dots, \phi_n$.
- ▶ Razonaremos por inducción matemática sobre la **longitud de esta prueba**, i.e., su **número de líneas**.
- ▶ Demostrar que **para todo consecuente** $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ ($n \geq 0$) que tiene una prueba de longitud k , es el caso que $\phi_1, \phi_2, \dots, \phi_n \models \psi$.
- ▶ Denotaremos esta propiedad como $M(k)$.



Ejemplo

- ▶ La inferencia $p \wedge q \rightarrow r \vdash p \rightarrow (q \rightarrow r)$ tiene una prueba:



Prueba

- | | | |
|----|-----------------------------------|-------------------------|
| 1. | $p \wedge q \rightarrow r$ | <i>premisa</i> |
| 2. | p | <i>supuesto</i> |
| 3. | q | <i>supuesto</i> |
| 4. | $p \wedge q$ | $(\wedge i) 2, 3$ |
| 5. | r | $(\rightarrow e) 1, 4$ |
| 6. | $q \rightarrow r$ | $(\rightarrow i) 3 - 5$ |
| 7. | $p \rightarrow (q \rightarrow r)$ | $(\rightarrow i) 2 - 6$ |



Continuación...

- ▶ Si eliminamos la última línea, ya no tenemos una prueba.
- ▶ Sin embargo, podemos obtener una prueba re-escribiendo el **supuesto** de la caja más externa como una **premisa**:

1.	$p \wedge q \rightarrow r$	<i>premisa</i>
2.	p	<i>premisa</i>
3.	q	<i>supuesto</i>
4.	$p \wedge q$	$(\wedge i) 2, 3$
5.	r	$(\rightarrow e) 1, 4$
6.	$q \rightarrow r$	$(\rightarrow i) 3 - 5$



Continuación....

- ▶ Lo anterior es una prueba de que $p \wedge q \rightarrow r, p \vdash q \rightarrow r$.
- ▶ La hipótesis inductiva nos dice que entonces $p \wedge q \rightarrow r, p \models q \rightarrow r$.
- ▶ Pero entonces podemos razonar que $p \wedge q \rightarrow r \models p \rightarrow (q \rightarrow r)$



Prueba por inducción

- ▶ Asumamos que $M(k')$ para cada $k' < k$ y tratemos de probar $M(k)$.
- ▶ **Caso base:** Pruebas de longitud 1. Si $k = 1$ entonces la prueba debe ser de la forma:

1. ϕ premisa

puesto que todas las demas reglas involucran más de una linea.

- ▶ Este es el caso cuando en el consecuente $n = 1$ y ϕ_1 y ψ son iguales a ϕ , i.e., $\phi \vdash \phi$.
- ▶ Evidentemente si ϕ evalua a T , es el caso que $\phi \models \phi$.



Continuación...

- ▶ **Paso inductivo por curso-de-valores:** Asumamos que la prueba $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ tiene una longitud $k > 1$.
- ▶ Nuestra **prueba** tiene la siguiente estructura:
 1. ϕ_1 premisa
 2. ϕ_2 premisa
 - ⋮
 - n. ϕ_n premisa
 - ⋮
 - k. ψ justificación
- ▶ ¿Cual fue la **justificación** de la última línea?



Casos: Introducción de la conjunción

- ▶ Supongamos que la última regla es $(\wedge i)$, entonces ψ tiene la forma $\psi_1 \wedge \psi_2$ y la justificación de la línea k hace referencia a dos líneas precedentes que tienen a ψ_1 y ψ_2 respectivamente, como conclusiones.
- ▶ Supongamos que esas líneas son k_1 y k_2 . Dado que $k_1, k_2 < k$ existen **pruebas** de $\phi_1, \phi_2, \dots, \phi_n \vdash \psi_1$ y $\phi_1, \phi_2, \dots, \phi_n \vdash \psi_2$ con una longitud menor que k .
- ▶ Usando la **hipótesis inductiva** concluimos que $\phi_1, \phi_2, \dots, \phi_n \models \psi_1$ y $\phi_1, \phi_2, \dots, \phi_n \models \psi_2$.
- ▶ Estas dos relaciones **implican** que $\phi_1, \phi_2, \dots, \phi_n \models \psi_1 \wedge \psi_2$.



Casos: Eliminación de la disyunción

- ▶ Supongamos que la última regla es $(\vee e)$, entonces $\eta_1 \vee \eta_2$ es una **premisa** o **supuesto** en alguna línea $k' < k$, referenciada en la línea k .
- ▶ Por lo tanto, tenemos una **prueba** más corta del consecuente $\phi_1, \phi_2, \dots, \phi_n \vdash \eta_1 \vee \eta_2$, obtenido al convertir los supuestos de las cajas que se abren en la línea k' en premisas.
- ▶ De forma similar obtenemos pruebas de los consecuentes $\phi_1, \phi_2, \dots, \phi_n, \eta_1 \vdash \psi$ y $\phi_1, \phi_2, \dots, \phi_n, \eta_2 \vdash \psi$.
- ▶ Por la **hipótesis inductiva** $\phi_1, \phi_2, \dots, \phi_n \models \eta_1 \vee \eta_2$, $\phi_1, \phi_2, \dots, \phi_n, \eta_1 \models \psi$ y $\phi_1, \phi_2, \dots, \phi_n, \eta_2 \models \psi$.
- ▶ Por lo que $\phi_1, \phi_2, \dots, \phi_n \models \psi$.



Resto de los casos.

- ▶ La argumentación continua probando que todas las reglas de prueba se comportan semánticamente en la **misma forma** que las tablas de verdad correspondiente.



No existencia de prueba

- ▶ Digamos que queremos probar que $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es válida, pero no lo conseguimos ¿Cómo podemos estar seguros de que **no hay una prueba** para ese caso?
- ▶ Basta con encontrar un modelo en donde ϕ_i evalúa a T mientras que ψ evalúa a F .
- ▶ Entonces $\phi_1, \phi_2, \dots, \phi_n \not\vdash \psi$ y, usando la **robustez**, esto significa que $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ **no es válido**
- ▶ Y por tanto, **no tiene una prueba**.



Planteamiento

- ▶ En esta sección probaremos que las reglas de la deducción natural de la lógica proposicional son **completas**.
- ▶ Cuando es el caso que $\phi_1, \phi_2, \dots, \phi_n \models \psi$, entonces **existe una prueba** de deducción natural para el consecuente $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$.
- ▶ Combinando esto con el resultado anterior obtenemos que $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es válida, si y sólo si $\phi_1, \phi_2, \dots, \phi_n \models \psi$.
- ▶ Esto nos da libertad sobre qué método usar:
 - ▶ Una **búsqueda de prueba** al estilo de la programación lógica; o
 - ▶ La construcción de la **tabla de verdad**.



Pasos de la demostración

- ▶ El argumento que construiremos en esta sección se da en tres pasos, asumiendo que $\phi_1, \phi_2, \dots, \phi_n \models \psi$ es el caso:
 1. Mostraremos que $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\dots (\phi_n \rightarrow \psi)))$ es el caso.
 2. Mostraremos que $\vdash \phi_1 \rightarrow (\phi_2 \rightarrow (\dots (\phi_n \rightarrow \psi)))$ es válida.
 3. Finalmente, mostraremos que $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es válida.



Paso 1

- ▶ Una fórmula de la lógica proposicional ϕ es llamada una **tautología** si y sólo si evalúa T bajo cualquier asignación de verdad, i.e., $\models \phi$.
- ▶ Supongamos que $\phi_1, \phi_2, \dots, \phi_n \models \psi$ es el caso.
- ▶ Entonces $\phi_1 \rightarrow (\phi_2 \rightarrow (\dots (\phi_n \rightarrow \psi)))$ es una **tautología**: Solo puede evaluar F ssi todas las ϕ_i evalúan a T y ψ evalúa a F .
- ▶ Pero esto **contradice** el hecho de que $\phi_1, \phi_2, \dots, \phi_n \models \psi$; por lo tanto $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\dots (\phi_n \rightarrow \psi)))$.



Paso 2

- ▶ Si $\models \eta$, entonces $\vdash \eta$ es válida. En otras palabras, si η es una **tautología**, entonces η es un **teorema**.
- ▶ Asumamos que $\models \eta$.
- ▶ Dado que η contiene n distintos átomos proposicionales p_1, p_2, \dots, p_n sabemos que η es T para todas las 2^n **líneas** de su tabla de verdad.
- ▶ La clave está en **codificar** cada línea de la tabla de verdad de η como un **consecuente**.
- ▶ Entonces construimos pruebas para los 2^n consecuentes y las ensamblamos en la prueba de η .



Proposición

- ▶ Sea ϕ una fórmula tal que p_1, p_2, \dots, p_n son sus únicos **átomos proposicionales**.
- ▶ Sea l cualquier **número de línea** en la tabla de verdad de ϕ .
- ▶ Para todo $1 \leq i \leq n$, \hat{p}_i es p_i si la entrada de la línea l de p_i es T ; en cualquier otro caso \hat{p}_i es $\neg p_i$.
- ▶ Entonces tenemos:
 1. $\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n \vdash \phi$ es demostrable si la entrada para ϕ en la línea l es verdadera.
 2. $\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n \vdash \neg\phi$ es demostrable si la entrada para ϕ en la línea l es falsa.



Prueba: Átomos proposicionales

- ▶ Esta prueba se lleva a cabo por inducción estructural sobre la fórmula ϕ , i.e, una inducción matemática sobre la **altura del árbol sintáctico** de ϕ .
- ▶ Si ϕ es un **átomo proposicional** p , debemos mostrar que $p \vdash p$ y que $\neg p \vdash \neg p$.
- ▶ La **prueba** es trivial: de la premisa se deriva la premisa misma.



Prueba: Negación

- ▶ Si ϕ es de la forma $\neg\phi_1$, tenemos dos casos a considerar:
 1. ϕ es **verdadera**. En ese caso ϕ_1 es falsa. Observen que ϕ_1 tiene las mismas proposiciones atómicas que ϕ . Debemos usar la hipótesis de inducción sobre ϕ_1 para concluir que $\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n \vdash \neg\phi_1$, pero $\neg\phi_1$ es $\phi \dots$ Hecho ;
 2. ϕ es **falsa**. Entonces ϕ_1 es verdadera y tenemos que, por inducción $\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n \vdash \phi_1$. Usando $(\neg\neg i)$ podemos extender esta prueba en $\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n \vdash \neg\neg\phi_1$. Pero $\neg\neg\phi_1$ es $\neg\phi \dots$ Hecho.



Los demás casos

- ▶ ϕ tiene la forma $\phi_1 \circ \phi_2$, donde $\circ \in \{ \rightarrow, \wedge, \vee \}$.
- ▶ Sean $\{q_1, \dots, q_l\}$ los átomos proposicionales de ϕ_1 y $\{r_1, \dots, r_k\}$ los de ϕ_2 .
- ▶ Entonces $\{q_1, \dots, q_l\} \cup \{r_1, \dots, r_k\} = \{p_1, \dots, p_n\}$.
- ▶ Entonces, cuando $q_1, \dots, q_l \vdash \psi_1$ y $r_1, \dots, r_k \vdash \psi_2$ son **válidos**, también lo es $p_1, \dots, p_n \vdash \psi_1 \wedge \psi_2$ por $(\wedge i)$.
- ▶ Usaremos la **hipótesis inductiva** para que estas conjunciones nos permitan saber si ϕ o $\neg\phi$ son el caso.



Prueba: Implicación falsa

- ▶ Sea ϕ de la forma $\phi_1 \rightarrow \phi_2$.
- ▶ Si ϕ es **falsa**, entonces sabemos que ϕ_1 es verdadera, mientras que ϕ_2 es falsa.
- ▶ Usando nuestra **hipótesis inductiva** tenemos que $\hat{q}_1, \dots, \hat{q}_l \vdash \phi_1$ y que $\hat{r}_1, \dots, \hat{r}_k \vdash \neg\phi_2$.
- ▶ De forma que $\hat{p}_1, \dots, \hat{p}_n \vdash \phi_1 \wedge \neg\phi_2$.
- ▶ El resto consiste entonces en probar que el **consecuente** $\phi_1 \wedge \neg\phi_2 \vdash \neg(\phi_1 \rightarrow \phi_2)$



Prueba: implicación verdadera (caso 1)

- ▶ Si ϕ_1 y ϕ_2 son **falsos**.
- ▶ Por **hipótesis inductiva** tenemos que: $\hat{q}_1, \dots, \hat{q}_l \vdash \neg\phi_1$ y $\hat{r}_1, \dots, \hat{r}_k \vdash \neg\phi_2$.
- ▶ De forma que $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\phi_1 \wedge \neg\phi_2$.
- ▶ Solo queda probar que el **consecuente** $\neg\phi_1 \wedge \neg\phi_2 \vdash \phi_1 \rightarrow \phi_2$ es válido.



Implicación verdadera (casos 2 y 3)

- ▶ Si ϕ_1 es falso y ϕ_2 es verdadero usamos la hipótesis inductiva para llegar a $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\phi_1 \wedge \phi_2$ y tendríamos que probar que $\neg\phi_1 \wedge \phi_2 \vdash \phi_1 \rightarrow \phi_2$ es un consecuente válido.
- ▶ Si ϕ_1 y ϕ_2 son verdaderos, llegamos a $\hat{p}_1, \dots, \hat{p}_n \vdash \phi_1 \wedge \phi_2$ y solo resta probar que $\phi_1 \wedge \phi_2 \vdash \phi_1 \rightarrow \phi_2$.



Verificando tautologías

- ▶ Apliquemos la técnica a $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\dots (\phi_n \rightarrow \psi) \dots))$.
- ▶ Como es una **tautología**, evalúa T en las 2^n líneas de su tabla de verdad.
- ▶ Obtendremos entonces 2^n pruebas de $\hat{p}_1, \dots, \hat{p}_n \vdash \eta$.
- ▶ Hay que **ensamblar** todas estas pruebas en una sola prueba de η **sin premisas**.
- ▶ Existe una manera de hacerlo **uniformemente**.



Ejemplo

- ▶ Si queremos probar que $p \wedge q \rightarrow p$ es una tautología, tendremos que considerar que las cuatro líneas (2^2) de su tabla de verdad deberían ser verdaderas.
- ▶ Por ello tendríamos cuatro pruebas del tipo $\hat{p}_1, \hat{p}_2 \vdash \eta$:

$$\begin{array}{l}
 p, q \vdash p \wedge q \rightarrow p \\
 \neg p, q \vdash p \wedge q \rightarrow p \\
 p, \neg q \vdash p \wedge q \rightarrow p \\
 \neg p, \neg q \vdash p \wedge q \rightarrow p
 \end{array}$$



Estructura de la prueba (Gracias LEM)

1. $p \vee \neg p$ *LEM*
2. p *supuesto*
3. $q \vee \neg q$ *LEM*
4. q *supuesto*
5. \vdots \vdots
6. $p \wedge q \rightarrow p$
7. $\neg q$ *supuesto*
8. \vdots \vdots
9. $p \wedge q \rightarrow p$
10. $p \wedge q \rightarrow p$ $(\forall e)3, 4 - 6, 7 - 9$
11. $\neg p$ *supuesto*
12. $q \vee \neg q$ *LEM*
13. q *supuesto*
14. \vdots \vdots
15. $p \wedge q \rightarrow p$
16. $\neg q$ *supuesto*
17. \vdots \vdots
18. $p \wedge q \rightarrow p$
19. $p \wedge q \rightarrow p$ $(\forall e)12, 13 - 15, 16 - 18$
20. $p \wedge q \rightarrow p$ $(\forall e)1, 2 - 10, 11 - 19$



Paso 3

- ▶ Finalmente, necesitamos encontrar una prueba de que $\phi_1, \phi_2, \dots, \phi_n \vdash \psi$ es un consecuente válido.
- ▶ Tomamos la prueba de que $\vdash \phi_1 \rightarrow (\phi_2 \rightarrow (\dots (\phi_n \rightarrow \psi)))$ obtenida en el paso 2 y aumentamos la prueba introduciendo ϕ_1, \dots, ϕ_n como premisas.
- ▶ Aplicamos ($\rightarrow e$) n veces sobre cada una de las premisas y llegaremos a la conclusión que ψ lo cual nos da la prueba buscada. □



Corolario (robustez y completitud)

- ▶ Sean ϕ_1, \dots, ϕ_n y ψ fórmulas de la lógica proposicional.
- ▶ $\phi_1, \dots, \phi_n \models \psi$ es el caso, si y sólo si el consecuente $\phi_1, \dots, \phi_n \vdash \psi$ es válido.



De lo sintáctico a lo semántico

- ▶ La robustez implica que cualquier cosa que demostremos es un hecho verdadero.
- ▶ La completitud implica que todo lo verdadero tiene una prueba en el sistema de deducción natural.
- ▶ Esta conexión nos permite usar indistintamente las nociones de prueba (\vdash) y consecuencia lógica (\models).
- ▶ La deducción natural es solo **una** forma de demostración de muchas posibles.
- ▶ Exploraremos otras explotando la **equivalencia semántica**.



Equivalencia semántica

- ▶ Sean ϕ y ψ dos fórmulas de la lógica proposicional.
- ▶ Decimos que son **semánticamente equivalentes** ssi $\phi \models \psi$ y $\psi \models \phi$.
- ▶ En ese caso, solemos escribir $\phi \equiv \psi$.
- ▶ Decimos que ϕ es **válida** si $\models \phi$.
- ▶ También pudimos definir $\phi \equiv \psi$ para denotar que $\models (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$.
- ▶ Dadas la robustez y la completitud, la equivalencia semántica y de prueba son **idénticas**.



Ejemplos

$$\blacktriangleright p \rightarrow q \equiv \neg q \rightarrow \neg p$$

$$\blacktriangleright p \rightarrow q \equiv \neg p \vee q$$

$$\blacktriangleright p \wedge q \rightarrow p \equiv r \vee \neg r$$

$$\blacktriangleright p \wedge q \rightarrow r \equiv p \rightarrow (q \rightarrow r)$$



Lemma

- ▶ Dadas la fórmulas $\phi_1, \phi_2, \dots, \phi_n$ y ψ de la lógica proposicional, $\phi_1, \phi_2, \dots, \phi_n \models \psi$ ssi $\models \phi_1 \rightarrow (\phi_2 \rightarrow \dots \rightarrow (\phi_n \rightarrow \psi))$.
- ▶ **Prueba:** Suponemos que $\models \phi_1 \rightarrow (\phi_2 \rightarrow \dots \rightarrow (\phi_n \rightarrow \psi))$ es el caso. Si las ϕ_i son verdaderas bajo una valuación, también lo debe ser ψ . A la inversa es el paso 1 de la prueba de completitud. \square



CNF

- ▶ Una **literal** L es un átomo p o su negación (3). Una fórmula C está en **Formal Normal Conjuntiva** (CNF) si es una conjunción de cláusulas (5), donde cada cláusula es una disyunción de literales (4).

$$L ::= p \mid \neg p \quad (3)$$

$$D ::= L \mid L \vee D \quad (4)$$

$$C ::= D \mid D \wedge C \quad (5)$$



Ejemplos

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

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



Relevancia de la CNF

- ▶ Permite verificar validez **facilmente**, proceso que de otra forma es exponencial en el número de átomos de la fbf a verificar.
- ▶ **Lema.** Una disyunción de literales $L_1 \vee \dots \vee L_m$ es **válida** ssi existe un $L_i = \neg L_j$ para $1 \leq i, j \leq m$.
- ▶ **Ej.** $p \vee q \vee r \vee \neg q$ no puede ser falso.
- ▶ ¿El costo? Convertir una fbf proposicional a su CNF.



Satisfacción

- ▶ Sea ϕ una fbf proposicional, ϕ es **satisfacible** ssi $\neg\phi$ no es válida.
- ▶ **Prueba:** Asumimos que ϕ es satisfacible. Por definición, existe una valuación donde ϕ es verdadera, pero eso implica que $\neg\phi$ sería falsa en la misma valuación, por lo que $\neg\phi$ no puede ser válida; Asumimos que $\neg\phi$ no es válida, debe haber una valuación donde $\neg\phi$ sea falsa, en cuyo caso ϕ es verdadera y por tanto satisfacible. \square
- ▶ **Resultado:** Solo necesitamos un procedimiento de decisión para ambos conceptos!



Ideas

- ▶ Comenzaremos por un algoritmo **determinista** que siempre computa la misma CNF para una fbf ϕ de entrada.
- ▶ Características:
 1. CNF termina para toda fbf de la lógica proposicional.
 2. Para cada fbf de la lógica proposicional CNF computa una fbf equivalente.
 3. La fbf resultante esta en forma normal conjuntiva.



Estrategia

- ▶ Proceder por inducción estructural sobre la fbf ϕ .
- ▶ **Ejemplo:** Si ϕ es de la forma $\phi_1 \wedge \phi_2$ computar la CNF η_1 para ϕ_i , $i = 1, 2$; de forma que $\eta_1 \wedge \eta_2$ es la CNF equivalente a ϕ .
- ▶ La inducción estructural garantiza las características deseables del algoritmo.



Preprocesamiento

- free_impl.** Convierte las implicaciones en disyunciones, usando $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$.
- nnf.** Convierte las fbf en su equivalente bajo negación en forma normal (solo los átomos están negados). Se usan las leyes de Morgan.
- cnf.** Computa la forma conjuntiva normal de $nnf(impl_free(\phi))$.



CNF

► Se resuelve por casos:

1. Si ϕ es una literal, por definición está en CNF y la salida es ϕ .
2. Si ϕ es de la forma $\phi_1 \wedge \phi_2$ se llama a CNF recursivamente sobre cada ϕ_i para obtener η_1 y η_2 . La CNF es $\eta_1 \wedge \eta_2$.
3. Si ϕ tiene la forma $\phi_1 \vee \phi_2$ se llama a CNF sobre cada ϕ_i , pero no regresamos $\eta_1 \vee \eta_2$ puesto que esta solo es una forma normal si η_i son literales.
4. Se debe distribuir la disyunción sobre la conjunción, garantizando que la disyunciones generadas sean de literales. Una función $distr(\eta_1, \eta_2)$ haría ese trabajo.



CNF

```
1: function CNF( $\phi$ )
2:   switch  $\phi$  do
3:     case literal( $\phi$ )
4:       return  $\phi$ 
5:     case  $\phi_1 \wedge \phi_2$ 
6:       return CNF( $\phi_1$ )  $\wedge$  CNF( $\phi_2$ )
7:     case  $\phi_1 \vee \phi_2$ 
8:       return DISTR(CNF( $\phi_1$ ), CNF( $\phi_2$ ))
9:   end function
```

▷ ϕ es una fbf sin implicación en NNF



DISTR

```

1: function DISTR( $\eta_1, \eta_2$ )
2:   if  $\eta_1 = \eta_{11} \wedge \eta_{12}$  then
3:     return DISTR( $\eta_{11}, \eta_2$ )  $\wedge$  DISTR( $\eta_{12}, \eta_2$ )
4:   else if  $\eta_2 = \eta_{21} \wedge \eta_{22}$  then
5:     return DISTR( $\eta_1, \eta_{21}$ )  $\wedge$  DISTR( $\eta_1, \eta_{22}$ )
6:   else
7:     return  $\eta_1 \vee \eta_2$ 
8:   end if
9: end function

```

▷ η_1 y η_2 están en CNF

▷ No hay conjunciones



NNF

```

1: function NNF( $\phi$ )
2:   switch  $\phi$  do
3:     case literal( $\phi$ )
4:       return  $\phi$ 
5:     case  $\neg\neg\phi_1$ 
6:       return  $\phi_1$ 
7:     case  $\phi_1 \wedge \phi_2$ 
8:       return  $NNF(\phi_1) \wedge NNF(\phi_2)$ 
9:     case  $\phi_1 \vee \phi_2$ 
10:      return  $NNF(\phi_1) \vee NNF(\phi_2)$ 
11:    case  $\neg(\phi_1 \wedge \phi_2)$ 
12:      return  $NNF(\neg\phi_1) \vee NNF(\neg\phi_2)$ 
13:    case  $\neg(\phi_1 \vee \phi_2)$ 
14:      return  $NNF(\neg\phi_1) \wedge NNF(\neg\phi_2)$ 
15:  end function

```

▷ ϕ es libre de implicaciones



Ejemplo

► $\phi = (\neg p \wedge q \rightarrow p \wedge (r \rightarrow q))$



IMPL_FREE

$$\begin{aligned}
 \text{IMPL_FREE}(\phi) &= \neg \text{IMPL_FREE}(\neg p \wedge q) \vee \text{IMPL_FREE}(p \wedge (r \rightarrow q)) \\
 &= \neg((\text{IMPL_FREE} \neg p) \wedge (\text{IMPL_FREE} q)) \vee \text{IMPL_FREE}(p \wedge (r \rightarrow q)) \\
 &= \neg((\neg p) \wedge \text{IMPL_FREE} q) \vee \text{IMPL_FREE}(p \wedge (r \rightarrow q)) \\
 &= \neg(\neg p \wedge q) \vee \text{IMPL_FREE}(p \wedge (r \rightarrow q)) \\
 &= \neg(\neg p \wedge q) \vee ((\text{IMPL_FREE} p) \wedge \text{IMPL_FREE}(r \rightarrow q)) \\
 &= \neg(\neg p \wedge q) \vee (p \wedge \text{IMPL_FREE}(r \rightarrow q)) \\
 &= \neg(\neg p \wedge q) \vee (p \wedge (\neg(\text{IMPL_FREE} r) \vee (\text{IMPL_FREE} q))) \\
 &= \neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee (\text{IMPL_FREE} q))) \\
 &= \neg(\neg p \wedge q) \vee (p \wedge (\neg r \vee q)).
 \end{aligned}$$



NNF

$$\begin{aligned}
\text{NNF}(\text{IMPL_FREE } \phi) &= \text{NNF}(\neg(\neg p \wedge q)) \vee \text{NNF}(p \wedge (\neg r \vee q)) \\
&= \text{NNF}(\neg(\neg p) \vee \neg q) \vee \text{NNF}(p \wedge (\neg r \vee q)) \\
&= (\text{NNF}(\neg\neg p)) \vee (\text{NNF}(\neg q)) \vee \text{NNF}(p \wedge (\neg r \vee q)) \\
&= (p \vee (\text{NNF}(\neg q))) \vee \text{NNF}(p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee \text{NNF}(p \wedge (\neg r \vee q)) \\
&= (p \vee \neg q) \vee ((\text{NNF } p) \wedge (\text{NNF}(\neg r \vee q))) \\
&= (p \vee \neg q) \vee (p \wedge (\text{NNF}(\neg r \vee q))) \\
&= (p \vee \neg q) \vee (p \wedge ((\text{NNF}(\neg r)) \vee (\text{NNF } q))) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee (\text{NNF } q))) \\
&= (p \vee \neg q) \vee (p \wedge (\neg r \vee q)).
\end{aligned}$$



Corrida en Prolog

```

1  ?- impl_free(~p ^ q => p ^ (r => q), IMPLFREE).
2  IMPLFREE = (~ (~p^q)v p^(~r v q))
3
4  ?- impl_free(~p ^ q => p ^ (r => q), IMPLFREE), nnf(IMPLFREE,NNF).
5  IMPLFREE = (~ (~p^q)v p^(~r v q)),
6  NNF = ((p v ~q)v p^(~r v q))
7
8  ?- cnf(~p ^ q => p ^ (r => q), CNF).
9  CNF = ((p v ~q)v p)^((p v ~q)v~r v q)
10
11 ?- cnf(r => (s => (t ^ s => r)), CNF).
12 CNF = (~r v ~s v (~t v ~s)v r)

```



SAT (Davis, Logemann y Loveland [1])

```

1: function DPLL( $f, \theta$ )
2:    $\theta_1 \leftarrow \theta \cup \text{unit-propagation}(f, \theta)$ 
3:   if is-satisfied( $f, \theta_1$ ) then
4:     return  $\theta_1$ 
5:   else if is-conflicting( $f, \theta_1$ ) then
6:     return  $\perp$ 
7:   else
8:      $x \leftarrow \text{choose-free-variable}(f, \theta_1)$ 
9:      $\theta_2 \leftarrow \text{DPLL}(f, \theta_1 \cup \{x \mapsto \text{true}\})$ 
10:    if  $\theta_2 \neq \perp$  then
11:      return  $\theta_2$ 
12:    else
13:      return  $\text{DPLL}(f, \theta_1 \cup \{x \mapsto \text{false}\})$ 
14:    end if
15:  end if
16: end function

```

▷ f es una fbf en CNF, θ es una asignación de verdad



Prolog implementation (Howe y King [2]) I

```

1  sat(Clauses, Vars) :-
2      problem_setup(Clauses), elim_var(Vars).
3
4  elim_var([]).
5  elim_var([Var | Vars]) :-
6      elim_var(Vars), assign(Var).
7
8  assign(true).
9  assign(false).
10
11 problem_setup([]).
12 problem_setup([Clause | Clauses]) :-
13     clause_setup(Clause),
14     problem_setup(Clauses).
15
16 clause_setup([Pol-Var | Pairs]) :-
17     set_watch(Pairs, Var, Pol).
18
19 set_watch([], Var, Pol) :-
20     Var = Pol.
21 set_watch([Pol2-Var2 | Pairs], Var1, Pol1) :-
22     when((nonvar(Var1); nonvar(Var2)), watch(Var1, Pol1, Var2, Pol2, Pairs)).
23
24 watch(Var1, Pol1, Var2, Pol2, Pairs) :-
25     nonvar(Var1) ->
26     update_watch(Var1, Pol1, Var2, Pol2, Pairs);
27     update_watch(Var2, Pol2, Var1, Pol1, Pairs).
28

```



Prolog implementation (Howe y King [2]) II

```
29 | update_watch(Var1, Pol1, Var2, Pol2, Pairs) :-  
30 |     Var1 == Pol1 -> true; set_watch(Pairs, Var2, Pol2).
```



Ejemplo

- ▶ Probar que $\neg x \vee (y \wedge \neg z)$ es satisfacible.
- ▶ En CNF: $(\neg x \vee y) \wedge (\neg x \vee \neg z)$.
- ▶ Representación para el algoritmo:

```

1  |- Clauses = [[false-X, true-Y],[false-X, false-Z]], sat(Clauses,[X,Y,Z]).
2  Clauses = [[false-false, true-true], [false-false, false-true]],
3  X = false,
4  Y = Z, Z = true ;
5  Clauses = [[false-false, true-false], [false-false, false-true]],
6  X = Y, Y = false,
7  Z = true ;
8  Clauses = [[false-true, true-true], [false-true, false-false]],
9  X = Y, Y = true,
10 Z = false ;
11 Clauses = [[false-false, true-true], [false-false, false-false]],
12 X = Z, Z = false,
13 Y = true ;
14 Clauses = [[false-false, true-false], [false-false, false-false]],
15 X = Y, Y = Z, Z = false.

```



Referencias I



M Davis, D Logemann y D Loveland. "A machine program for theorem proving". En: *Communications of the ACM* 5.7 (1962), págs. 394-397.



JM Howe y A King. "A pearl in SAT and SMT solving in Prolog". En: *Theoretical Computer Science* 435.2012 (2012), págs. 43-55.



M Huth y M Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge, UK: Cambridge University Press, 2004.

