Lógica Proposicional: Deducciones formales

Pablo Barceló

La noción de consecuencia lógica

La noción de consecuencia es fundamental para cualquier lenguaje, y es de particular interés en AI:

- Formaliza la idea de que un estado de cosas es *implicado* por nuestra representación del mundo.
- Representa la idea de que nuestros sistemas deben ser capaces de *razonar*:

  Si Pedro es un hombre y todo hombre es mamífero, entonces el sistema debería concluir que Pedro es mamífero.
- Lo último es particularmente importante puesto que lo más normal es que nuestra representación del mundo sea incompleta.

La noción de consecuencia lógica

Uno de los problemas fundamentales de la representación del conocimiento es cómo automatizar un proceso de razonamiento deductivo.

La noción de consecuencia lógica

Uno de los problemas fundamentales de la representación del conocimiento es cómo automatizar un proceso de razonamiento deductivo.

Un algoritmo naïve para chequear si $\Sigma \models \phi$.

- Construya tabla de verdad para Σ y ϕ.
- Compruebe si para toda asignación σ tal que $\sigma(\Sigma) = 1$ se tiene que $\sigma(\phi) = 1$.

La noción de consecuencia lógica

Uno de los problemas fundamentales de la representación del conocimiento es cómo automatizar un proceso de razonamiento deductivo.

Un algoritmo naïve para chequear si $\Sigma \models \phi$.

- Construya tabla de verdad para Σ y ϕ.
- Compruebe si para toda asignación σ tal que $\sigma(\Sigma) = 1$ se tiene que $\sigma(\phi) = 1$.

¿Cuántos recursos utiliza este algoritmo?

La noción de consecuencia lógica

Uno de los problemas fundamentales de la representación del conocimiento es cómo automatizar un proceso de razonamiento deductivo.

Un algoritmo naïve para chequear si $\Sigma \models \phi$.

- Construya tabla de verdad para Σ y ϕ.
- Compruebe si para toda asignación σ tal que $\sigma(\Sigma) = 1$ se tiene que $\sigma(\phi) = 1$.

¿Cuántos recursos utiliza este algoritmo?

Espacio lineal y tiempo exponencial.

Paréntesis sobre eficiencia computacional

Número estimado de electrones en el universo $\leq 10^{130}$.

Si $n = 1000$ y en cada electrón del universo tuviéramos un supercomputador que ejecuta 10^{50} operaciones por segundo, entonces para verificar si φ es satisfacible necesitaríamos:

$$\frac{2^{1000}}{10^{50} \cdot 10^{130}} \approx 10^{121} \text{ segundos.}$$

Edad estimada del universo $< 10^{18}$ segundos! Y φ se puede almacenar en algunos kilobytes de memoria!

¿Existe un algoritmo eficiente para el problema de satisfacibilidad?

Complejidad de consecuencia lógica

Recapitulemos un poco:

$$\Sigma \models \phi \iff \Sigma \cup \{\neg\phi\} \text{ es insatisfacible.}$$

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

Complejidad de consecuencia lógica

Recapitulemos un poco:

$$\Sigma \models \phi \iff \Sigma \cup \{\neg\phi\} \text{ es insatisfacible.}$$

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

Pero el problema de (in)satisfacibilidad es (co)NP-completo (Cook'71).

Complejidad de consecuencia lógica

Es muy poco probable que exista un algortimo *eficiente* que resuelva el problema de consecuencia lógica:

> Si lo encontraramos resolveríamos muchos otros problemas que se creen difíciles: Vendedor viajero, 3-colorabilidad, etc.

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

Formas normales: DNF

Decimos que una fórmula φ está en forma normal disjuntiva (DNF) si φ es de la forma:

$$\bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{n_i} l_{i,j} \right),$$

donde cada $l_{i,j}$ es un literal, es decir, una letra proposicional o la negación de una letra proposicional.

Ejemplo: $(p \wedge q) \vee (\neg p \wedge r)$.

Teorema

Toda fórmula es equivalente a una fórmula en DNF.

Formas normales: DNF

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

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

Formas normales: DNF

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

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

¿Es DNF la forma natural en la que expresamos nuestro conocimiento?

Formas normales: DNF

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

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

¿Es DNF la forma natural en la que expresamos nuestro conocimiento?

No podemos dar una respuesta definitiva a esta pregunta. Pero sí parece que muchas especificaciones naturales no están en DNF:

Lo natural es que nuestras especificaciones sean conjuntos de restrcccciones de la forma:

$$p_1 \wedge \cdots \wedge p_n \rightarrow q_1 \vee \cdots \vee q_m$$

Formas normales: CNF

Cada una de estas especificaciones es en realidad una fórmula en forma normal conjuntiva (CNF):

$$\bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n_i} l_{i,j} \right),$$

donde cada $l_{i,j}$ es un literal, es decir, una letra proposicional o la negación de una letra proposicional.

Proposición

Toda fórmula es equivalente a una fórmula en CNF.

Desafortunadamente el problema de satisfacibilidad para CNF también es NP-completo (Cook'71).

La resolución proposicional

De todas formas, mostraremos un procedimiento general de razonamiento deductivo que verifica si un conjunto de fórmulas en CNF es satisfacible.

Nuestro método procedural se conoce como resolución:

- Creado por Alan Robinson en 1965.
- Generaliza a lenguajes más expresivos.
- Base del motor de inferencia de Prolog.

El método general

Para verificar si $\Sigma \models \phi$:

- Transforme cada fórmula en $\Sigma \cup \{\neg\phi\}$ a una fórmula en CNF.
- Utilize resolución para verificar si el conjunto de cláusulas resultante es insatisfacible.

Por resultados anteriores, el segundo paso es equivalente a verificar si una contradicción cualquiera puede ser derivada desde el conjunto de fórmulas:

> En nuestro caso usaremos una contradicción fija: La disyunción vacía $\{\}$.

Relevancia: Al tener un objetivo fijo nos ahorramos mucho trabajo en el diseño de heurísticas.

Reducción a cláusulas

Una cláusula es una disyunción de literales, pero la representaremos como conjuntos de literales.

Ejemplo: $p \vee \neg q \vee \neg r$ es representada como el conjunto $\{p, q, r\}$, y la clusula $p \vee p$ es representada como $\{p\}$.

Por tanto, toda fórmula en CNF es de la forma $C_1 \wedge C_2 \wedge \cdots \wedge C_n$, donde cada C_i es una clasula.

Podemos representar esta fórmula como $\{C_1, C_2, \ldots, C_n\}$.

La regla de resolución

Para verificar que $\Sigma \models \square$ no queremos usar valuaciones, queremos usar alguna regla sintáctica.

Notación: Si $l = p$, entonces $\bar{l} = \neg p$, y si $l = \neg p$, entonces $\bar{l} = p$.

Dadas cláusulas C_1, C_2, C_3, C_4 y literal l.

Regla de resolución:

$$\frac{C_1 \cup \{l\} \quad C_2 \cup \{\bar{l}\}}{C_1 \cup C_2}$$

La regla es correcta: $\{C_1 \vee l,\ C_2 \vee \bar{l}\} \models C_1 \vee C_2$

Ejemplo de aplicación de resolución

Ejemplo:

$$\frac{\{\neg p, q\} \quad \{\neg q, r\}}{\{\neg p, r\}}$$

Tenemos que: $\{\neg p \vee q,\ \neg q \vee r\} \models \neg p \vee r$.

- O equivalentemente, $\{p \rightarrow q,\ q \rightarrow r\} \models p \rightarrow r$.

Casos particulares de resolución

Algunos casos particulares de aplicación de la regla de resolución:

$$\frac{C_1 \cup \{l\} \quad \{\bar{l}\}}{C_1} \qquad\qquad \frac{\{l\} \quad \{\bar{l}\}}{\square}$$

En el último caso estamos diciendo que $\{l, \bar{l}\}$ es insatisfacible.

Demostraciones por resolución

Dado: Conjunto de cláusulas Σ y una cláusula C.

Una demostración por resolución de C desde Σ es una secuencia de cláusulas $C_1, C_2, \ldots, C_n$ tal que:

- Para cada $i \leq n$:
  - $C_i \in \Sigma$ o
  - existen $j, k < i$ tales que C_i es obtenido desde C_j y C_k usando la regla de resolucin.
- $C_n = C$.

Notación: $\Sigma \vdash C$

Ejemplo de demostración por resolución

Ejemplo: $\Sigma = \{p \vee q \vee r,\ \neg p \vee s,\ \neg q \vee s,\ \neg r \vee s\}$ y $C = q \vee r \vee s$.

Una demostración de C desde Σ:

(1)	$\{p, q, r\}$	pertenece a Σ.
(2)	$\{\neg p, s\}$	pertenece a Σ.
(3)	$\{q, r, s\}$	resolución de (1) y (2).

¿Existe otra demostración de C desde Σ?

Ejemplo de demostración por resolución

Ejemplo: $\Sigma = \{p \vee q \vee r,\ \neg p \vee s,\ \neg q \vee s,\ \neg r \vee s,\ \neg s\}$ y $C = \square$.

(1)	$\{p, q, r\}$	pertenece a Σ.
(2)	$\{\neg p, s\}$	pertenece a Σ.
(3)	$\{q, r, s\}$	resolución de (1) y (2).
(4)	$\{\neg q, s\}$	pertenece a Σ.
(5)	$\{r, s\}$	resolución de (3) y (4).
(6)	$\{\neg r, s\}$	pertenece a Σ.
(7)	$\{s\}$	resolución de (5) y (6).
(8)	$\{\neg s\}$	pertenece a Σ.
(9)	$\square$	resolución de (7) y (8).

Correctitud y completidad

Podríamos agregar otras reglas a nuestro sistema de demostración. ¿Cómo sabemos si un conjunto de reglas es *bueno*?

Usamos dos criterios: Correctitud y completidad.

Correctitud: Si C se puede deducir desde Σ usando el conjunto de reglas, entonces $\Sigma \models C$.

Teorema (Correctitud)

La regla de resolución es correcta.

Por tanto, si $\Sigma \vdash \square$ entonces Σ es insatisfacible.

Correctitud y completidad

Completidad: Si $\Sigma \models C$, entonces es posible deducir C desde Σ usando el conjunto de reglas.

Correctitud y completidad

Completidad: Si $\Sigma \models C$, entonces es posible deducir C desde Σ usando el conjunto de reglas.

Ejercicio: Demuestre que la regla de resolución no es completa.

Correctitud y completidad

Completidad: Si $\Sigma \models C$, entonces es posible deducir C desde Σ usando el conjunto de reglas.

Ejercicio: Demuestre que la regla de resolución no es completa.

Tendríamos que agregar otras reglas si queremos completidad.

Pero: Sólo queremos usar resolución para demostrar que un conjunto de cláusulas es insatisfacible.

Forma débil de completidad: Si $\Sigma \models \square$, entonces es posible deducir $\square$ desde Σ usando el conjunto de reglas.

Correctitud y completidad

Teorema (Completidad débil)

La regla de resolución es correcta en la forma débil.

Por tanto, $\Sigma \models \square \Leftrightarrow \Sigma \vdash \square$, y, por tanto, resolución es un buen conjunto de reglas.

Pregunta: ¿Qué otros criterios podrían ser útiles para evaluar un conjunto de reglas?

Demostración del teorema de completidad

Demostraremos el teorema anterior para el caso cuando Σ es finito. El caso infinito puede ser demostrado usando técnicas más avanzadas (compacidad).

La demostración será por inducción en el número de variables proposicionales.

Dado: conjunto insatisfacible de fórmulas Σ.

Por demostrar: $\Sigma \vdash \square$

Demostración del teorema de completidad

Caso base: Sea $n = 0$. Entonces $\Sigma = \emptyset$, y trivialmente $\Sigma \vdash \square$.

Demostración del teorema de completidad

Caso base: Sea $n = 0$. Entonces $\Sigma = \emptyset$, y trivialmente $\Sigma \vdash \square$.

Caso inductivo: Supongamos que la propiedad es válida para n y que Σ contiene $n + 1$ letras proposicionales.

Sea l un literal mencionado en Σ. Defina
$\Sigma(l) = \{C - \bar{l} \mid C \in \Sigma, l \notin C\}$.

Lemma: Si Σ es insatisfacible, entonces $\Sigma(l)$ es insatisfacible.

Ejercicio: Demuestre el lemma.

Demostración del teorema de completidad

Entonces, existen:

- Demostración por resolución $C_1, \ldots, C_n$ de $\square$ desde $\Sigma(l)$.
- Demostración por resolución $D_1, \ldots, D_m$ de $\square$ desde $\Sigma(\bar{l})$.

Idea: Convertir a $C_1, \ldots, C_n$ en una demostración por resolución de $\bar{l}$ o $\square$ desde Σ, y a $D_1, \ldots, D_m$ en una demostración por resolución de l o $\square$ desde Σ.

En cualquier caso, hay una demostración por resolución de $\square$ desde Σ.