Programación Declarativa, 2023-2
Nota adicional: Resolución Binaria en Lógica de Primer Orden

Juan Pablo Yamamoto Zazueta

8 de febrero de 2023
Facultad de Ciencias UNAM

La **Resolución Binaria**, en el contexto de la lógica proposicional, es un algoritmo para la decisión de satisfacibilidad de una fórmula. Este cumple con las propiedades de consistencia (si la fórmula no es satisfacible, el algoritmo decide la insatisfacibilidad) y completitud (si el algoritmo reporta insatisfacibilidad, la fórmula no es satisfacible).

Cuando se lleva este mismo procedimiento a la lógica de primer orden (o de predicados), encontramos que sigue cumpliendo las propiedades anteriores, aunque ya no se puede garantizar que siempre termina. Esto es debido a la naturaleza de la lógica de primer orden, no a algún factor particular del algoritmo utilizado. No obstante, para fines prácticos ha mostrado empiricamente ser suficientemente bueno. Es por ello que se ha utilizado como fundamento para la implementación de sistemas deductivos, lenguajes lógicos (Prolog, por ejemplo) y sistemas de tipado.

El procedimiento de resolución binaria se va a apoyar de un **algoritmo de unificación**. Si bien en esta nota no veremos en detalle algún algoritmo de este estilo, es importante recordar en qué consiste.[1]

Definición 0.1. Una sustitución σ es un **unificador más general** (`umg`) para un conjunto de términos W si cumple: (1) $|W\sigma| = 1$ (al aplicar la sustitución a términos en un conjunto, todos son iguales entre sí) y (2) para cualquier unificador τ de W existe una sustitución θ tal que $\sigma\theta = \tau$.

Un algoritmo de unificación recibe como entrada un conjunto de términos W y regresa un `umg` σ, o reporta un **error** en caso de no existir el `umg`.

Entre las opciones conocidas destaca el algoritmo de Martinelli-Montanari. De igual manera, a partir de este se han desarrollado otros algoritmos similares con la finalidad de adaptarse mejor a los recursos de cómputo disponibles.

Además, para los propósitos de la resolución binaria necesitamos conocer en qué consiste la **Forma Clausular** de una fórmula.

Definición 0.2. Sean φ una fórmula y $fns(\varphi) = \forall x_1 \forall x_2 ... \forall x_n \psi$ su forma normal de Skolem de φ. Si $\psi = \mathcal{C}_1 \wedge \cdots \wedge \mathcal{C}_n$ entonces la forma clausular de φ, denotada $Cl(\varphi)$ es la secuencia de cláusulas

$$Cl(\varphi) = \mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n$$

donde cualesquiera dos cláusulas tienen variables ajenas.

[1] En las referencias se pueden encontrar notas de cursos y libros que cubren el tema.

Para llegar a esta forma se realiza el siguiente proceso:[2]

1. Rectificación de fórmulas.
2. Forma Normal Negativa.
3. Forma Normal Prenex.
4. Forma Normal de Skolem.
5. Forma Clausular.

Ejercicio 0.1. Sea $\varphi = \forall y\,(\forall y\,(T(y,y) \wedge \exists x P(x,y) \to \forall x R(x,y)) \to \exists z R(y,z))$. Encontrar $Cl(\varphi)$.

Solución: Aplicamos el procedimiento mencionado anteriormente mostrando sus 5 etapas:

1. Rectificación: $\forall y\,(\forall u\,(T(u,u) \wedge \exists v P(v,u) \to \forall x R(x,u)) \to \exists z R(y,z))$
2. Forma Normal Negativa: $\forall y\,(\exists u\,(T(u,u) \wedge \exists v P(v,u) \wedge \exists x \neg R(x,u)) \vee \exists z R(y,z))$
3. Forma Normal Prenex: $\forall y \exists z \exists u \exists x \exists v\,((T(u,u) \wedge P(v,u) \wedge \neg R(x,u)) \vee R(y,z))$
4. Forma Normal de Skolem: $\forall y\,(T(g(y),g(y)) \vee R(y,f(y))) \wedge (P(j(y),g(y)) \vee R(y,f(y)))$ $\wedge\,(\neg R(h(y),g(y)) \vee R(y,f(y)))$
5. Forma Clausular: $(T(g(y),g(y)) \vee R(y,f(y)))\,,\ (P(j(y),g(y)) \vee R(y,f(y)))\,,$ $(\neg R(h(y),g(y)) \vee R(y,f(y)))$

Teniendo lo anterior, es posible entender el procedimiento de Resolución Binaria. Este procedimiento está dado por la siguiente regla:

$$\frac{\mathcal{C} =_{\text{def}} C_1 \vee l \quad \mathcal{D} =_{\text{def}} D_1 \vee l' \quad \text{Var}(\mathcal{C}) \cap \text{Var}(\mathcal{D}) = \varnothing \quad \sigma = \texttt{umg}\{l^c, l'\}}{(C_1 \vee D_1)\sigma}$$

Nótese que llevar las fórmulas a su forma clausular garantiza que la condición de que las variables sean ajenas se cumple, además de facilitar el cómputo. Es por ello que la entrada al procedimiento de resolución binaria para la fórmula φ se da como el conjunto de elementos de $Cl(\varphi)$. El procedimiento es el siguiente:

Entrada: Un conjunto $\mathcal{C}$ de cláusulas.

Salida: Si el procedimiento termina, indica si las cláusulas son *satisfacibles* o *insatisfacibles*.

[2]Revisar la nota A del curso: *Nota Adicional sobre Formas Normales en Lógica de Primer Orden.*

Procedimiento:

1. Sea $S_0 = S$.
2. Dado S_i:
   a) Tomar $\mathcal{C}, \mathcal{D} \in S_i$ tal que $\mathcal{C} = C_1 \vee l$ y $\mathcal{D} = D_1 \vee l'$ tal que existe $\sigma = \texttt{umg}\{l^c, l'\}$ (l^c y l' son unificables).
   b) Definir $R = (C_1 \vee D_1)\,\sigma$.
   c) Si $R = \square$ (es la cláusula vacía), termina y reporta que S es insatisfacible. En otro caso, se define $S_{i+1} = S_i \cup \{R\}$.
   d) Repetir para todos los posibles pares de cláusulas.
3. Si $S_{i+1} = S_i$, termina y reporta que S es satisfacible. Además, se puede retornar la composición de todos los σ generados para indicar con qué sustituciones es satisfacible. En otro caso, repetir desde el paso 2 para S_{i+1}.

La manera en que se utiliza el procedimiento anterior para decidir la consecuencia lógica $\Gamma \vDash \varphi$ es a través de una prueba por contradicción: se niega la conclusión, se agrega al conjunto de premisas y se verifica que no es satisfacible (se llega a la cláusula vacía $\square$). Cuando esto sucede, se concluye que de Γ se sigue φ.

De igual manera, si se desea decidir si una fórmula φ se satisface, podemos hacer una prueba por contradicción. Al negar la fórmula, se busca una prueba de no satisfacibilidad. Es decir, si Γ es el conjunto de cláusulas $Cl(\neg\varphi)$, se resuelve $\Gamma \vDash \square$.

Ejercicio 0.2. Sea $\varphi = \neg\exists x \neg\left((\exists y\,(\exists w Q(w,x) \wedge P(x,y) \to P(x,y))) \wedge (\exists v\,(Q(v,x) \to \forall z R(z,x) \vee Q(v,x)))\right)$. Decidir si la fórmula es válida.

Solución: Primero se niega la fórmula para realizar la prueba por contradicción:

$$\exists x \neg\left((\exists y\,(\exists w Q(w,x) \wedge P(x,y) \to P(x,y))) \wedge (\exists v\,(Q(v,x) \to \forall z R(z,x) \vee Q(v,x)))\right)$$

Y se lleva la fórmula a su Forma Clausular:

$$\begin{array}{l}(Q(g(v,y),a) \vee Q(v,a))\,,\ (Q(g(v,y),a) \vee \neg R(f(v),a))\,,\ (Q(g(v,y),a) \vee \neg Q(v,a))\,,\\ (P(a,y) \vee Q(v,a))\,,\ (P(a,y) \vee \neg R(f(v),a))\,,\ (P(a,y) \vee \neg Q(v,a))\\ (\neg P(a,y) \vee Q(v,a))\,,\ (\neg P(a,y) \vee \neg R(f(v),a))\,,\ (\neg P(a,y) \vee \neg Q(v,a))\end{array}$$

Con ello, ya es posible realizar el proceso de Resolución Binaria. Primero se enumeran las hipótesis dadas por la Forma Clausular:

(1)	$Q(g(v,y),a) \vee Q(v,a)$	Hipótesis
(2)	$Q(g(v,y),a) \vee \neg R(f(v),a)$	Hipótesis
(3)	$Q(g(v,y),a) \vee \neg Q(v,a)$	Hipótesis
(4)	$P(a,y) \vee Q(v,a)$	Hipótesis
(5)	$P(a,y) \vee \neg R(f(v),a)$	Hipótesis
(6)	$P(a,y) \vee \neg Q(v,a)$	Hipótesis
(7)	$\neg P(a,y) \vee Q(v,a)$	Hipótesis
(8)	$\neg P(a,y) \vee \neg R(f(v),a)$	Hipótesis
(9)	$\neg P(a,y) \vee \neg Q(v,a)$	Hipótesis

Luego, se realiza el proceso de Resolución Binaria descrito anteriormente:

(10)	$Q(g(v,y),a)$	Res 1,3
(11)	$Q(g(v,y),a) \vee P(a,y)$	Res 1,6
(12)	$Q(g(v,y),a) \vee \neg P(a,y)$	Res 1,9
(13)	$P(a,y)$	Res 4,6
(14)	$Q(v,a)$	Res 4,7
(15)	$Q(v,a) \vee \neg R(f(v),a)$	Res 4,8
(16)	$\neg Q(v,a) \vee Q(v,a)$	Res 4,9
(17)	$\neg P(a,y) \vee P(a,y)$	Res 4,9
(18)	$\neg R(f(v),a)$	Res 5,8
(19)	$\neg Q(v,a) \vee \neg R(f(v),a)$	Res 5,9
(20)	$\neg Q(v,a)$	Res 6,9
(21)	$\neg P(a,y)$	Res 7,9
(22)	$\square$	Res 13,21

Como se alcanzó la cláusula vacía, es posible concluir que la fórmula dada es válida.

Referencias

[1] Ben-Ari, M. (2012). *Mathematical logic for computer science.* Springer.

[2] Hedman, S. (2008). *A first course in logic: An introduction in model theory, proof theory, computability, and complexity.* Oxford University Press.

[3] Krishnamurthi, S. (2007). *Programming languages: Application and interpretation.* Brown Univ.

[4] Miranda, F. E., & Reyes, A. L. (2023). *Notas para el curso de Lógica Computacional.* Revisión 2020-2. Facultad de Ciencias UNAM; Facultad de Ciencias UNAM.

[5] Soto Romero, M. (2023). *Notas para el curso de Programación Declarativa.* Revisión 2023-2. Facultad de Ciencias UNAM; Facultad de Ciencias UNAM.