

Lectura

Inteligencia Artificial

Texto tomado desde los libros, Introduction to Logic Programming de Michael Genesereth y Vinay K. Chaudhri y Introduction to Artificial Intelligence Second Edition 2018 de Wolfgang Ertel.

"El presente texto ha sido preparado de manera exclusiva para los alumnos del Curso de Inteligencia Artificial, que forma parte de la Plan de Estudio de la Escuela de Ciencia de Computación, según el artículo 44 de la Ley sobre el Derecho de Autor, D.L. N°822. Queda prohibida su difusión y reproducción por cualquier medio o procedimiento, total o parcialmente fuera del marco del presente curso".

1 Notas adicionales de lógica proposicional

1.1 Semántica

Definición Un mapeo $I : \Sigma \rightarrow \{T, F\}$, que asigna un valor de verdad a cada variable de proposición, se llama interpretación.

Debido a que cada variable de proposición puede tomar dos valores de verdad, cada fórmula de lógica proposicional con n variables diferentes tiene 2^n interpretaciones diferentes. Definimos los valores de verdad para las operaciones básicas mostrando todas las posibles interpretaciones en una tabla de verdad.

Definición: Dos fórmulas F y G se denominan semánticamente equivalentes si toman el mismo valor de verdad para todas las interpretaciones. Escribimos $F \equiv G$.

La equivalencia semántica sirve sobre todo para poder utilizar el metalenguaje, es decir, el lenguaje natural, para hablar del lenguaje objeto, es decir, la lógica. El enunciado $A \equiv B$ transmite que las dos fórmulas A y B son semánticamente equivalentes. El enunciado $A \Leftrightarrow B$, por otro lado, es un objeto sintáctico del lenguaje formal de la lógica proposicional.

Según el número de interpretaciones en las que una fórmula es verdadera, podemos dividir las fórmulas en las siguientes clases:

Definición 2.5 Una fórmula se llama:

- Satisfactoria si es cierta para al menos una interpretación.
- Lógicamente válida o simplemente válida si es cierta para todas las interpretaciones. Las fórmulas verdaderas también se llaman tautologías.
- Insatisfactoria si no es cierta para alguna interpretación.

Toda interpretación que satisface una fórmula se llama modelo de la fórmula.

La negación de toda fórmula generalmente válida es insatisfactoria. La negación de una fórmula F satisfactoria, pero no generalmente válida, es satisfactoria.

1.2 Sistemas de prueba

En IA nos interesa tomar el conocimiento existente y de él, derivar nuevos conocimientos o responder preguntas. En lógica proposicional, esto significa mostrar que una base de conocimiento KB , es decir,

una fórmula de lógica proposicional (posiblemente extensa), sigue una fórmula Q . Por lo tanto, primero definimos el término *entailment*.

Definition Una fórmula KB implica (entails) una fórmula Q (o Q se sigue de KB) si cada modelo de KB es también un modelo de Q . Escribimos $KB \models Q$.

En otras palabras, en cada interpretación en la que KB es verdadera, Q también lo es. Más sucintamente, siempre que KB es verdadero, Q también lo es.

Toda fórmula que no es válida elige, por así decirlo, un subconjunto del conjunto de todas las interpretaciones como modelo. Tautologías como $A \vee \neg A$, por ejemplo, no restringen el número de interpretaciones satisfactorias porque su proposición es vacía. Por tanto, la fórmula vacía es cierta en todas las interpretaciones. Para cada tautología T entonces $\emptyset \models T$. Intuitivamente esto significa que las tautologías son siempre verdaderas, sin restricción de las interpretaciones por una fórmula. Para abreviar escribimos $\models T$.

Ahora mostramos una conexión importante entre el concepto semántico de implicación e implicación sintáctica.

Teorema de deducción

$$A \models B \quad \text{si solo si} \quad \vdash A \Rightarrow B.$$

Si deseamos demostrar que KB implica Q , también podemos demostrar mediante el método de la tabla de verdad que $KB \Rightarrow Q$ es una tautología. Por lo tanto, tenemos nuestro primer sistema de prueba para la lógica proposicional, que se automatiza fácilmente. La desventaja de este método es el tiempo de cálculo en el peor de los casos. Específicamente, en el peor de los casos con n variables de proposición, para todas las 2^n interpretaciones de las variables se debe evaluar la fórmula $KB \Rightarrow Q$. Por tanto, el tiempo de cálculo crece exponencialmente con el número de variables. Por lo tanto, este proceso no se puede utilizar para grandes cantidades de variables, al menos en el peor de los casos.

Si una fórmula KB implica una fórmula Q , entonces, según el teorema de deducción $KB \Rightarrow Q$ es una tautología. Por tanto, la negación $\neg(KB \Rightarrow Q)$ es insatisfactorio. Tenemos entonces:

$$\neg(KB \Rightarrow Q) \equiv \neg(\neg KB \vee Q) \equiv KB \wedge \neg Q.$$

Por lo tanto, $KB \wedge \neg Q$ también es insatisfactorio.

Teorema (Prueba por contradicción) $KB \models Q$ si y solo si $KB \wedge \neg Q$ es insatisfactorio.

Para mostrar que la consulta Q se deriva de la base de conocimientos KB , también podemos agregar la consulta $\neg Q$ a la base de conocimientos y derivar una contradicción. Debido a la equivalencia $A \wedge \neg A \Leftrightarrow F$, sabemos que una contradicción no es satisfactoria. Por tanto, Q ha sido probado. Este procedimiento, que se utiliza con frecuencia en matemáticas, también se utiliza en varios cálculos de prueba automáticos, como el cálculo de resolución y en el procesamiento de programas PROLOG.

Una forma de evitar tener que probar todas las interpretaciones con el método de las tabla de verdad es la manipulación sintáctica de las fórmulas KB y Q mediante la aplicación de reglas de inferencia con el objetivo de simplificarlas, de manera que al final podamos ver instantáneamente que $KB \models Q$. Llamamos a este proceso derivación sintáctica y escribimos $KB \vdash Q$. Estos sistemas de prueba sintáctica se denominan cálculos (calculi). Para asegurarse de que un cálculo no generar errores, definimos dos propiedades fundamentales de los cálculos.

Definición Un cálculo se llama sólido (sound) si cada proposición derivada sigue semánticamente. Es decir, si se cumple para las fórmulas KB y Q que:

$$\text{si } KB \vdash Q \quad \text{entonces} \quad KB \models Q.$$

Un cálculo se llama completo si se pueden derivar todas las consecuencias semánticas. Es decir, para las fórmulas KB y Q se cumple lo siguiente:

$$\text{si } KB \models Q \quad \text{entonces} \quad KB \vdash Q.$$

La solidez de un cálculo asegura que todas las fórmulas derivadas sean de hecho consecuencias semánticas de la base de conocimiento. El cálculo no produce consecuencias falsas. La completitud de un cálculo, por otro lado, asegura que el cálculo no pase por alto nada. Un cálculo completo siempre encuentra una prueba si la fórmula que se va a demostrar se deriva de la base de conocimientos. Si un cálculo es sólido y completo, entonces la derivación sintáctica y la implicación semántica son dos relaciones equivalentes.

Para mantener los sistemas de prueba automáticos lo más simples posible, estos se hacen generalmente para operar en fórmulas en forma normal conjuntiva.

Definición Una fórmula está en forma conjuntiva normal (CNF) si y solo si consta de una conjunción

$$K_1 \wedge K_2 \wedge \cdots K_m$$

de cláusulas. Una cláusula K_i consta de una disyunción

$$L_{i1} \vee L_{i2} \vee \cdots \vee L_{in_i}$$

de literales. Un literal es una variable (literal positivo) o una variable negada (literal negativo).

Teorema Toda fórmula lógica proposicional puede transformarse en una forma normal conjuntiva equivalente.

Ahora solo nos falta un cálculo para la prueba sintáctica de fórmulas lógicas proposicionales. Comenzamos con el modus ponens, una regla de inferencia simple e intuitiva que, a partir de la validez de A y $A \Rightarrow B$, permite la derivación de B . Escribimos esto formalmente como:

$$\frac{A, \quad A \Rightarrow B}{B}.$$

Esta notación significa que podemos derivar la(s) fórmula(s) debajo de la línea de las fórmulas separadas por comas arriba de la línea. El modus ponens como regla por sí solo, aunque es sólido, no es completo. Si agregamos reglas adicionales podemos crear un cálculo completo, que, sin embargo, no deseamos considerar aquí (por ahora). En su lugar, investigaremos la regla de resolución:

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C} \quad (2.1)$$

La cláusula derivada se llama **resolvente**. Mediante una simple transformación obtenemos la forma equivalente:

$$\frac{A \vee B, \quad B \Rightarrow C}{A \vee C}.$$

Si ponemos A en F , vemos que la regla de resolución es una generalización del modus ponens. La regla de resolución es igualmente útil si falta C o si faltan A y C . En el último caso, la cláusula vacía se puede derivar de la contradicción $B \wedge \neg B$.

1.3 Resolución

Ahora generalizamos de nuevo la regla de resolución permitiendo cláusulas con un número arbitrario de literales. Con los literales $A_1, \dots, A_m, B, C_1, \dots, C_n$, la regla de resolución general dice:

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), \quad (\neg B \vee C_1 \vee \cdots \vee C_n)}{(A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n)}. \quad (2.2)$$

Llamamos complementarios a los literales B y $\neg B$. La regla de resolución elimina un par de literales complementarios de las dos cláusulas y combina el resto de literales en una nueva cláusula.

Para demostrar que a partir de una base de conocimiento KB , sigue una consulta Q , realizamos una prueba por contradicción. Siguiendo el teorema de contradicción debemos demostrar que se puede derivar una contradicción de $KB \wedge \neg Q$. En fórmulas en forma normal conjuntiva, aparece una contradicción en forma de dos cláusulas (A) y $(\neg A)$, que llevan a la cláusula vacía como su resolvente. El siguiente teorema nos asegura que este proceso realmente funciona como se desea.

Para que el cálculo esté completo, necesitamos una pequeña adición, como se muestra en el siguiente ejemplo. Sea la fórmula $(A \vee A)$ como nuestra base de conocimientos. Para mostrar mediante la regla de resolución que de allí podemos derivar $(A \wedge A)$, debemos mostrar que la cláusula vacía se puede derivar de $(A \vee A) \wedge (\neg A \vee \neg A)$. Solo con la regla de resolución, esto es imposible. Con la factorización, que permite eliminar copias de literales de cláusulas, este problema se elimina. En el ejemplo, una aplicación doble de factorización conduce a $(A) \wedge (\neg A)$ y un paso de resolución a la cláusula vacía.

Teorema El cálculo de resolución para la prueba de insatisfacción de fórmulas en forma normal conjuntiva es sólido y completo.

Debido a que el trabajo del cálculo de resolución es derivar una contradicción de $KB \wedge \neg Q$ es muy importante que la base de conocimiento KB sea consistente:

Definición Una fórmula KB se llama consistente si es imposible derivar de ella una contradicción, es decir, una fórmula de la forma $\phi \wedge \neg\phi$.

De lo contrario, cualquier cosa puede derivarse de KB . Esto es cierto no solo para la resolución, sino también para muchos otros cálculos.

De los cálculos para la deducción automática, la resolución juega un papel excepcional.

A diferencia de otros cálculos, la resolución tiene solo dos reglas de inferencia y funciona con fórmulas en forma normal conjuntiva. Esto simplifica su implementación. Una ventaja adicional en comparación con muchos cálculos radica en su reducción en el número de posibilidades para la aplicación de reglas de inferencia en cada paso de la demostración, por lo que el espacio de búsqueda es reducido y el tiempo de cálculo disminuido.

1.4 Ejemplo

El puzzle de lógica número 7, titulado *A charming English family*, del libro alemán *Fallgruben für Kopfüssler* dice (traducido al inglés):

Despite studying English for seven long years with brilliant success, I must admit that when I hear English people speaking English I'm totally perplexed. Recently, moved by noble feelings, I picked up three hitchhikers, a father, mother, and daughter, who I quickly realized were English and only spoke English. At each of the sentences that follow I wavered between two possible interpretations. They told me the following (the second possible meaning is in parentheses):
The father: "We are going to Spain (we are from Newcastle)."
The mother: "We are not going to Spain and are from Newcastle (we stopped in Paris and are not going to Spain)."
The daughter: "We are not from Newcastle (we stopped in Paris)." What about this charming English family?.

Para resolver este tipo de problema procedemos en tres pasos: formalización, transformación a la forma normal y prueba. En muchos casos, la formalización es, con mucho, el paso más difícil porque es fácil cometer errores u olvidar pequeños detalles.

Aquí utilizamos las variables S para *We are going to Spain*, N para *We are from Newcastle* y P para *We stopped in Paris* y obtenemos como formalización de las tres proposiciones de padre, madre e hija:

$$(S \vee N) \wedge [(\neg S \wedge N) \vee (P \wedge \neg S)] \wedge (\neg N \vee P).$$

Factorizar $\neg S$ en la subfórmula central trae la fórmula a CNF en un solo paso. Numerar las cláusulas con índices subíndexados produce:

$$KB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4.$$

Ahora comenzamos la prueba de resolución, al principio todavía sin una consulta Q. Una expresión de la forma $Res(m, n) : \langle clause \rangle_k$ significa que $\langle clause \rangle$ se obtiene mediante la resolución de la cláusula m con la cláusula n y es numerado k .

$$Res(1, 2): (N)_5$$

$$Res(3, 4): (P)_6$$

$$Res(1, 4): (S \vee P)_7$$

Podríamos haber derivado la cláusula P también de $Res(4, 5)$ o $Res(2, 7)$. Cada paso de resolución adicional conduciría a la derivación de cláusulas que ya están disponibles. Debido a que no permite la derivación de la cláusula vacía, se ha demostrado que la base de conocimiento no es contradictoria. Hasta ahora hemos derivado N y P . Para demostrar que $\neg S$ se cumple, agregamos la cláusula $(S)_8$ al conjunto de cláusulas como una consulta negada. Con el paso de resolución tenemos:

$$Res(2, 8): ()_9$$

la prueba está completa. Por tanto, $\neg S \wedge N \wedge P$ se cumple.

1.5 Cláusulas de Horn

Una cláusula en forma normal conjuntiva contiene literales positivos y negativos y se puede representar en la forma:

$$(\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n)$$

con las variables A_1, \dots, A_m y B_1, \dots, B_n . Esta cláusula se puede transformar en dos simples pasos a la forma equivalente:


$$A_1 \wedge \dots \wedge A_m \Rightarrow B_1 \vee \dots \vee B_n.$$

Esta implicación contiene la premisa, una conjunción de variables y la conclusión, una disyunción de variables. Por ejemplo, si hace buen tiempo y hay nieve en el suelo, iré a esquiar o trabajaré. es una proposición de esta forma. El receptor de este mensaje sabe con certeza que el remitente no va a nadar. Una declaración significativamente más clara sería si hace buen tiempo y hay nieve en el suelo, iré a esquiar. El receptor ahora tiene información definida. Por lo tanto, llamamos cláusulas que tienen como máximo un literal positivo de cláusulas definidas. Estas cláusulas tienen la ventaja de que solo permiten una conclusión y por lo tanto, son claramente más sencillas de interpretar. Muchas relaciones pueden describirse mediante cláusulas de este tipo. Por tanto, definimos:

Definición: Las cláusulas con a lo más un literal positivo de la forma:

$$(\neg A_1 \vee \dots \vee \neg A_m \vee B) \text{ or } (\neg A_1 \vee \dots \vee \neg A_m) \text{ or } B$$

o equivalente)

$$A_1 \wedge \dots \wedge A_m \Rightarrow B \text{ or } A_1 \wedge \dots \wedge A_m \Rightarrow f \text{ or } B.$$


se denominan cláusulas de Horn. Una cláusula con un solo literal positivo se llama hecho (fact). En las cláusulas con un literal negativo y uno positivo, el literal positivo se llama encabezado (head).

Las cláusulas de Horn son más fáciles de manejar no solo en la vida diaria, sino también en el razonamiento formal, como podemos ver en el siguiente ejemplo.

Sea la base de conocimientos que consiste de las siguientes cláusulas:

$$\begin{aligned} &(\text{buen_tiempo})_1 \\ &(\text{nevando})_2 \\ &(\text{nevando} \Rightarrow \text{nevar})_3 \\ &(\text{buen_tiempo} \wedge \text{nevar} \Rightarrow \text{esquiar})_4 \end{aligned}$$

Si ahora queremos saber si el esquiar se mantiene, esto se puede deducir fácilmente. Un modus ponens ligeramente generalizado es suficiente aquí como regla de inferencia:

$$\frac{A_1 \wedge \dots \wedge A_m, \quad A_1 \wedge \dots \wedge A_m \Rightarrow B}{B}.$$

La prueba de esquí tiene la siguiente forma $MP(i_1, \dots, i_k)$ que representa la aplicación del modus ponens en las cláusulas i_1 a i_k :

$$\begin{aligned} &MP(2, 3) : (\text{nevar})_5 \\ &MP(1, 5, 4) : (\text{esquiar})_6 : \end{aligned}$$

Con modus ponens obtenemos un cálculo completo para fórmulas que constan de cláusulas de Horn de lógica proposicional. Sin embargo, en el caso de grandes bases de conocimiento, modus ponens puede derivar muchas fórmulas innecesarias si se comienza con cláusulas incorrectas. Por lo tanto, en muchos casos es mejor utilizar un cálculo que comience con la consulta y trabaje hacia atrás hasta llegar a los hechos (facts). Dichos sistemas se denominan encadenamiento hacia atrás (**backward chaining**), en contraste con los sistemas de encadenamiento hacia adelante (**forward chaining**), que comienzan con hechos (facts) y finalmente derivan la consulta, como en el ejemplo anterior con el modus ponens.

Para el encadenamiento hacia atrás de cláusulas Horn, se utiliza la resolución SLD. SLD significa Selection rule driven linear resolution for definite clauses. En el ejemplo anterior, aumentado por la consulta negada ($\text{esqui} \Rightarrow F$):

$$\begin{aligned} &(\text{buen_tiempo})_1 \\ &(\text{nevando})_2 \\ &(\text{nevando} \Rightarrow \text{nevar})_3 \\ &(\text{buen_tiempo} \wedge \text{nevar} \Rightarrow \text{esquiar})_4 \\ &(\text{esquiar} \Rightarrow F)_5 \end{aligned}$$

Realizamos la resolución SLD comenzando con los pasos de resolución que se derivan de esta cláusula:

$$\begin{aligned} &\text{Res}(5, 4) : (\text{buen_tiempo} \wedge \text{nevar} \Rightarrow F)_6 \\ &\text{Res}(6, 1) : (\text{nevar} \Rightarrow F)_7 \\ &\text{Res}(7, 3) : (\text{nevando} \Rightarrow F)_8 \\ &\text{Res}(8, 2) : () \end{aligned}$$

y derivar una contradicción con la cláusula vacía. Aquí podemos ver fácilmente la *linear resolution*, lo que significa que el procesamiento posterior siempre se realiza en la cláusula derivada actualmente. Esto conduce a una gran reducción del espacio de búsqueda. Además, los literales de la cláusula actual siempre se procesan en un orden fijo (por ejemplo, de derecha a izquierda) (Selection rule driven). Los literales de la cláusula actual se denominan subobjetivos (subgoals). Los literales de la consulta negada son los objetivos (goals). La regla de inferencia para un paso dice:

$$\frac{A_1 \wedge \dots \wedge A_m \Rightarrow B_1, \quad B_1 \wedge B_2 \wedge \dots \wedge B_n \Rightarrow f}{A_1 \wedge \dots \wedge A_m \wedge B_2 \wedge \dots \wedge B_n \Rightarrow f} \quad \begin{matrix} \text{F} \\ \text{F} \end{matrix}$$

Antes de la aplicación de la regla de inferencia, deben probarse B_1, B_2, \dots, B_n , los subobjetivos actuales. Después la aplicación B_1 es reemplazada por el nuevo subobjetivo $A_1 \wedge \dots \wedge A_m$. Para demostrar que B_1 es verdadero, ahora debemos demostrar que $A_1 \wedge \dots \wedge A_m$ son verdaderos. Este proceso continúa hasta que la lista de subobjetivos de las cláusulas actuales (la denominada pila de objetivos (goal stack)) esté vacía. Con eso, se ha encontrado una contradicción. Si, para un subobjetivo $\neg B_i$, no hay una cláusula con el literal complementario B_i como encabezado de la cláusula, la prueba termina y no se puede encontrar ninguna contradicción. Por tanto, la consulta no puede demostrarse.

La resolución SLD juega un papel importante en la práctica porque los programas en el lenguaje de programación lógica PROLOG constan de cláusulas Horn de lógica predicada y su procesamiento se logra mediante la resolución SLD.