1 Representación e inferencia lógicas

Camilo Palazuelos Calderón

REPRESENTACIÓN DEL CONOCIMIENTO
Grado en Ingeniería Informática
Mención en Computación

Curso 2023-2024

Información útil

Sobre la práctica y su entrega

- **Objetivos de la práctica**
  - Manejar estructuras de datos básicas en Python
  - Implementar el encadenamiento hacia delante en lógica proposicional
  - Desarrollar un algoritmo que decida si el encadenamiento es completo
- **Laboratorio:** 6 y 13 de octubre de 14:30 a 16:30
  - La fecha límite de entrega es el 19 de octubre a las 23:59

L	M	X	J	V
2	3	4	5	6
9	10	11	12	13
16	17	18	19	20
23	24	25	26	27
30	31			

- **Qué entregar**
  - Memoria con respuestas a las preguntas formuladas en el guion de la práctica
  - Código desarrollado (y material adicional si lo consideráis oportuno)

Lógicas y modelos

De la semántica a la consecuencia lógica

- Las lógicas son lenguajes formales que posibilitan **razonar**
  1. *representando e*
     - La sintaxis del lenguaje indica qué fórmulas están bien formadas;
     - la semántica, el valor de verdad de cada fórmula en cada modelo
  2. *infiriendo*
     - $M(\varphi) = \{\mu \mid \mu(\varphi)\}$, donde $\mu(\varphi)$ es el valor de verdad de φ en μ
     - Una fórmula φ implica otra ψ si y solo si $\mu(\psi)$ para todo $\mu \in M(\varphi)$

Representación lógica

$$M(\underbrace{\varphi_1 \wedge \cdots \wedge \varphi_k}_{\text{Base de conocimiento BC}}) = \bigcap_i M(\varphi_i)$$

Inferencia lógica

$$\text{BC} \models \psi \iff \underbrace{M(\text{BC}) \subseteq M(\psi)}_{\text{Verificación de modelos}}$$

Inferencia y satisfacibilidad

La inferencia en lógica proposicional se reduce a SAT

- **Proposición 1.1.** Sean BC y ψ dos fórmulas proposicionales.

$$\text{BC} \models \psi \iff \text{BC} \wedge \neg\psi \text{ es insatisfacible}$$

  - Una fórmula φ es *insatisfacible* si $\neg\mu(\varphi)$ para todo modelo μ
  - Es equivalente a demostrar BC $\models \psi$ por *reducción al absurdo*

i	$\mu_i(\text{BC})$	$\mu_i(\psi)$	$\mu_i(\text{BC} \wedge \neg\psi)$
1	0	0	0
2	0	1	0
3	1	0	1
4	1	1	0

- BC $\models \psi$ en lógica proposicional es un problema **co-NP-completo**

Lógicas y demostraciones

De la sintaxis a la consecuencia lógica

- Sea P un conjunto de fórmulas y κ, una fórmula
  - Una *regla de inferencia* (P, κ) indica que de P se deriva κ
  - A P se lo denomina conjunto de *premisas* y a κ, *conclusión*
- Un **encadenamiento de reglas** i permite demostrar que $\mathrm{BC} \vdash_i \psi$
  - Si y solo si $\forall \mathrm{BC}\, \forall \psi\, [\mathrm{BC} \vdash_i \psi \implies \mathrm{BC} \models \psi]$, se dice que i es *correcto*
  - Si y solo si $\forall \mathrm{BC}\, \forall \psi\, [\mathrm{BC} \models \psi \implies \mathrm{BC} \vdash_i \psi]$, se dice que i es *completo*

Modus ponens

$$\left.\begin{array}{l} \varphi \to \kappa \\ \varphi \end{array}\right\} P \quad \Big/ \; \kappa$$

Encadenamiento de reglas ($\mathrm{BC} \vdash_i \psi$)

while $\exists \varphi\, \exists \kappa\, [\{\varphi \to \kappa, \varphi\} \subseteq \mathrm{BC}$ **and** $\kappa \notin \mathrm{BC}]$

$\quad \mathrm{BC} \leftarrow \mathrm{BC} \cup \{\kappa\}$

return $\psi \in \mathrm{BC}$

Encadenamiento de *modus ponens*

Completo en lógica proposicional para BC de Horn

- **Proposición 1.2.** Para toda BC de Horn proposicional,

$$\{q \mid \mathrm{BC} \vdash_i q\} = \{q \mid \mathrm{BC} \models q\}$$

  - Una BC de Horn es una conjunción de *cláusulas de Horn*

Modus ponens

$$\frac{p_1 \wedge \cdots \wedge p_\ell \to q \qquad p_1, \ldots, p_\ell}{q}$$

Cláusulas de Horn

$$\top \to p \quad \textcircled{1}\ \textit{hecho}$$
$$p_1 \wedge \cdots \wedge p_\ell \to q \quad \textcircled{2}\ \textit{regla}$$
$$p_1 \wedge \cdots \wedge p_\ell \to \bot \quad \textcircled{3}\ \textit{objetivo}$$

- $\mathrm{BC} \models \psi$ con BC de Horn proposicionales es un problema **P-completo**

Tareas y preguntas

Qué hacer y a qué dar respuesta en la memoria

1. [8 PUNTOS] **Codificación de los algoritmos descritos**
   - Implementad el encadenamiento hacia delante en lógica proposicional
   - Desarrollad un algoritmo que decida si el encadenamiento es completo
2. [2 PUNTOS] **Eficacia de vuestras propuestas**
   - Mostrad, con ejemplos variados, que todo funciona correctamente