Matemática Discreta

Clase 3: Expresividad de conectivos de la lógica proposicional y deducción natural

---

Andrés Abeliuk*

Departamento de Ciencias de la Computación
Universidad de Chile

* Estas diapositivas fueron diseñadas a partir de diapositivas del profesor Alejandro Hevia y Federico Olmedo.

Contenido clase de hoy

Lógica de predicados y métodos de demostración

1. Complejidad de consecuencia lógica
2. Expresividad de los conectivos
3. Reglas de deducción natural

Complejidad de consecuencia lógica

Consecuencia lógica

La noción de consecuencia es fundamental para cualquier lenguaje, y automatizar un proceso de razonamiento deductivo es de particular interés en IA.

Consecuencia lógica

Sea Γ un conjunto de fórmulas (premisas) en $\mathcal{L}(P)$ y α una fórmula (conclusión) en $\mathcal{L}(P)$. Decimos que α es consecuencia lógica de Γ, notado $\Gamma \models \alpha$, sii para cada valuación $\sigma\colon P \to \{0,1\}$,

$$\text{si } \hat{\sigma}(\Gamma) = 1, \text{entonces } \hat{\sigma}(\alpha) = 1$$

$\hat{\sigma}(\gamma) = 1$ para cada $\gamma \in \Gamma$

Consecuencia lógica

Como verificamos que la validez de un argumento en lógica proposicional?

Un algoritmo de fuerza bruta para validar si $\Gamma \models \alpha$:

- Construir la tabla de verdad para Γ y α
- Comprobar si para toda asignación σ tal que $\hat{\sigma}(\Gamma) = 1$ se tiene que $\hat{\sigma}(\alpha) = 1$.

Consecuencia lógica

Como verificamos que la validez de un argumento en lógica proposicional?

Un algoritmo de fuerza bruta para validar si $\Gamma \models \alpha$:

- Construir la tabla de verdad para Γ y α
- Comprobar si para toda asignación σ tal que $\hat{\sigma}(\Gamma) = 1$ se tiene que $\hat{\sigma}(\alpha) = 1$.

Pregunta: ¿Cual es la eficiencia computacional de este algoritmo?

Consecuencia lógica

Como verificamos que la validez de un argumento en lógica proposicional?

Un algoritmo de fuerza bruta para validar si $\Gamma \models \alpha$:

- Construir la tabla de verdad para Γ y α
- Comprobar si para toda asignación σ tal que $\hat{\sigma}(\Gamma) = 1$ se tiene que $\hat{\sigma}(\alpha) = 1$.

Pregunta: ¿Cual es la eficiencia computacional de este algoritmo?

Tiempo exponencial en el numero de proposiciones.

Complejidad de consecuencia lógica

Tenemos que :

$$\Gamma \models \alpha \text{ si y sólo si } \Gamma \cup \{\neg\alpha\} \text{ es insatisfactible.}$$

Por tanto, la complejidad del problema de consecuencia lógica es al menos tan difícil como el problema de (in)satisfacibilidad

Complejidad de consecuencia lógica

Tenemos que :

$$\Gamma \models \alpha \text{ si y sólo si } \Gamma \cup \{\neg\alpha\} \text{ es insatisfactible.}$$

Por tanto, la complejidad del problema de consecuencia lógica es al menos tan difícil como el problema de (in)satisfacibilidad

Teorema de Cook (1971)

El Problema de satisfacibilidad booleana (SAT) es NP-completo.

Es muy poco probable que exista un algoritmo eficiente que resuelva el problema de consecuencia lógica.

Podemos empezar pensando en restricciones en la forma de las fórmulas.

Expresividad de los conectivos

Expresividad de la lógica proposicional

Dada una función Booleana $\phi\colon \{0,1\}^n \to \{0,1\}$, ¿siempre puede representarse a través de una fórmula de la lógica proposicional?

Expresividad de la lógica proposicional

Dada una función Booleana $\phi\colon \{0,1\}^n \to \{0,1\}$, ¿siempre puede representarse a través de una fórmula de la lógica proposicional?

Ejemplo:

p_1	p_2	p_3	$\phi(p_1, p_2, p_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Expresividad de la lógica proposicional

Dada una función Booleana $\phi\colon \{0,1\}^n \to \{0,1\}$, ¿siempre puede representarse a través de una fórmula de la lógica proposicional?

Ejemplo:

p_1	p_2	p_3	$\phi(p_1, p_2, p_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Concentrándonos en las 3 valuaciones que hacen ϕ verdadera, vemos que $\phi(p_1, p_2, p_3)$ puede expresarse como

$$(\neg p_1 \wedge \neg p_2 \wedge p_3) \vee (p_1 \wedge \neg p_2 \wedge \neg p3) \vee (p_1 \wedge p_2 \wedge \neg p3)$$

Expresividad de la lógica proposicional

Toda tabla de verdad puede expresarse a través de una fórmula de la lógica proposicional.

Expresividad de la lógica proposicional

Toda tabla de verdad puede expresarse a través de una fórmula de la lógica proposicional.

Caso general:

valuación	p_1	$\ldots$	p_n	valor de verdad
σ_1	0	$\ldots$	0	b_1
σ_2	0	$\ldots$	1	b_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
σ_{2^n}	1	$\ldots$	1	b_{2^n}

La tabla de verdad puede describirse por la siguiente fórmula de la lógica proposicional[1]:

$$\bigvee_{i\,:\,b_i=1} \left(\left(\bigwedge_{j\,:\,\sigma_i(p_j)=1} p_j \right) \wedge \left(\bigwedge_{k\,:\,\sigma_i(p_k)=0} \neg p_k \right) \right)$$

Expresividad de la lógica proposicional

Toda tabla de verdad puede expresarse a través de una fórmula de la lógica proposicional.

Caso general:

valuación	p_1	$\ldots$	p_n	valor de verdad
σ_1	0	$\ldots$	0	b_1
σ_2	0	$\ldots$	1	b_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
σ_{2^n}	1	$\ldots$	1	b_{2^n}

La tabla de verdad puede describirse por la siguiente fórmula de la lógica proposicional[1]:

$$\bigvee_{i\,:\,b_i=1} \left(\left(\bigwedge_{j\,:\,\sigma_i(p_j)=1} p_j \right) \wedge \left(\bigwedge_{k\,:\,\sigma_i(p_k)=0} \neg p_k \right) \right)$$

[1] Si la tabla representa una contradicción, la fórmula de arriba es "vacía", pero la tabla puede igualmente representarse como $p_1 \wedge \neg p_1$.

Completitud de conectivos lógicos

Del resultado anterior se desprende el siguiente corolario

Teorema (completitud de conectivos lógicos)

Los conectivos $\neg$, $\wedge$ y $\vee$ son suficientes para expresar todas las fórmulas de la lógica proposicional. Formalmente decimos que el conjunto de conectivos $\{\neg, \wedge, \vee\}$ es funcionalmente completo.

Completitud de conectivos lógicos

Del resultado anterior se desprende el siguiente corolario

Teorema (completitud de conectivos lógicos)

Los conectivos $\neg$, $\wedge$ y $\vee$ son suficientes para expresar todas las fórmulas de la lógica proposicional. Formalmente decimos que el conjunto de conectivos $\{\neg, \wedge, \vee\}$ es funcionalmente completo.

Ejercicio: Demostrar que los conjuntos de conectivos $\{\neg, \wedge\}$ y $\{\neg, \vee\}$ también son funcionalmente completos.

Completitud de conectivos lógicos

Del resultado anterior se desprende el siguiente corolario

Teorema (completitud de conectivos lógicos)

Los conectivos $\neg$, $\wedge$ y $\vee$ son suficientes para expresar todas las fórmulas de la lógica proposicional. Formalmente decimos que el conjunto de conectivos $\{\neg, \wedge, \vee\}$ es funcionalmente completo.

Ejercicio: Demostrar que los conjuntos de conectivos $\{\neg, \wedge\}$ y $\{\neg, \vee\}$ también son funcionalmente completos.

Hint: Pensar cómo se puede reescribir $\alpha \vee \beta$ (resp. $\alpha \wedge \beta$) en términos de $\neg$ y $\wedge$ (resp. $\neg$ y $\vee$) usando las Leyes de De Morgan.

Formas normales: FND

Comenzamos con unas definiciones preliminares:

Formas normales: FND

Comenzamos con unas definiciones preliminares:

- Un literal es una proposición o su negación.

Formas normales: FND

Comenzamos con unas definiciones preliminares:

- Un literal es una proposición o su negación.
- Una fórmula está en forma normal disyuntiva sii es una *disyunción de conjunciones de literales*, por ejemplo

$$(p_1 \wedge \neg p_2) \vee (p_2 \wedge p_4) \vee (p_1 \wedge p_3 \wedge \neg p_4)$$

.

Formas normales: FND

Comenzamos con unas definiciones preliminares:

- Un literal es una proposición o su negación.
- Una fórmula está en forma normal disyuntiva sii es una *disyunción de conjunciones de literales*, por ejemplo

$$(p_1 \wedge \neg p_2) \vee (p_2 \wedge p_4) \vee (p_1 \wedge p_3 \wedge \neg p_4)$$

.

Teorema (forma normal disyuntiva)

Toda fórmula de la lógica proposicional es equivalente a una fórmula en forma normal disyuntiva.

Formas normales: FND

Comenzamos con unas definiciones preliminares:

- Un literal es una proposición o su negación.
- Una fórmula está en forma normal disyuntiva sii es una *disyunción de conjunciones de literales*, por ejemplo

$$(p_1 \wedge \neg p_2) \vee (p_2 \wedge p_4) \vee (p_1 \wedge p_3 \wedge \neg p_4)$$

.

Teorema (forma normal disyuntiva)

Toda fórmula de la lógica proposicional es equivalente a una fórmula en forma normal disyuntiva.

Demostración

Ya la hicimos 2 diapositivas atrás.

Formas normales: FND

Observación: El problema de satisfacibilidad para fórmulas en FND puede ser resuelto eficientemente.

Formas normales: FND

Observación: El problema de satisfacibilidad para fórmulas en FND puede ser resuelto eficientemente.

Por tanto, no existe algoritmo eficiente que toma una fórmula y la convierte en una fórmula en FND equivalente.

Formas normales: FNC

Una fórmula está en forma normal conjuntiva sii es una *conjunción de disyunciones de literales*, por ejemplo

$$(p_1 \vee \neg p_2) \wedge (p_2 \vee p_4) \wedge (p_3 \vee \neg p_3)$$

.

Formas normales: FNC

Una fórmula está en forma normal conjuntiva sii es una *conjunción de disyunciones de literales*, por ejemplo

$$(p_1 \vee \neg p_2) \wedge (p_2 \vee p_4) \wedge (p_3 \vee \neg p_3)$$

.

Teorema (forma normal conyuntiva)

Toda fórmula de la lógica proposicional es equivalente a una fórmula en forma normal conjuntiva.

Formas normales: FNC

Una fórmula está en forma normal conjuntiva sii es una *conjunción de disyunciones de literales*, por ejemplo

$$(p_1 \vee \neg p_2) \wedge (p_2 \vee p_4) \wedge (p_3 \vee \neg p_3)$$

.

Teorema (forma normal conyuntiva)

Toda fórmula de la lógica proposicional es equivalente a una fórmula en forma normal conjuntiva.

Demostración

Pensar cómo se puede expresar una tabla de verdad en forma normal conjuntiva.

Formas normales: FNC

Ejemplo:

p_1	p_2	p_3	$\phi(p_1, p_2, p_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Formas normales: FNC

Ejemplo:

p_1	p_2	p_3	$\phi(p_1, p_2, p_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Concentrándonos en las 3 valuaciones que hacen ϕ falsa, vemos que $\phi(p_1, p_2, p_3)$ puede expresarse como

$$(p_1 \wedge p_2 \wedge \neg p_3) \vee (\neg p_1 \wedge p_2 \wedge p3) \vee (\neg p_1 \wedge \neg p_2 \wedge p3)$$

Formas normales: FNC

valuación	p_1	$\ldots$	p_n	valor de verdad
σ_1	0	$\ldots$	0	b_1
σ_2	0	$\ldots$	1	b_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
σ_{2^n}	1	$\ldots$	1	b_{2^n}

La tabla de verdad puede describirse por la siguiente fórmula de la lógica proposicional:

$$\bigwedge_{i\,:\,b_i=0} \left(\left(\bigvee_{j\,:\,\sigma_i(p_j)=0} p_j \right) \vee \left(\bigvee_{k\,:\,\sigma_i(p_k)=1} \neg p_k \right) \right)$$

Formas normales: FNC

valuación	p_1	$\ldots$	p_n	valor de verdad
σ_1	0	$\ldots$	0	b_1
σ_2	0	$\ldots$	1	b_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
σ_{2^n}	1	$\ldots$	1	b_{2^n}

La tabla de verdad puede describirse por la siguiente fórmula de la lógica proposicional:

$$\bigwedge_{i\,:\,b_i=0} \left(\left(\bigvee_{j\,:\,\sigma_i(p_j)=0} p_j \right) \vee \left(\bigvee_{k\,:\,\sigma_i(p_k)=1} \neg p_k \right) \right)$$

Teorema (forma normal conyuntiva)

El problema de satisfacibilidad para FNC es NP-completo (Cook'71)

Ejercicio: Cómo transformar una fórmula en FND a una fórmula equivalente en FNC.

Reglas de deducción natural

Deducción natural

Existen algoritmos de razonamiento deductivo que verifican si un conjunto de fórmula pero no son eficientes en el peor de los casos.

Deducción natural

Existen algoritmos de razonamiento deductivo que verifican si un conjunto de fórmula pero no son eficientes en el peor de los casos.

- Contamos con una colección de reglas de inferencia que nos permiten establecer que fórmulas se pueden derivar a partir de otras.
- Al aplicar las reglas a las fórmulas en las premisas, esperamos obtener nuevas fórmulas hasta obtener eventualmente la conclusión.
- Construir estas pruebas es un ejercicio creativo, semejante a programar. No es necesariamente obvio que regla aplicar, ni en qué orden.

Deducción natural

Existen algoritmos de razonamiento deductivo que verifican si un conjunto de fórmula pero no son eficientes en el peor de los casos.

- Contamos con una colección de reglas de inferencia que nos permiten establecer que fórmulas se pueden derivar a partir de otras.
- Al aplicar las reglas a las fórmulas en las premisas, esperamos obtener nuevas fórmulas hasta obtener eventualmente la conclusión.
- Construir estas pruebas es un ejercicio creativo, semejante a programar. No es necesariamente obvio que regla aplicar, ni en qué orden.

A continuación revisaremos en detalle las reglas usadas en la deducción natural

Regla de la conjunción

Esta regla permite concluir $\phi \wedge \psi$ dado que ϕ y ψ son el caso.

Introducción de la conjunción

$$\frac{\phi \qquad \psi}{\phi \wedge \psi}\,[\wedge i]$$

Eliminación de la conjunción

$$\frac{\phi \wedge \psi}{\phi}\,[\wedge e_1] \qquad \frac{\phi \wedge \psi}{\psi}\,[\wedge e_2]$$

Sobre las líneas están las dos premisas de la regla. Abajo de la línea se encuentra la conclusión. A la derecha de la regla tenemos el nombre de la misma abreviado

Ejemplo de demostración

Queremos probar:

$$p \wedge q, r \vdash q \wedge r$$

Ejemplo de demostración

Queremos probar:

$$p \wedge q, r \vdash q \wedge r$$

Representación grafica de la demostración:

$$\dfrac{\dfrac{p \wedge q}{q}[\wedge e_2] \qquad r}{q \wedge r}[\wedge i]$$

Regla de doble negación

Una fórmula es equivalente a su doble negación.

$$\dfrac{\neg\neg\phi}{\phi}[\neg\neg e] \qquad \dfrac{\phi}{\neg\neg\phi}[\neg\neg i]$$

Regla de eliminación de la implicación

El "Modus Ponens" (modo que afirma en latín) se introduce con la siguiente regla, conocida como eliminación de la implicación.

$$\frac{\phi \qquad \phi \to \psi}{\psi}[\to e]$$

Regla de eliminación de la implicación

El "Modus Ponens" (modo que afirma en latín) se introduce con la siguiente regla, conocida como eliminación de la implicación.

$$\frac{\phi \qquad \phi \to \psi}{\psi}[\to e]$$

Una regla similar

$$\frac{\phi \to \psi \qquad \neg\psi}{\neg\phi}[MT]$$

es llamada "Modus Tollens"' (MT), que significa modo que niega negando.

Regla de introducción de la implicación

Queremos demostrar

$$p \to q \vdash \neg q \to \neg p$$

La mecánica de la regla es más complicada que las anteriores.

1	$p \to q$	premise
2	$\neg q$	assumption
3	$\neg p$	MT 1,2
4	$\neg q \to \neg p$	$\to_i$ 2–3

Regla de introducción de la implicación

Queremos demostrar

$$p \to q \vdash \neg q \to \neg p$$

La mecánica de la regla es más complicada que las anteriores.

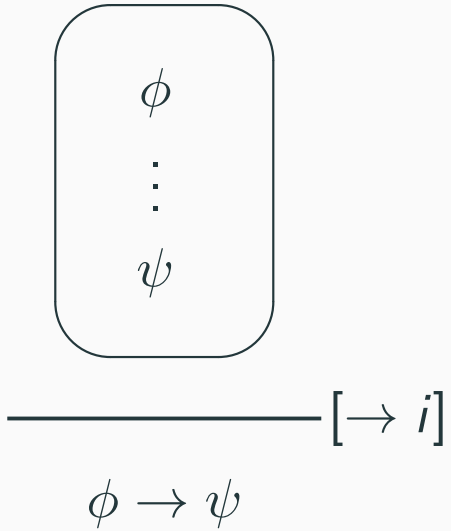
1	$p \to q$	premise
2	$\neg q$	assumption
3	$\neg p$	MT 1,2
4	$\neg q \to \neg p$	$\to_i$ 2–3

El cuadro sirve para delimitar el alcance del supuesto temporal $\neg q$

No podemos usar suposiciones dentro del cuadro en reglas fuera del cuadro.

Regla de introducción de la implicación

Esta es una forma de razonamiento hipotético.



La regla expresa que para probar $\phi \rightarrow \psi$ debemos asumir temporalmente ϕ y entonces probar ψ.

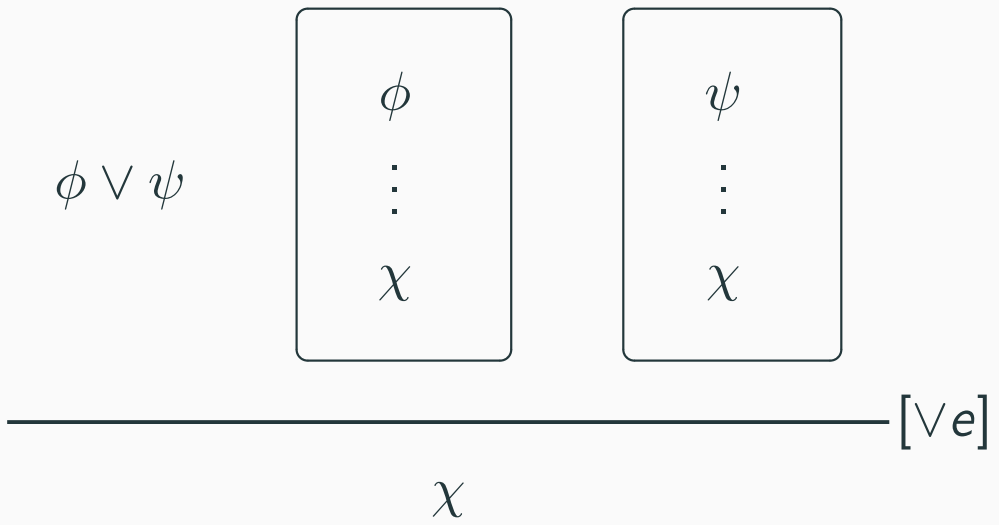
En la prueba de ψ se pueden usar cualquiera de las fórmulas obtenidas hasta el momento, como premisas y conclusiones provisionales.

Reglas de la disyunción

$$\frac{\phi}{\phi \vee \psi}[\vee i_i] \qquad \frac{\psi}{\phi \vee \psi}[\vee i_2]$$

Reglas de eliminación por la disyunción

Supongamos que queremos probar una proposición χ asumiendo que $\phi \vee \psi$. Puesto que no sabemos que argumento de la disyunción es verdadero, debemos obtener dos pruebas por separado que deberemos combinar en un solo argumento:



Reglas de la negación

Las contradicciones son expresiones de la forma $\phi \wedge \neg\phi$, donde ϕ es cualquier fórmula. Una contradicción se denota $\bot$.

$$\frac{\phi \qquad \neg\phi}{\bot}[\neg e] \qquad \frac{\bot}{\phi}[\bot e] \qquad \frac{\begin{array}{c}\phi \\ \vdots \\ \bot\end{array}}{\neg\phi}[\neg i]$$

Asumimos temporalmente ϕ, y demostramos que conduce a una contradicción, por lo que concluimos que $\neg\phi$ se cumple.

Ejemplo

Demostración de $p \wedge (q \vee r) \vdash (p \wedge q) \vee (p \wedge r)$

1	$p \wedge (q \vee r)$	premisa
2	p	$\wedge e_1$ 1
3	$q \vee r$	$\wedge e_2$ 1
4	q	supuesto
5	$p \wedge q$	$\wedge i$ 2,4
6	$(p \wedge q) \vee (p \wedge r)$	$\vee i_1$ 5
7	r	supuesto
8	$p \wedge r$	$\wedge i$ 2,7
9	$(p \wedge q) \vee (p \wedge r)$	$\vee i_2$ 8
10	$(p \wedge q) \vee (p \wedge r)$	$\vee e$ 3, 4–6, 7–9

Conclusión

Tenemos un cálculo para demostrar basado en reglas

- ¿Es consistente? (Sólo prueba cosas verdaderas)
- ¿Es completo?(Prueba todo lo verdadero)

Conclusión

Tenemos un cálculo para demostrar basado en reglas

- ¿Es consistente? (Sólo prueba cosas verdaderas)
- ¿Es completo?(Prueba todo lo verdadero)

Las reglas de deducción que hemos dado anteriormente son consistentes y completas para la lógica proposicional