# Capítulo 2

# El Método de Resolución

En este capítulo se realiza una descripción general del método de resolución, dado que el programa de razonamiento automático OTTER lo utiliza y prueba a través de refutación. Asimismo, se da una descripción y definición de lo que es un modelo (que es uno de los objetivos de la herramienta a desarrollar). Dado que ambas herramientas prueban a partir de fórmulas que se encuentran en forma clausular, se presenta un algoritmo para transformar una fórmula a su forma clausular.

## 2.1 Resolución

El método de resolución es una regla de inferencia que toma dos cláusulas y produce una tercera que es consecuencia lógica de éstas. El proceso consiste en identificar y borrar parejas complementarias de dos cláusulas, una de cada cláusula y luego, combinar las otras literales para formar una cláusula nueva [Ore94].

El método de resolución requiere que una fórmula se encuentre en forma clausular. Afortunadamente, es posible convertir cualquier conjunto de fórmulas en su "equivalente" [1] conjunto de fórmulas en forma clausular.

En el método de resolución, se dice que una lista de cláusulas es inconsistente si se puede derivar la cláusula vacía de una lista de cláusulas. En este método, para demostrar que una fórmula se puede derivar de un conjunto de sentencias, se realiza lo siguiente: se niega la conclusión y se agrega al conjunto de sentencias que conforman la premisa, y si a partir de estos argumentos es posible derivar la cláusula vacía se dice que la premisa deriva a la sentencia de la conclusión.

[1] La palabra "equivalente" la trataremos mas adelante en la sección 2.1.1.

## 2.2 Ejemplo del Método de Resolución

**Ejemplo 2.1** [**Ore94**]

Mostrar que la premisa dada por ~{A∧B} puede derivar la conclusión ~A∨B.

La premisa la podemos representar como {~A, ~B}, al aplicar la negación dentro de los argumentos.

Ahora al negar la conclusión resulta: {A∧B}, por lo tanto podemos denotarla como {A}, {B}.

Reescribiendo, y aplicando resolución se obtiene lo siguiente:

1. {~A, ~B}
2. {A} (Agregada)
3. {B} (Agregada)
4. {~B} (Resolución 1,2)
5. {} (Resolución 3,4)

A una derivación por resolución de una cláusula vacía se le denomina ***derivación por refutación***. De hecho lo que se busca, es mostrar que las premisas y la conclusión son inconsistentes refutando la consistencia de las cláusulas que resultan de unir ambos conjuntos de fórmulas. Por lo anterior, podemos decir que en el ejemplo 2.1, se probó por refutación.

Un probador de teorema automático utiliza un procedimiento de afirmación o de refutación para demostrar que un teorema T es una consecuencia lógica de los axiomas A. Un ***procedimiento de afirmación*** muestra que A⇒T es válido, mientras que un ***procedimiento de refutación*** prueba que la negación del teorema junto con los axiomas A∧~T constituyen una contradicción.

A continuación se describirá y definirá lo que es un ***modelo*** en lógica, junto con un algoritmo para transformar una fórmula cualquiera en un lenguaje de primer orden a su ***forma clausular***. Se iniciará con una descripción de la terminología que se utilizará para la explicación, para de esta forma poder abordar los temas mencionados.

## 2.3 ¿Qué es un Modelo?

En lo siguiente de esta sección, se utilizarán las letras griegas $\beta$, $\delta$, $\alpha$ o $\alpha_1,...,\alpha_n$, para referirnos a fórmulas de un lenguaje de lógica de primer orden.

Considerando a $\Sigma$ un conjunto de enunciados pertenecientes a un lenguaje de lógica de primer orden, y pensando en $\Sigma$ como un conjunto de hipótesis, como axiomas de una teoría o como una base de datos, y si $\beta$ es un enunciado particular, es natural preguntarnos:

- ¿Es $\beta$ consecuencia lógica de $\Sigma$?
- ¿ $\beta$ se sigue de $\Sigma$, lógicamente? o bien, ¿ $\beta$ se puede deducir (es teorema) a partir de $\Sigma$?

Lo anterior lo denotaremos de la siguiente manera:

$\Sigma \models \beta$ "$\beta$ es consecuencia lógica de $\Sigma$ "

$\Sigma \vdash \beta$ "$\beta$ es deducible (es teorema) a partir de $\Sigma$ "

En la segunda expresión se sobreentiende que nos referimos a un sistema formal de Cálculo de Predicados (CP), que cumple con el Teorema de Completud de la Lógica [Amo98]:

$$\Sigma \vdash \beta \Leftrightarrow \Sigma \models \beta$$

Si $\Sigma$ es finito, digamos $\Sigma=\{\alpha_1,...,\alpha_n\}$ entonces en vez de $\Sigma \vdash \beta \Leftrightarrow \Sigma \models \beta$ escribimos:

$$\alpha_1,...,\alpha_n \vdash \beta \quad \Leftrightarrow \quad \alpha_1,...,\alpha_n \models \beta$$

Y el problema que nos hemos planteado consiste en responder a la pregunta

$$¿\ \alpha_1,...,\alpha_n \models \beta?$$

Es importante precisar que el concepto "$\beta$ es una consecuencia lógica de $\alpha_1,...,\alpha_n$" (denotado por $\alpha_1,...,\alpha_n \models \beta$) significa que para toda interpretación respecto a la cual $\alpha_1,...,\alpha_n$ sean verdaderos, $\beta$ sea verdadero también.

Llamamos ***modelo*** de un conjunto de enunciados a cualquier interpretación respecto a la cual el conjunto de enunciados es verdadero. Con esta precisión de lo que significa la palabra modelo en lógica, podemos definir "$\beta$ es una consecuencia lógica de $\alpha_1,...,\alpha_n$" si y solo si todo modelo de $\alpha_1,...,\alpha_n$ es un modelo de $\beta$. Es decir, para cualquier interpretación, $\beta$ es verdadera cada vez que $\alpha_1,...,\alpha_n$ son verdaderas [Amo98].

**Teorema 2.1** (Teorema básico)

$$\alpha_1,...,\alpha_n \models \beta \quad \Leftrightarrow \quad (\alpha_1 \Lambda,...., \Lambda\alpha_n \Lambda\neg\beta) \quad \text{no tiene modelo.}$$

Demostración [Amo98]:

$\Rightarrow$ ) Si todo modelo de $\alpha_1,...,\alpha_n$ ha de ser modelo de $\beta$, no es posible que haya un modelo de $\alpha_1,...,\alpha_n$ que sea modelo de $\neg\beta$.

$\Leftarrow$ ) Si $(\alpha_1 \Lambda,...., \Lambda\alpha_n \Lambda\neg\beta)$ no tiene modelo, entonces cualquier modelo de $\alpha_1,...,\alpha_n$ será modelo de $(\alpha_1 \Lambda,...., \Lambda\alpha_n)$ y también de $\beta$, pues no lo será de $\neg\beta$.

Así pues, para probar que $\beta$ es consecuencia lógica de $\Sigma=\{\alpha_1,....,\alpha_n\}$, es suficiente probar que el conjunto de enunciados $\{\alpha_1,....,\alpha_n, \neg\beta\}$ no tiene modelos, lo cual es equivalente a que el enunciado $(\alpha_1 \Lambda ,...., \Lambda\alpha_n \Lambda \neg\beta)$ no tenga modelos o que sea inconsistente.

Sea $\varphi$ una fórmula cualquiera. Por ejemplo $\varphi=(\alpha_1 \Lambda,...., \Lambda\alpha_n \Lambda\neg\beta)$. Sea $\varphi$ la forma normal conjuntiva de la forma normal de Skolem de $\varphi$. Es un resultado de la lógica matemática que siempre es posible transformar de manera mecánica una fórmula cualquiera $\varphi$ en una fórmula normal conjuntiva $\varphi$, donde además los cuantificadores existenciales han sido eliminados por medio de constantes o funciones de Skolem, y donde los cuantificadores universales están presupuestos aunque no se escriben [Amo98]. Así, tal $\varphi$ es una conjunción de disyunciones de literales atómicas o negaciones atómicas. A esta fórmula se le llama forma clausular y la denotaremos con $Cl(\varphi)$. Cada una de las

disyunciones de una forma clausular se llama *cláusula*; es decir, una cláusula es una disyunción de literales tal que cada una de esas literales es un átomo o la negación de un atómo.

**Teorema 2.2** [**Amo98**]

Sea φ un enunciado y sea Cl(φ) la conjunción de las cláusulas correspondientes a φ (o forma clausular de φ). Entonces:

$$\varphi \text{ tiene modelo} \Leftrightarrow Cl(\varphi) \text{ tiene modelo}$$

Demostración en la literatura de J. Malitz [Mal84].

Al inicio de la sección 2.1 se mencionó que existe "equivalencia" entre una teoría y su respectiva conversión a forma clausular. Sin embargo, esta afirmación de equivalencia es débil debido a que no asegura que los modelos de ambas sean los mismos. Sin embargo, el teorema 2.2 es suficientemente útil en el presente trabajo

**Definición 2.1** [**Mal84**]

Una fórmula está rectificada si:

- No tiene presencias libres y acotadas de una misma variable, ni cuantificadores con alcances ajenos con la misma variable.
- No tiene dos cuantificadores que acotan presencias de una misma variable (cuantificaciones dobles).
- No tiene cuantificadores que no cuantifican (cuantificadores vacuos).

## 2.4 Algoritmo para transformar una fórmula a su forma clausular [Mal84]

**U**n algoritmo para transformar una fórmula cualquiera en un lenguaje de primer orden a su forma clausular, es el siguiente:

1. Rectificar la fórmula.
2. Eliminar las implicaciones y las dobles implicaciones de la fórmula:

$$(\alpha \rightarrow \beta) \equiv (\neg\alpha \vee \beta),\ \ (\alpha \leftrightarrow \beta) \equiv (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha)$$

3. Introducir al máximo las negaciones de la fórmula:

$$\neg\neg\alpha \equiv \alpha,\ \ \neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta,\ \ \ \neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$$

$$\neg\forall x\alpha \equiv \exists x\neg\alpha,\ \ \ \neg\exists x\alpha \equiv \forall x\neg\alpha$$

4. Skolemnizar, es decir, eliminar los cuantificadores existenciales, de izquierda a derecha. Por ejemplo, la skolemnización de $\neg\forall x\ \exists y\ P(x,y)$ es $\forall x\ P(x,f(x))$ y la skolemnización de $\exists y\neg Q(y)$ es $Q(c)$. Para más información sobre este tema consultar [Gen88].
5. Presuponer los cuantificadores universales: todas las variables (libres) se consideran universalmente cuantificadas.
6. Pasar a la forma normal conjuntiva:

$$\alpha \vee (\delta \wedge \beta) \equiv (\alpha \vee \delta) \wedge (\alpha \vee \beta)$$

El resultado de aplicar este algoritmo a una fórmula es la forma clausular de la misma. Como hemos visto, esta forma clausular es una conjunción de disyunciones de literales tal que cada una de esas literales es un átomo o la negación de un atómo.

Para poder comprender mejor el algoritmo de transformar una fórmula cualquiera en un lenguaje de primer orden a su forma clausular, se presenta el siguiente ejemplo:

## 2.5 Ejemplo de transformación de una fórmula a su forma clausular

**Ejemplo 2.2** [**Amo98**]

Sea $\varphi \equiv \exists x\ \forall y\ \neg P(x,y) \rightarrow (\neg \forall y\ Q(y) \wedge \exists x\ P(a,x)\ )$ transformar a su forma clausular.

| | | | |
|---|---|---|---|
| $\varphi$ | $\equiv$ | $\exists x\ \forall y\ \neg P(x,y) \rightarrow (\neg \forall w\ Q(w) \wedge \exists z\ P(a,z)\ )$ | Paso 1 |
| | $\equiv$ | $\neg \exists x\ \forall y\ \neg P(x,y) \vee (\neg \forall w\ Q(w) \wedge \exists z\ P(a,z)\ )$ | Paso 2 |
| | $\equiv$ | $\forall x\ \exists y\ P(x,y) \vee (\exists w\ \neg Q(w) \wedge \exists z\ P(a,z)\ )$ | Paso 3 |
| | $\equiv$ | $\forall x\ P(x,f(x)) \vee (\neg Q(c) \wedge P(a,d)\ )$ | Paso 4 |
| | $\equiv$ | $P(x,f(x)) \vee (\neg Q(c) \wedge P(a,d)\ )$ | Paso 5 |
| | $\equiv$ | $(P(x,f(x)) \vee \neg Q(c)) \wedge (P(x,f(x)) \vee P(a,d))$ | Paso 6 |

A la última fórmula obtenida se le denomina forma clausular de $\varphi$, es decir, $Cl(\varphi)$.