Resolución con orden y selección para la lógica $\mathcal{H}(@)$

Daniel Alejandro Gorín
Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires
Ciudad Universitaria – Pabellón I (1428) Buenos Aires, Argentina

Director: Carlos Eduardo Areces
INRIA Lorraine
615, rue du Jardin Botanique 54602 Villers les Nancy Cedex, France

Resumen

Las lógicas modales son reconocidas por combinar buenas propiedades computacionales (e.g. decidibilidad) con un alto poder expresivo. Estas propiedades han sido explotadas con éxito en Computación; un caso paradigmático lo constituyen las *description logics* (DLs), una familia de lógicas modales utilizadas en la construcción de *sistemas de representación de conocimiento* (knowledge representation systems o KRS). En este tipo de sistemas se puede dar una descripción conceptual de un *universo* junto con una enumeración de los elementos que lo constituyen, y a partir de ahí, inferir información que no se encuentra declarada en forma explícita. Este tipo de sistemas se utilizan, por ejemplo, en Inteligencia Artificial y en Lingüística Computacional, y han sido propuestos como base para el lenguaje de representación de ontologías que se usaría en la Web Semántica [4].

Las lógicas modales tradicionales, sin embargo, no permiten hacer referencia a elementos particulares del modelo, ni permiten tampoco representar igualdades. La lógica híbrida $\mathcal{H}(@)$ es la que se obtiene al agregar nominales y el operador de satisfacción @ a la lógica modal básica. Los nominales permiten *nombrar* elementos del modelo y junto con el operador @ incorporan una noción débil de igualdad. A pesar de su mayor poder expresivo, $\mathcal{H}(@)$ sigue siendo decidible: de hecho, su complejidad para el problema de la satisfaciblidad es igual al de la lógica modal básica, PSPACE-completo.

Habitualmente, los demostradores de teoremas para las lógicas modales están basados en algoritmos de tableau que, combinados con diversas heurísticas y optimizaciones, muestran buen comportamiento empírico. Sin embargo, muchas de estas heurísticas no son correctas cuando la lógica incorpora igualdad. Es interesante, entonces, investigar qué sucede con otras familias de algoritmos.

En [2] se propone un cálculo basado en resolución para $\mathcal{H}(@)$. Este cálculo, si bien consistente y completo, carece de las estrategias de orden y selección que son parte esencial de los demostradores para lógica de primer orden basados en resolución. Una implementación computacionalmente realista de un demostrador basado en resolución requiere de este tipo de estrategias para regular la generación desproporcionada de nuevas cláusulas y como guía dentro de un espacio de búsqueda extremadamente complejo.

El primer resultado de esta tesis es la definición de una estrategia de orden y selección muy general para el cálculo de resolución propuesto en [2], que preserva completitud refutacional (i.e., si bien el espacio de búsqueda disminuye drásticamente, el algoritmo de refutación sigue siendo correcto y completo). Como un paso necesario para la demostración general de completitud, se presenta un Teorema de Herbrand [11] para la lógica $\mathcal{H}(@)$. Este resultado permite reducir el problema de satisfacibilidad sobre la clase de todos los modelos posibles a la subclase de los modelos de Herbrand (este es, hasta donde sabemos, el primer resultado de este tipo para lógicas modales).

El segundo resultado que la tesis presenta es una demostración de terminación para el cálculo propuesto previamente. Tomando ventaja de la generalidad de la demostración anterior, podemos modificar el cálculo para asegurar terminación dada cualquier entrada. Si bien en [1] se demuestra que la complejidad del problema de satisfacibilidad para $\mathcal{H}(@)$ es PSPACE-completo (y por lo tanto decidible), nuestra demostración provee el primer algoritmo computacionalmente realizable.

Por último, los resultados teóricos de esta tesis fueron puestos a prueba en la re-implementación del prototipo HyLoRes, un demostrador automático basado en el cálculo de [2]. Los tests que presentamos muestran claramente que las estrategias de orden y selección también producen una mejora drástica en un algoritmo de resolución para lógicas híbridas, obteniendo tiempos de cómputo varios órdenes de magnitud menores.

1. Introducción

1.1. Lógicas modales e híbridas

Las lógicas modales proposicionales fueron propuestas para formalizar conceptos tales como posibilidad, obligación, conocimiento, etc. Sin embargo, en la década del '60, trabajos como [12, 13, 14] mostraron que estas lógicas también podían interpretarse usando ciertas estructuras relacionales, hoy llamadas *modelos de Kripke*. Estas estructuras se pueden ver como multigrafos dirigidos, donde los nodos se etiquetan con el conjunto de todas las proposiciones que valen en cada uno. A partir de ese momento, se amplió considerablemente el campo de aplicación de las lógicas modales; en particular se encontró que eran de gran utilidad en diversas disciplinas de Computación.

La ventaja de las lógicas modales en este terreno sobre otras lógicas clásicas está en que presentan un poder expresivo relativamente alto, sin perder decidibilidad: el problema de determinar si una fórmula de la lógica modal básica es *válida* (o, dualmente, *satisfacible*) es PSPACE-completo. Si bien con esta complejidad de peor caso este problema cae dentro de los denominados *intratables*, pruebas empíricas muestran que para un número importante de instancias de utilidad práctica, demostradores automáticos pueden dar una respuesta en tiempos razonables.

A pesar de su alto poder expresivo, las lógicas modales tradicionales presentan algunas limitaciones importantes, e.g. no permiten hacer referencia explícita a elementos concretos del dominio ni hablar de igualdad entre elementos. Las *lógicas híbridas* son una familia de extensiones de la lógica modal clásica que intentan resolver esta limitación mediante la introducción de *nominales* y operadores modales especiales.

Intuitivamente, un nominal es un *nombre único* para un elemento del modelo. Desde un punto de vista sintáctico, un nominal es similar a un símbolo de proposición, y puede ser usado en cualquier contexto donde éste sea aceptable. Por ejemplo, si i y j son nominales, y p es un símbolo de proposición, podemos escribir fórmulas como $i \wedge p \wedge \langle r \rangle (p \wedge [r]j)$. En este trabajo consideraremos la lógica híbrida básica $\mathcal{H}(@)$, que extiende a la lógica modal básica con nominales y el operador de satisfacción @, el cual permite *evaluar* una fórmula en un elemento particular del modelo.

Formalmente, el conjunto de fórmulas de la lógica $\mathcal{H}(@)$ se define en relación a una signatura $\langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$, donde PROP, NOM, REL son conjuntos no vacíos, infinitos numerables y disjuntos dos a dos. PROP es el conjunto de símbolos de proposición, NOM el conjunto de nominales y REL define el conjunto de modalidades. $\mathsf{ATOM} = \mathsf{PROP} \cup \mathsf{NOM}$ es el conjunto de símbolos atómicos. Dada una signatura $\mathcal{S} = \langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$ el conjunto de $\mathcal{H}(@)$-fórmulas sobre $\mathcal{S}$ se define inductivamente como $\mathcal{H}(@) = a \mid \neg\varphi \mid \varphi \wedge \varphi' \mid @_n \varphi \mid \langle r \rangle \varphi$, donde $a \in \mathsf{ATOM}$, $n \in \mathsf{NOM}$, $r \in \mathsf{REL}$ y $\varphi, \varphi' \in \mathcal{H}(@)$. Los demás operadores ($\vee$, $\rightarrow$, [ ], etc.) se definen en la forma usual; en particular $[r]\varphi = \neg\langle r \rangle \neg\varphi$.

Definición 1.1. Un *modelo híbrido* es una estructura $M = \langle W, \{R_i\}_{r_i \in \mathsf{REL}}, V \rangle$ donde

$$\begin{aligned} W &\quad \text{es un conjunto no vacío} \\ R_i \subseteq W \times W &\quad \text{es una relación binaria para cada } r_i \in \mathsf{REL} \\ V(p_i) \subseteq W &\quad \text{para cada } p_i \in \mathsf{PROP} \\ V(n_i) = \{w_i\} \subseteq W &\quad \text{para cada } n_i \in \mathsf{NOM} \end{aligned}$$

Definición 1.2. Dado un modelo híbrido $M = \langle W, \{R_i\}_{r_i \in \mathsf{REL}}, V \rangle$, la relación de satisfacibilidad $M, w \models \varphi$ (con $w \in W$) se lee "el modelo M satisface la fórmula φ en w" y se define inductivamente de la siguiente forma:

$$\begin{aligned} M, w \models a_i &\quad \text{sii} \quad w \in V(a_i), a_i \in \mathsf{ATOM} \\ M, w \models \neg\varphi &\quad \text{sii} \quad M, w \not\models \varphi \\ M, w \models \varphi_1 \wedge \varphi_2 &\quad \text{sii} \quad M, w \models \varphi_1 \text{ y } M, w \models \varphi_2 \\ M, w \models \langle r_i \rangle \varphi &\quad \text{sii} \quad \text{existe } w' \in W \text{ tal que } R_i(w, w') \text{ y } M, w' \models \varphi \\ M, w \models @_n \varphi &\quad \text{sii} \quad M, w' \models \varphi, \text{ con } w' \in V(n). \end{aligned}$$

De lo definición anterior, se desprende trivialmente que

$$M, w \models [r_i]\varphi \quad \text{sii} \quad \text{para todo } w' \in W \text{ tal que } R_i(w, w') \text{ y } M, w' \models \varphi$$

La lógica $\mathcal{H}(@)$, a través de fórmulas de la forma $@_i\, j$, donde i y j son nominales, introduce una noción débil de igualdad[1] que no tiene la lógica modal básica y es, por lo tanto, más expresiva que ésta. Sin embargo, es sabido que su problema de satisfacibilidad sigue siendo PSPACE-completo [1].

1.2. El cálculo de resolución para $\mathcal{H}(@)$

Las implementaciones más exitosas de demostradores de teoremas para lógicas modales están basados en el método de tableaux. Los buenos resultados obtenidos hasta el momento se apoyan en el uso de un número importante de heurísticas y refinamientos. Sin embargo, muchas de estas heurísticas dejan de funcionar cuando la lógica subyacente incluye alguna forma de igualdad. Al introducir nominales, la performance de los demostradores basados en tableaux baja considerablemente.

En este escenario, tiene sentido investigar qué sucede con otras familias de algoritmos. En particular, es interesante analizar el *método de resolución* que ha dado los mejores resultados en demostradores de teoremas para la lógica de primer orden con igualdad.

En primer término repasaremos el método de resolución para la lógica de primer orden; a continuación presentamos un cálculo de resolución para $\mathcal{H}(@)$ propuesto en [2]. Finalmente mostramos las limitaciones de dicho cálculo, que son la motivación de esta tesis.

1.2.1. Resolución para lógica de primer orden

El *cálculo de resolución* para lógica de primer orden fue propuesto por Robinson [18] a mediados de la década del '60 y es hoy la base de la gran mayoría de los demostradores automáticos para esta lógica. Para presentar el cálculo, utilizaremos la terminología estándar:

Definición 1.3. Los *términos de primer orden* se definen en forma inductiva: las variables son términos, y si $t_1, \ldots, t_n$ son términos y f es un símbolo de función de aridad n, entonces $f(t_1, \ldots, t_n)$ es un término. Dado un símbolo de relación R de aridad n, y $t_1, \ldots, t_n$, términos de primer orden, se dice que $R(t_1, \ldots, t_n)$ es un *átomo*. Finalmente si X es un átomo, se dice que tanto X como $\neg X$ son *literales*. Una *cláusula de primer orden* es un conjunto de literales; las cláusulas se interpretan como la disyunción de los literales que la conforman.

En resolución se opera sobre un conjunto de cláusulas, que se interpreta como la conjunción de todas las cláusulas. Para obtener esta representación, las fórmulas de entrada del cálculo deben ser pasadas a forma prenexa, *skolemnizadas* (reemplazar cada variable cuantificada existencialmente por un término) y expresadas en forma normal disyuntiva. El cálculo consiste, básicamente, en la saturación del conjunto de cláusulas por medio de una regla de inferencia, y una regla de factorización:

$$\text{(RES)} \quad \frac{C_1 \cup \{\varphi\} \quad C_2 \cup \{\neg\psi\}}{(C_1 \cup C_2)\sigma} \qquad\qquad \text{(FACT)} \quad \frac{C \cup \{\varphi, \psi\}}{(C \cup \{\varphi\})\sigma}$$

donde σ es el unificador más general de los átomos φ y ψ [6]. Ambas reglas se interpretan de la siguiente forma: si se encuentran cláusulas como las de la premisa, se debe agregar una cláusula como la que indica la conclusión al conjunto de cláusulas.

Es fácil convencerse de que la regla de resolución es consistente: si suponemos que $C_1 \cup \{\varphi\}$ y $C_2 \cup \{\neg\psi\}$ son verdaderas, y sabemos que $\varphi\sigma$ y $\neg\psi\sigma$ no pueden ser simultáneamente verdaderas (ya

[1] *Débil* en cuanto siempre debe participar al menos una constante.

que $\varphi\sigma = \psi\sigma$), entonces debemos concluir que o bien C_1 o bien C_2 es verdadera, con lo cual $C_1 \cup C_2$ también debe serlo. La consistencia de la regla de factorización es aún más evidente; ésta es necesaria para garantizar la completitud del cálculo en lógica de primer orden. No es necesaria, por ejemplo, en una versión proposicional del cálculo.

Sea N un conjunto de cláusulas, y sea N^* el conjunto que se obtiene al saturar N con las reglas de resolución y factorización; se puede demostrar que si N no es satisfacible (i.e. no existe un modelo que satisfaga todas las cláusulas), entonces N^* contiene la cláusula vacía [6]. El cálculo de resolución como algoritmo para determinar la satisfacibilidad de una fórmula de la lógica de primer orden φ consiste, entonces, en partir del conjunto inicial de cláusulas asociadas a φ, obtener su saturación con respecto a las reglas de resolución y factorización, y comprobar si este conjunto saturado contiene la cláusula vacía. De hecho, en este algoritmo no es necesario continuar con la saturación del conjunto una vez que se genera la cláusula vacía.

Es interesante notar que mientras que el *peor caso* del método de tableau se da cuando la fórmula de entrada es insatisfacible (ya que es necesario cerrar todas las ramas del árbol), el peor caso del cálculo de resolución se presenta cuando la fórmula de entrada es satisfacible, ya que en este caso debemos obtener la saturación completa del conjunto de cláusulas.

1.2.2. El cálculo de resolución híbrido

En [2] se propone un cálculo basado en resolución que puede ser usado sobre $\mathcal{H}(@)$. La versión que aquí presentaremos del cálculo trabaja con fórmulas en *forma normal negativa*, es decir, sólo es posible aplicar el operador de negación sobre átomos. Esto significa que tanto $\vee$ como $[\cdot]$ serán símbolos *primitivos*.

Definición 1.4. Sea una signatura $\mathcal{S} = \langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$, definimos inductivamente el conjunto $\mathcal{H}^{NNF}(@)$ como:

$$\mathcal{H}^{NNF}(@) ::= a \mid \neg a \mid \varphi \vee \varphi' \mid \varphi \wedge \varphi' \mid \langle r \rangle \varphi \mid [r]\varphi \mid @_i\, \varphi$$

donde $a \in \mathsf{PROP} \cup \mathsf{NOM}$, $r \in \mathsf{REL}$, $i \in \mathsf{NOM}$ y $\varphi, \varphi' \in \mathcal{H}^{NNF}(@)$. Además, a las fórmulas de la forma $@_i\, \varphi$ las llamaremos *fórmulas-@*.

Al igual que el cálculo de resolución para lógica de primer orden, el cálculo de resolución híbrido trabaja con conjuntos de *cláusulas*. Una cláusula, en este contexto, es un conjunto de fórmulas-@ arbitrarias pertenecientes a $\mathcal{H}^{NNF}(@)$. Aquí también, una cláusula representa una disyunción, pero no hay ninguna otra restricción en cuanto a la *forma* que deben tener las fórmulas; son las propias reglas del cálculo las que se encargan de armar cláusulas con fórmulas cada vez más simples. Vale notar que considerar sólo fórmulas-@ en las cláusulas no es una restricción en términos de satisfacción: una fórmula φ es satisfacible, si y sólo si para cualquier nominal i que no aparezca en φ, $@_i\, \varphi$ lo es.

Dada una fórmula $\varphi \in \mathcal{H}^{NNF}(@)$, llamamos $\mathit{ClSet}(\varphi) = \{\{@_t\, \varphi\}\}$, donde t es un nominal que no aparece en φ. Podemos ahora definir $\mathit{ClSet}^*(\varphi)$ – el conjunto saturado de cláusulas correspondiente a φ – como el menor conjunto que incluye a $\mathit{ClSet}(\varphi)$ y que está clausurado bajo las reglas de la Figura 1.

Podemos agrupar las reglas del cálculo de acuerdo a la función que cumplen. Las reglas ($\wedge$), ($\vee$) y (@) se encargan de simplificar las fórmulas. La regla ($\langle r \rangle$) es un paso de skolemnización mediante el cual se asigna explícitamente un nombre (un nominal nuevo) a un elemento del modelo previamente cuantificado existencialmente (por medio de un diamante). La regla (RES) es el equivalente de la regla de resolución para lógica de primer orden, mientras que la regla ($[r]$) codifica una unificación no trivial y un paso de resolución. Finalmente, las reglas (SYM), (REF) y (PARAM) son el equivalente del conjunto estándar de reglas para manejar igualdad en resolución para lógica de primer orden [5].

La construcción de $\mathit{ClSet}^*(\varphi)$ es un algoritmo correcto y completo para verificar la satisfacibilidad de fórmulas de $\mathcal{H}^{NNF}(@)$: φ es insatisfacible si y sólo si la cláusula vacía $\{\}$ es un elemento

$$(\wedge)\quad \frac{Cl \cup \{@_t\,(\varphi_1 \wedge \varphi_2)\}}{\begin{array}{c} Cl \cup \{@_t\,\varphi_1\} \\ Cl \cup \{@_t\,\varphi_2\} \end{array}} \qquad (\vee)\quad \frac{Cl \cup \{@_t\,(\varphi_1 \vee \varphi_2)\}}{Cl \cup \{@_t\,\varphi_1, @_t\,\varphi_2\}} \qquad (@)\quad \frac{Cl \cup \{@_t\,@_s\,\varphi\}}{Cl \cup \{@_s\,\varphi\}}$$

$$(\text{RES})\quad \frac{Cl_1 \cup \{@_t\,\varphi\} \quad Cl_2 \cup \{@_t\,\neg\varphi\}}{Cl_1 \cup Cl_2}$$

$$([r])\quad \frac{Cl_1 \cup \{@_t\,[r]\varphi\} \quad Cl_2 \cup \{@_t\,\langle r\rangle s\}}{Cl_1 \cup Cl_2 \cup \{@_s\,\varphi\}} \qquad (\langle r\rangle)\quad \frac{Cl \cup \{@_t\,\langle r\rangle\varphi\}}{\begin{array}{c} Cl \cup \{@_t\,\langle r\rangle n\} \\ Cl \cup \{@_n\,\varphi\} \end{array}} \text{ para un } n \text{ nuevo}$$

$$(\text{SYM})\quad \frac{Cl \cup \{@_t\,s\}}{Cl \cup \{@_s\,t\}} \qquad (\text{REF})\quad \frac{Cl \cup \{@_t\,\neg t\}}{Cl} \qquad (\text{PARAM})\quad \frac{Cl_1 \cup \{@_t\,s\} \quad Cl_2 \cup \{\varphi(s)\}}{Cl_1 \cup Cl_2 \cup \{\varphi(s/t)\}}$$

Figura 1: Reglas de resolución para $\mathcal{H}(@)$

de $\mathit{ClSet}^*(\varphi)$. Sin embargo, $\mathit{ClSet}^*(\varphi)$ puede ser un conjunto infinito, por lo tanto, existen fórmulas cuya satisfacibilidad este algoritmo no puede decidir en un número finito de pasos. En la Sección 3 investigamos cómo convertir este cálculo en un método de decisión para $\mathcal{H}(@)$.

1.2.3. Resolución con orden y selección

El cálculo de resolución para lógica de primer orden, tal como fue presentado en la Sección 1.2.1, es refutacionalmente completo; es decir, se puede mostrar que dado un conjunto insatisfacible de cláusulas, el conjunto que se obtiene al saturarlo respecto a la regla de resolución contiene la cláusula vacía [6]. A pesar de ello, no es viable construir demostradores en base a la aplicación directa de estas reglas.

El problema surge de la combinación de dos hechos fácilmente verificables: por un lado, el cálculo permite derivar cláusulas *redundantes*; por el otro, dado un conjunto de cláusulas C, si decimos que $d(C)$ representa el conjunto de todas las cláusulas que se pueden derivar de la aplicación de la regla de resolución exclusivamente sobre cláusulas que aparecen en C, entonces el tamaño de $d(C)$ puede crecer de manera *exponencial* conforme aumenta el número de cláusulas de C. Veamos un ejemplo simple de esto.

Ejemplo 1. Supongamos que N es un conjunto satisfacible de cláusulas formado por:

$$\begin{aligned} C_1 &= \{\neg P(x), \neg S(x)\} \\ C_2 &= \{\neg P(x), S(x)\} \\ C_3 &= \{P(x), Q(x), \neg R(x)\} \\ C_4 &= \{\neg Q(x), S(x)\} \end{aligned}$$

El siguiente es parte del conjunto de cláusulas que el demostrador debería generar a partir de N:

$$\begin{array}{ll} C_5 \equiv \{\neg P(x)\} & (C_2, C_1 \text{usando (RES)}) \\ C_6 \equiv \{Q(x), \neg R(x), \neg S(x)\} & (C_3, C_1 \quad " \quad) \\ C_7 \equiv \{Q(x), \neg R(x), S(x)\} & (C_3, C_2 \quad " \quad) \\ C_8 \equiv \{\neg P(x), \neg Q(x)\} & (C_4, C_1 \quad " \quad) \\ C_9 \equiv \{P(x), \neg R(x), S(x)\} & (C_3, C_4 \quad " \quad) \\ C_{10} \equiv \{Q(x), \neg R(x)\} & (C_5, C_3 \quad " \quad) \\ C_{11} \equiv \{\neg P(x), Q(x), \neg R(x)\} & (C_2, C_6 \quad " \quad) \\ C_{12} \equiv \{\neg R(x), S(x), \neg S(x)\} & (C_6, C_4 \quad " \quad) \\ C_{13} \equiv \{P(x), \neg Q(x), \neg R(x)\} & (C_4, C_6 \quad " \quad) \\ C_{14} \equiv C_{11} & (C_7, C_1 \quad " \quad) \\ C_{15} \equiv C_{12} & (C_7, C_4 \quad " \quad) \end{array}$$

$C_{16} \equiv C_{10}$ $(C_7, C_6$ " $)$
$C_{17} \equiv C_{13}$ $(C_3, C_8$ " $)$
$C_{18} \equiv \{P(x), \neg P(x), \neg R(x)\}$ $(C_3, C_8$ " $)$
$C_{19} \equiv \{\neg P(x), \neg R(x), \neg S(x)\}$ $(C_6, C_8$ " $)$
$C_{20} \equiv \{\neg P(x), \neg R(x), S(x)\}$ $(C_7, C_8$ " $)$
$\vdots$

Notemos que C_3 *resuelve* con C_1 eliminando $Q(x)$ para generar C_4 y con C_2 para generar C_5 eliminando $P(x)$. Lo que se debe observar es que esta última cláusula es *redundante*: el objetivo de un demostrador basado en resolución es lograr derivar la cláusula vacía; si suponemos que C_3 puede participar en una derivación de la cláusula vacía, concluiremos que debemos ser capaces de *eliminar* todos los literales que aparecen en C_3. Ahora bien, en C_4 ya habíamos logrado eliminar $Q(x)$, con lo cual convendría intentar eliminar $P(x)$ de esta cláusula como se hace al generar C_7; generar C_5 (ó C_6) resulta redundante. Se puede ver con claridad en este ejemplo cómo las cláusulas redundantes pueden contribuir al rápido crecimiento del conjunto de cláusulas que el demostrador debe manejar.

El mecanismo estándar para controlar la generación de cláusulas en resolución para lógica de primer orden es el llamado *resolución con orden y selección* [6]. La idea general es establecer condiciones bajo las cuáles sea seguro *elegir* un literal de cada cláusula de forma tal que se acepte que la cláusula sea premisa de un paso de inferencia sólo si es para *eliminar* dicho literal.

En el contexto de resolución para lógica de primer orden, decimos que una *función de selección* es una función que asigna a cada cláusula C un conjunto vacío o un conjunto unitario con un literal negativo de C. Dada una función de selección S y cierta relación de orden entre literales $\succ$, el cálculo de resolución para lógica de primer orden con orden y selección se define mediante la regla

$$\text{(RES–OS)} \quad \frac{C_1 \cup \{\varphi\} \quad C_2 \cup \{\neg\psi\}}{(C_1 \cup C_2)\sigma}$$

tal que

1. σ es el unificador más general de los átomos φ y ψ
2. $S(C_1 \cup \{\varphi\}) = \{\}$ y $\varphi\sigma \succ \varphi'$ para todo $\varphi' \in C_1\sigma$
3. $S(C_2 \cup \{\neg\psi\}) = \{\neg\psi\}$ o bien $S(C_2 \cup \{\neg\psi\}) = \{\}$ y $\neg\psi\sigma \succ \psi'$ para todo $\psi' \in C_2\sigma$

Se puede demostrar que el cálculo de resolución con orden y selección (que incluye la regla de factorización) es refutacionalmente completo para la lógica de primer orden cuando se usa un orden $\succ$ con las propiedades adecuadas [6].

Ejemplo 2. A modo de ejemplo, veamos en cuántos pasos llegamos a un conjunto saturado partiendo del mismo conjunto inicial del Ejemplo 1, suponiendo $\neg S(x) \succ S(x) \succ \neg R(x) \succ R(x) \succ \neg Q(x) \succ Q(x) \succ \neg P(x) \succ P(x)$ y una función de selección tal que $S(C) = \{\}$ para todo C. En todos los casos, la fórmula maximal de la cláusula es la que aparece más a la izquierda.

$C'_5 = \{\neg P(x)\}$ $(C_2, C_1$ usando (RES))
$C'_6 = \{\neg P(x), \neg Q(x)\}$ $(C_4, C_1$ usando (RES))

Ninguna otra cláusula puede generarse y el conjunto está saturado.

El Ejemplo 2 muestra en forma elocuente la importancia de contar con un mecanismo de regulación basado en orden y selección si se desea construir un demostrador basado en algún cálculo de resolución. El cálculo de resolución híbrida presentado en la Figura 1 no escapa a este problema. Sin embargo, hasta el momento no se contaba con un mecanismo apropiado de orden y selección para el mismo.

En la Sección 2 proponemos una forma de incorporar orden y selección al cálculo de resolución híbrida y demostramos que con esta extensión se conserva la completitud refutacional.

2. Resolución híbrida con orden y selección

En esta sección veremos cómo logramos incorporar al cálculo de la Figura 1 estrategias de orden y selección como las presentadas en la Sección 1.2.3. Comenzaremos, por dar un tipo especial de orden para fórmulas de $\mathcal{H}^{NNF}(@)$ que nos permite garantizar que el cálculo de resolución con orden y selección que luego proponemos es refutacionalmente completo. La demostración de este hecho es similar, aunque más compleja, a la estándar para el caso de resolución para lógica de primer de orden [6]. Esta demostración requiere una noción adecuada de *modelo de Herbrand*, que presentaremos, previamente, en la Sección 2.3.

2.1. Relaciones de orden entre fórmulas

Se llama *orden* a cualquier relación transitiva e irreflexiva, y se dice que un orden $\succ$ es *total* cuando para cualquier par de elementos x e y, si $x \neq y$, entonces $x \succ y$ o $y \succ x$. Un orden $\succ$ está *bien fundado* cuando no es posible dar con una cadena infinita $x_1 \succ x_2 \succ x_3 \ldots$. Algunos órdenes entre fórmulas son particularmente importantes:

Definición 2.1. Si con $\varphi[\psi]_p$ indicamos que la fórmula ψ es una subfórmula de φ que aparece en posicion p, entonces podemos decir que un orden $\succ$ entre fórmulas tiene la *propiedad de subfórmulas* si $\varphi[\psi]_p \succ \psi$ cuando $\varphi \neq \psi$, y que es un *orden de reescritura* si $\varphi[\psi_1]_p \succ \varphi[\psi_2]_p$ sii $\psi_1 \succ \psi_2$. Un *orden de reducción* es un orden de reescritura bien fundado; y si además tiene la propiedad de subfórmulas se lo llama *orden de simplificación*.

Un método estándar para construir órdenes con estas propiedades es usando el llamado *lexicographic path ordering* (lpo).

Definición 2.2 (Lexicographic Path Ordering). Dado un álfabeto Σ donde cada símbolo tiene asociada una aridad, y dado un orden $\succ$ entre los elementos de Σ al que llamaremos *precedencia*, se define $\succ_{lpo}$, el *lexicographic path ordering* (lpo) entre términos de Σ^* como sigue.

Sean $\varphi = o_1(\varphi_1 \ldots \varphi_m)$ y $\psi = o_2(\psi_1 \ldots \psi_n)$, $n, m \geq 0$ dos términos de Σ^*; diremos que $\varphi \succ_{lpo} \psi$ si y sólo si:

1. $o_1 \succ o_2$ y $\varphi \succ_{lpo} \psi_i$, para todo i tal que $1 \leq i \leq n$; ó
2. $o_1 = o_2$ y para algún j se cumple, $(\varphi_1 \ldots \varphi_{j-1}) = (\psi_1 \ldots \psi_{j-1})$, $\varphi_j \succ_{lpo} \psi_j$ y $\varphi \succ_{lpo} \psi_k$ para todo k tal que $j \leq k \leq n$; ó
3. $\varphi_j \succeq_{lpo} \psi$ para algún j tal que $1 \leq j \leq m$.

Si la precedencia $\succ$ es bien fundada (total), entonces $\succ_{lpo}$ es un orden de simplificación (total) (ver, por ejemplo, [8]). La siguiente es una forma de construir, dado un orden total y bien fundado entre fórmulas, un orden total y bien fundado entre cláusulas (conjuntos finitos de fórmulas):

Definición 2.3. Dado $\succ$ un orden entre fórmulas, se define $\succ_{Cl}$, su extensión para cláusulas, como la relación que cumple: $C_1 \succ_{Cl} C_2$ si y sólo si $C_1 \not\subseteq C_2$ y para toda φ_2 tal que $\varphi_2 \in C_2$ y $\varphi_2 \notin C_1$ existe $\varphi_1 \in C_1$ tal que $\varphi_1 \notin C_2$ y $\varphi_1 \succ \varphi_2$.

De aquí en más, salvo que se indique lo contrario, usaremos el mismo símbolo para una relación de orden entre fórmulas y la extensión para cláusulas de dicho orden.

2.2. Un orden *admisible* para $\mathcal{H}(@)$

En el contexto de los sistemas de resolución se llama *admisible* a un orden entre fórmulas que garantiza que un cálculo con orden y selección conserva la completitud refutacional. En esta sección presentaremos un tipo de orden sobre $\mathcal{H}(@)$ que nos permite definir un cálculo completo con orden y selección basado en el cálculo híbrido. De aquí en más, salvo que se aclare lo contrario, trabajaremos únicamente con

fórmulas bien formadas de $\mathcal{H}^{NNF}(@)$. En este contexto, dado que definiremos un orden basado en lpo, nos resultará conveniente considerar a $@$, $\langle\rangle$ y $[]$ como operadores *binarios*: $@(\cdot,\cdot) : \mathcal{H}^{NNF}(@) \times \mathsf{NOM} \to \mathcal{H}^{NNF}(@)$[2], $\langle\rangle(\cdot,\cdot) : \mathsf{REL} \times \mathcal{H}^{NNF}(@) \to \mathcal{H}^{NNF}(@)$, $[](\cdot,\cdot) : \mathsf{REL} \times \mathcal{H}^{NNF}(@) \to \mathcal{H}^{NNF}(@)$, por más que usemos la notación estándar $@_n\,\varphi$, $\langle r\rangle\varphi$ y $[r]\varphi$.

Definición 2.4. Un orden $\succ$ es admisible si cumple simultáneamente con estas condiciones, para todo $\varphi, \psi \in \mathcal{H}^{NNF}(@)$:

- A1) es un orden de simplificación total
- A2) $\varphi \succ i$ para todo $\varphi \notin \mathsf{NOM}$ y todo $i \in \mathsf{NOM}$
- A3) si $\varphi \succ \psi$, entonces $@_i\,\varphi \succ @_j\,\psi$, para todo $i, j \in \mathsf{NOM}$
- A4) si $\langle r\rangle i$ es una subfórmula propia de φ con $i \in \mathsf{NOM}$, entonces $\varphi \succ \langle r\rangle j$ para todo nominal j
- A5) $[r]i \succ \langle r\rangle j$, para todo $i, j \in \mathsf{NOM}$

La condición A2 pide que un nominal sea menor que cualquier fórmula que no sea otro nominal, A3 pide que el operador $@$ no afecte al orden entre fórmulas, A4 introduce una noción muy débil de *complejidad estructural* y A5 prioriza a $[]$ por sobre $\langle\rangle$.

Es importante observar que si $\succ$ cumple con las condiciones de la Definición 2.4, entonces $i \succ j$ si y sólo si $@_j\,i \succ @_i\,j$ para todo par de nominales i y j. Esto significa que un orden admisible ordena en forma coherente las igualdades.

Debemos mostrar que las condiciones de la Definición 2.4 no son demasiado restrictivas. Para ello construimos un orden concreto que cumple con las condiciones pedidas. Nos basamos en el orden presentado en la Definición 2.2, aplicado sobre el alfabeto $\Sigma = \mathsf{PROP} \cup \mathsf{NOM} \cup \mathsf{REL} \cup \{\neg, \wedge, \vee, @, \langle\rangle, []\}$ (notar que $\mathcal{H}^{NNF}(@) \subset \Sigma^*$).

Definición 2.5. Dada una signatura $\langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL}\rangle$ con $\mathsf{PROP} = \{P_i \mid i \in \mathbb{N}\}$, $\mathsf{NOM} = \{N_i \mid i \in \mathbb{N}\}$ y $\mathsf{REL} = \{R_i \mid i \in \mathbb{N}\}$, definimos la relación $> \subseteq \Sigma \times \Sigma$ como la clausura transitiva del conjunto:

$$\begin{array}{ll} \{(@, \neg), (\neg, \wedge), (\wedge, \vee), (\vee, []), ([], \langle\rangle)\} & \bigcup \\ \{(\langle\rangle, R_i), (R_i, P_j), (P_j, N_k) \mid i, j, k \in \mathbb{N}\} & \bigcup \\ \{(R_i, R_j), (P_i, P_j), (N_i, N_j) \mid i > j\} & \end{array}$$

Por definición, $>$ es total, irreflexiva y está bien fundada. Sea $\succ_{lpo}$ el orden lexicográfico sobre Σ^* que usa $>$ como precedencia. De acuerdo a lo que hemos visto, $\succ_{lpo}$ es un orden de simplificación total. Finalmente, definimos $\succ_h$ como el orden que verifica $\varphi \succ_h \psi$ si y sólo si $\mathit{size}(\varphi) > \mathit{size}(\psi)$, ó $\mathit{size}(\varphi) = \mathit{size}(\psi)$ y $\varphi \succ_{lpo} \psi$, donde $\mathit{size}(\varphi)$ es la complejidad (en cuanto a cantidad de operadores) de φ.

Proposición 1. *$\succ_h$ es un orden admisible.*

Es interesante observar que no es posible construir un orden admisible usando *únicamente* lpo. Basta con ver que no se puede encontrar una manera de garantizar $\langle\rangle(r', \langle\rangle(r, i)) \succ_{lpo} \langle\rangle(r, j)$ cuando $r \succ_{lpo} r'$ y $j \succ_{lpo} i$, lo cual viola A4.

En lo que resta, usaremos $\succ$ para referirnos a algún orden admisible salvo que se indique específicamente lo contrario.

2.3. Un Teorema de Herbrand para $\mathcal{H}(@)$

Como ya mencionamos, la demostración estándar de completitud refutacional del cálculo de resolución para lógica de primer orden utiliza modelos de Herbrand. Como parte de la adaptación de esta

[2]El orden de los parámetros de este operador ha sido cuidadosamente elegido para simplificar la Definición 2.5 y la subsecuente demostración de la Proposición 1.

demostración al caso $\mathcal{H}(@)$, fue necesario previamente dar con un resultado análogo al Teorema de Herbrand para esta lógica. En esta sección comenzamos por repasar el Teorema de Herbrand clásico para luego presentar este resultado para el caso $\mathcal{H}(@)$.

Dada una signatura de primer orden (sin igualdad) $\mathcal{S} = \langle \mathsf{FUNC}, \mathsf{REL} \rangle$, un *modelo de Herbrand* [11] para $\mathcal{S}$ se define como la interpretación $I = \langle D, \cdot^I \rangle$ donde D es el conjunto de términos de $\mathcal{S}$, $t^I = t$ para todo término t, y $R^I \subseteq D^n$ para todo símbolo de relación R de aridad n.

Es decir, todos los modelos de Herbrand tienen el mismo dominio e interpretan los términos de la misma manera; la única variación está en la forma en que interpretan los símbolos de REL. Notar que podemos identificar un modelo de Herbrand con el conjunto de literales positivos que son verdaderos en el modelo. La importancia de los modelos de Herbrand está dada por el siguiente teorema:

Teorema 2 (Teorema de Herbrand). *Una teoría de primer orden T sobre la signatura $\mathcal{S} = \langle \mathsf{FUNC}, \mathsf{REL} \rangle$ tiene un modelo si y sólo si tiene un modelo de Herbrand sobre la signatura $\mathcal{S}' = \langle \mathsf{FUNC} \cup \mathsf{FUNC}', \mathsf{REL} \rangle$, donde FUNC' es un conjunto infinito numerable disjunto de FUNC.*

El Teorema 2 asegura que para determinar la satisfacibilidad de una teoría de primer orden, alcanza con considerar solamente los posibles modelos de Herbrand (formulaciones alternativas de este teorema en [7]). Para extender este teorema a la lógica $\mathcal{H}(@)$ empezamos por definir la noción apropiada de literal positivo.

Definición 2.6. Dada una signatura híbrida $\mathcal{S} = \langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$, definimos $\mathsf{LIT\text{-}P}$, el conjunto de *literales positivos* de $\mathcal{H}(@)$ sobre $\mathcal{S}$, como aquel formado por las fórmulas de la forma $@_i\, j$, $@_i\, p$, y $@_i\, \langle r \rangle j$, donde $i, j \in \mathsf{NOM}$, $p \in \mathsf{PROP}$ y $r \in \mathsf{REL}$.

Definición 2.7. Sea $\mathcal{S} = \langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$ una signatura híbrida dada. Un *modelo de Herbrand híbrido* para $\mathcal{H}(@)$ en $\mathcal{S}$ es un conjunto $I \subset \mathsf{LIT\text{-}P}$.

Como anteriormente, identificamos un modelo de Herbrand con un conjunto de literales positivos que el modelo satisface y que define unívocamente un determinado modelo híbrido. Dado I un modelo de Herbrand, sea $\sim_I$ (la *relación de equivalencia inducida por* I) la mínima relación de equivalencia que extiende el conjunto $\{(i,j) \mid @_i\, j \in I\} \cup \{(i,i) \mid i \in \mathsf{NOM}\}$. Definimos ahora el modelo híbrido unívocamente determinado por I como la estructura $\langle W^I, \{r_i^I\}, V^I \rangle$, donde

$$\begin{array}{lcl} W^I & = & \mathsf{NOM}/_{\sim_I} \\ r_i^I & = & \{([j],[k]) \mid @_j\, \langle r_i \rangle k \in I\} \\ V^I(p) & = & \{[j] \mid @_j\, p \in I\}, p \in \mathsf{PROP} \\ V^I(i) & = & \{[i]\}, i \in \mathsf{NOM}. \end{array}$$

donde $\mathsf{NOM}/_{\sim_I}$ es el conjunto de clases de equivalencia definido por $\sim_I$ sobre NOM y $[i]$ es la clase de equivalencia asociada a i por $\sim_I$. De ahora en más no diferenciaremos entre un modelo de Herbrand híbrido I y su modelo asociado; y diremos, por ejemplo, que una fórmula $@_i\varphi$ es válida en I, cuando el modelo asociado la satisfaga (siempre nos referiremos a fórmulas de la forma $@_i\varphi$ para que no sea necesaria la referencia a un punto de evaluación en el modelo).

El modelo asociado a I es más complejo que el que definimos para una signatura de primer orden porque $\mathcal{H}(@)$ contiene igualdad y necesitamos entonces particionar el dominio bajo la relación de identidad definida por I. Podemos ahora enunciar el equivalente al Teorema 2.

Teorema 3. *Dado T, un conjunto de fórmulas-@ en $\mathcal{H}(@)$ en la signatura $\mathcal{S} = \langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$, T tiene un modelo híbrido si y sólo si tiene un modelo de Herbrand híbrido sobre la signatura $\mathcal{S}' = \langle \mathsf{PROP}, \mathsf{NOM} \cup \mathsf{NOM}', \mathsf{REL} \rangle$ donde NOM' es disjunto de NOM, infinito y numerable.*

2.4. Resolución para $\mathcal{H}^{NNF}(@)$ con orden y selección

En resolución para lógica de primer orden, una función de selección puede elegir una fórmula de una cláusula, con la condición de que sea un literal negativo. Si bien en $\mathcal{H}^{NNF}(@)$ no contamos con una noción de literal que sea directamente aplicable, utilizaremos el complemento de la noción de literales positivos de la Definición 2.7.

Definición 2.8. Diremos que S es una *función de selección* si y sólo si, para cualquier cláusula C se cumple $S(C) \subseteq C$, $|S(C)| \leq 1$ y $S(C) \cap \mathsf{LIT\text{-}P} = \emptyset$

Lo que esta definición nos dice es que una función de selección elige *a lo sumo* una fórmula de cada cláusula, y que aquello que seleccione no puede ser un literal positivo. La Figura 2 contiene las reglas para el cálculo. En ella se asume que $S(C)$ es una función de selección y que $\succ$ es un orden admisible (ver Definición 2.4). La premisa principal de cada regla es siempre la que aparece más a la derecha.

$$(\wedge)\quad \frac{Cl \cup \{@_t\,(\varphi_1 \wedge \varphi_2)\}}{\begin{array}{c} Cl \cup \{@_t\,\varphi_1\} \\ Cl \cup \{@_t\,\varphi_2\} \end{array}} \qquad (\vee)\quad \frac{Cl \cup \{@_t\,(\varphi_1 \vee \varphi_2)\}}{Cl \cup \{@_t\,\varphi_1, @_t\,\varphi_2\}}$$

$$(\text{RES})\quad \frac{Cl_1 \cup \{@_t\,\varphi\} \quad Cl_2 \cup \{@_t\,\neg\varphi\}}{Cl_1 \cup Cl_2}$$

$$([r])\quad \frac{Cl_1 \cup \{@_t\,\langle r\rangle s\} \quad Cl_2 \cup \{@_t\,[r]\varphi\}}{Cl_1 \cup Cl_2 \cup \{@_s\,\varphi\}} \qquad (\langle r\rangle)\quad \frac{Cl \cup \{@_t\,\langle r\rangle\varphi\}}{\begin{array}{c} Cl \cup \{@_t\,\langle r\rangle n\} \\ Cl \cup \{@_n\,\varphi\} \end{array}} \quad \begin{array}{l} \text{para un } n \text{ nuevo} \\ \text{y } \varphi \notin \mathsf{NOM} \end{array}$$

$$(@)\quad \frac{Cl \cup \{@_t\,@_s\,\varphi\}}{Cl \cup \{@_s\,\varphi\}} \qquad (\text{REF})\quad \frac{Cl \cup \{@_t\,\neg t\}}{Cl}$$

$$(\text{SYM})\quad \frac{Cl \cup \{@_s\,t\}}{Cl \cup \{@_t\,s\}}\ \text{si } t \succ s \qquad (\text{PARAM})\quad \frac{Cl_1 \cup \{@_s\,t\} \quad Cl_2 \cup \{\varphi(s)\}}{Cl_1 \cup Cl_2 \cup \{\varphi(s/t)\}}\ \begin{array}{l} \text{si } s \succ t \text{ y} \\ \varphi(s) \succ @_s\,t \end{array}$$

Restricciones:

- Si $C = C' \cup \{\varphi\}$ es la premisa principal, entonces o bien $S(C) = \{\varphi\}$ o, en caso contrario, $S(C) = \emptyset$ y $\{\varphi\} \succ C'$
- Si $D = D' \cup \{\psi\}$ es la premisa auxiliar, entonces $\{\psi\} \succ D'$ y $S(D) = \emptyset$

(tanto φ como ψ representan la fórmula que participa en la inferencia)

Figura 2: Reglas de resolución para $\mathcal{H}^{NNF}(@)$ con orden y selección

Como se puede ver, la Figura 2 difiere de la Figura 1 sólo en el agregado de algunas restricciones tanto locales (reglas ($\langle r\rangle$), (SYM) y (PARAM)) como generales. De acuerdo a estas últimas restricciones, en cada cláusula existe siempre una única fórmula que puede participar en alguna inferencia. Nos referiremos a ella como la *fórmula distinguida* (para $\succ$ y S) de una cláusula y la notaremos $dist^{\succ}_{S}(C)$.

Definición 2.9. Dados un orden admisible $\succ$ y una función de selección S, definimos $max^{\succ}(C)$ como la fórmula maximal (respecto a $\succ$) de C, y $dist^{\succ}_{S}(C)$ como la función tal que $dist^{\succ}_{S}(C) = \varphi$ si $S(C) = \{\varphi\}$ o si $S(C) = \emptyset$ y $max^{\succ}(C) = \varphi$.

Se demuestra que un orden admisible garantiza que toda cláusula producto de una inferencia es menor que la cláusula principal que se utilizó en la aplicación de la regla.

Proposición 4. *Si C es la premisa principal de una inferencia que tiene a D como uno de sus consecuentes, entonces $C \succ D$.*

La de demostración de completitud refutacional está basada en la que se da en [6] para el caso de resolución para lógica de primer orden. Esencialmente, damos un mecanismo para construir un modelo de Herbrand a partir de un conjunto arbitrario (y potencialmente infinito) de cláusulas, de forma tal que si la menor de las cláusulas no es verdadera en este modelo, entonces el cálculo permita derivar una nueva cláusula, menor que la anterior, y que tampoco sea satisfecha por el modelo.

De aquí podremos concluir que en un conjunto clausurado respecto a la aplicación de las reglas, o bien este mecanismo nos provee un modelo que lo satisface, o bien, usando la Proposición 4, la cláusula vacía pertenece a este conjunto. Por esta razón, a estos modelos los llamamos *modelos tentativos*.

La definición de modelo tentativo que damos a continuación es más compleja que la utilizada en [6] ya que aquella estaba dada para el caso de lógica de primer orden sin igualdad, mientras que en $\mathcal{H}(@)$ tenemos que tomar en cuenta las igualdades de la forma $@_i\, j$ (con i y j nominales).

Definición 2.10. Dada I, una interpretación de Herbrand híbrida, definimos la sustitución de nominales por nominales $\sigma_I = \{i \mapsto j | i \sim_I j \wedge (\forall k)(k \sim_I j \rightarrow k \succeq j)\}$, que sustituye cada nominal por el menor nominal de su clase, el cual se toma como representante. Donde no hay ambigüedades, usaremos σ en lugar de σ_I.

Definición 2.11. Dada una signatura híbrida $\mathcal{S} = \langle \mathsf{PROP}, \mathsf{NOM}, \mathsf{REL} \rangle$, definimos SIMP, el conjunto de *fórmulas simples* de $\mathcal{H}^{NNF}(@)$ sobre $\mathcal{S}$ (i.e. las que no pueden ser simplificadas por las reglas ($\wedge$), ($\vee$), ($\langle r \rangle$) o (SYM)) como aquel formado por las fórmulas de la forma $@_i\, j$ (con $i \succ j$), $@_i\, p$, $@_i \neg a$, $@_i \langle r \rangle j$ y $@_i [r]\varphi$, donde $i, j \in \mathsf{NOM}$, $p \in \mathsf{PROP}$, $a \in \mathsf{NOM} \cup \mathsf{PROP}$, $r \in \mathsf{REL}$ y $\varphi \in \mathcal{H}^{NNF}(@)$.

Supongamos un conjunto fijo de cláusulas N. Las siguientes tres definiciones deben tomarse como una unidad. Se presentan en forma separada por claridad pero, como se verá, son las tres mutuamente recursivas.

Definición 2.12 (I_C). Sea C una cláusula (no necesariamente perteneciente a N), llamaremos I_C a la interpretación de Herbrand híbrida dada por $\bigcup_{C \succ D} \varepsilon_D$

Definición 2.13 (Forma reducida). Sean C una cláusula y φ su fórmula maximal. Si $\varphi \in \mathsf{SIMP}$ y se cumple que o bien $\varphi \in \mathsf{LIT\text{-}P}$ y $\varphi = \varphi\sigma_{I_C}$, o bien $\varphi = @_i [r]\psi$ y $i = i\sigma_{I_C}$, entonces decimos que tanto φ como C se encuentran en *forma reducida*.

Definición 2.14 (ε_C). Sea C una cláusula (no necesariamente de N); si se verifica simultáneamente:

1. $C \in N$
2. C está en forma reducida
3. $max^{\succ}(C) \in \mathsf{LIT\text{-}P}$
4. C es falsa en I_C
5. $S(C) = \emptyset$

entonces $\varepsilon_C = \{\varphi\}$; en caso contrario, ε_C es el conjunto vacío.

Decimos que C *produce* φ si $\varepsilon_C = \{\varphi\}$ y la llamaremos una *cláusula productiva*. I_C es la *interpretación parcial de N por debajo de C*. Sólo las cláusulas cuya fórmula maximal φ sea un literal positivo y no tengan fórmulas seleccionadas pueden ser productivas.

Definición 2.15. I_N, un modelo tentativo de N, se define como $\bigcup_{C \in N} \varepsilon_C$.

Si una cláusula C es falsa en I, decimos que C es un *contraejemplo* de I. Analizando cada regla del cálculo y considerando por separado las fórmulas distinguidas de una cláusula que no se encuentran en forma reducida, se obtiene el siguiente resultado.

Teorema 5. *Sea N un conjunto de cláusulas y sea C el contraejemplo mínimo de I_N, respecto de un orden admisible $\succ$. Si $C \neq \{\}$, entonces existe una inferencia usando alguna de las reglas del cálculo tal que:*

1. *C es la premisa principal*
2. *la premisa auxiliar (en los casos que corresponde) es una cláusula productiva*
3. *todos los consecuentes son menores, respecto de $\succ$, que C y al menos uno es un contraejemplo de I_N.*

Usando este teorema, podemos probar la completitud del cálculo.

Teorema 6. *El cálculo de resolución híbrido con orden y selección es refutacionalmente completo.*

3. Terminación del cálculo

El cálculo de resolución que hemos visto constituye, dado el Teorema 6, un método correcto y completo para resolver el problema de determinar si cierta fórmula de $\mathcal{H}(@)$ es satisfacible. En esta sección mostramos cómo convertir este cálculo en un *método de decisión* para dicho problema. Se dice que un método es de *decisión* para el problema P si para *cualquier* instancia de P podemos obtener una respuesta *correcta* acerca de P en un *número finito de pasos*. La consistencia y completitud refutacional del cálculo garantizan, respectivamente, que la respuesta sea *correcta* para *cualquier* instancia del problema. Lo que nos interesa ver, entonces, es cómo garantizar que ésta se obtenga en un *número finito de pasos*.

Lo que buscamos, concretamente, es lograr que para cualquier fórmula $\varphi \in \mathcal{H}(@)$, $\mathit{ClSet}^*(\varphi)$ sea un conjunto *finito*. Cuando esto se cumple, es fácil implementar un algoritmo que compute en tiempo finito $\mathit{ClSet}^*(\varphi)$ (la forma estándar es usando el "algoritmo de la cláusula dada" [20]).

El cálculo de la Figura 1 claramente puede generar un conjunto saturado de fórmulas que sea infinito. Basta con observar que la regla ($\langle r \rangle$) se aplica aun sobre fórmulas de la forma $@_i \langle r \rangle j$ donde j es un nominal. El cálculo con orden y selección que presentamos en la Figura 2, en cambio, evita la aplicación de ($\langle r \rangle$) a fórmulas de este estilo y, sin embargo, como mostramos en la Sección 3.1, también con este cálculo es posible generar un conjunto saturado infinito. Luego proponemos una variación del cálculo de la Figura 2 que preserva completitud con el cual podemos garantizar terminación. En todo momento, asumiremos dado un orden admisible $\succ$ (según la Definición 2.4).

3.1. Cómo generar infinitas fórmulas

Ejemplo 3. Tomemos la fórmula satisfacible $@_i([r](i \wedge \langle r \rangle p) \wedge \langle r \rangle p)$ y asumamos que i es el menor nominal de un orden admisible. Veamos qué sucede cuando aplicamos las reglas del cálculo sobre esta fórmula:

$$
\begin{array}{rll}
C = \{@_i([r](i \wedge \langle r \rangle p) \wedge \langle r \rangle p)\ \} & & \\
D_{1a} = \{@_i [r](i \wedge \langle r \rangle p)\} & (C, \text{ usando } (\wedge)) \\
D_{1b} = \{@_i \langle r \rangle p\} & & \\
D_{2a} = \{@_i \langle r \rangle k\} & (D_{1b}, \text{ usando } (\langle r \rangle)) \\
D_{2b} = \{@_k\, p\} & & \\
D_3 = \{@_k (i \wedge \langle r \rangle p)\} & (D_{1a} \text{ y } D_{2a}, \text{ usando } ([r])) \\
D_{4a} = \{@_k\, i\} & (D_3, \text{ usando } (\wedge)) \\
D_{4b} = \{@_k \langle r \rangle p\} & & \\
D_{5a} = \{@_k \langle r \rangle k_2\} & (D_{4b}, \text{ usando } (\langle\rangle)) \\
D_{5b} = \{@_{k_2}\, p\} & & \\
D_6 = \{@_i \langle r \rangle k_2\} & (D_{4a} \text{ y } D_{5a}, \text{ usando (PARAM)}) \\
\vdots & &
\end{array}
$$

Si comparamos D_{2a} y D_6 veremos que en esta derivación son cláusulas equivalentes. Es decir, podemos

repetir los pasos llevados a cabo para derivar D_6 a partir de D_{1a} y D_{2a} para generar $D_{10} = \{@_i \langle r \rangle k_3\}$, y a partir de D_{10} derivar $D_{14} = \{@_i \langle r \rangle k_4\}$ y así sucesivamente. Claramente, la saturación del conjunto inicial $\{C\}$ constituye un conjunto infinito.

Veamos más en detalle la fórmula inicial del Ejemplo 3. Para empezar, es satisfacible sólo en modelos donde p es verdadero en el elemento asociado a i, y éste esté relacionado con sí mismo y sólo con sí mismo. En segundo lugar, esta fórmula tiene dos niveles de profundidad modal, esto significa, intuitivamente, que está *predicando* sobre elementos que están a no más de dos "pasos de distancia" de i.

Por otro lado, observemos qué sucede con k, k_2, k_3, etc. k_2 es un nominal que se introduce a partir de $@_k \langle r \rangle p$, *bajo la hipótesis de que i y k son distintos*. Con esto nos referimos a que, dado que sobre D_{4b} es posible aplicar paramodulación con D_{4a}, podemos pensar que a partir de este hecho se abren dos ramas: aquella en que aplicamos paramodulación y aquella en que no. En la rama en que lo hacemos, estamos asumiendo que i y k están asociados al mismo elemento del dominio; en la rama en que no, estamos asumiendo que son distintos. Este último es el caso que estamos considerando. Ahora bien, k está a un paso de distancia de i, y si son distintos, k_2 debe estar a dos pasos de distancia de i. Sin embargo, en D_6 al reemplazar k por i estamos *anulando* la hipótesis que dio origen a k_2. Es decir, k_2 es un nominal que se crea con la idea (equivocada) de que esté a dos pasos de distancia de i.

En definitiva, hemos visto un ejemplo en el que se obtiene un conjunto $\mathit{ClSet}^*(\varphi)$ infinito mediante la generación de sucesivos nominales nuevos cada vez más *alejados* de aquellos presentes en φ. El siguiente ejemplo nos muestra que también podemos obtener un número infinito de nominales sin necesidad de que éstos estén cada vez más lejos.

Ejemplo 4. Consideremos la siguiente variación de la fórmula del Ejemplo 3: $@_i([r](i \wedge (q \vee \langle r \rangle p)) \wedge \langle r \rangle p)$. Ésta también es satisfacible y permite generar la siguiente derivación:

$$
\begin{array}{rll}
C = \{@_i([r](i \wedge (q \vee \langle r \rangle p)) \wedge \langle r \rangle p)\} & & \\
D_{1a} = \{@_i [r](i \wedge (q \vee \langle r \rangle p))\} & (C, \text{ usando } (\wedge)) \\
D_{1b} = \{@_i \langle r \rangle p\} & & \\
D_{2a} = \{@_i \langle r \rangle k\} & (D_{1b}, \text{ usando } (\langle r \rangle)) \\
D_{2b} = \{@_k p\} & & \\
D_3 = \{@_k (i \wedge (q \vee \langle r \rangle p))\} & (D_{1a} \text{ y } D_{2a}, \text{ usando } ([r])) \\
D_{4a} = \{@_k i\} & (D_3, \text{ usando } (\wedge)) \\
D_{4b} = \{@_k (q \vee \langle r \rangle p)\} & & \\
D_5 = \{@_k q, @_k \langle r \rangle p\} & (D_{4b}, \text{ usando } (\vee)) \\
D_6 = \{@_k q, @_i \langle r \rangle p\} & (D_{4a} \text{ y } D_5, \text{ usando (PARAM)}) \\
D_{7a} = \{@_k q, @_i \langle r \rangle k_2\} & (D_6, \text{ usando } (\langle r \rangle)) \\
D_{7b} = \{@_k q, @_{k_2} p\} & & \\
D_8 = \{@_k q, @_{k_2} (i \wedge (q \vee \langle r \rangle p))\} & (D_{1a} \text{ y } D_{7a}, \text{ usando } ([r])) \\
D_{9a} = \{@_k q, @_{k_2} i\} & (D_8, \text{ usando } (\wedge)) \\
D_{9b} = \{@_k q, @_{k_2} (q \vee \langle r \rangle p)\} & & \\
D_{10} = \{@_k q, @_{k_2} q, @_{k_2} \langle r \rangle p\} & (D_{9b}, \text{ usando } (\vee)) \\
D_{11} = \{@_k q, @_{k_2} q, @_i \langle r \rangle p\} & (D_{9a} \text{ y } D_{10}, \text{ usando (PARAM)}) \\
D_{12a} = \{@_k q, @_{k_2} q, @_i \langle r \rangle k_3\} & (D_{11}, \text{ usando } (\langle r \rangle)) \\
D_{12b} = \{@_k q, @_{k_2} q, @_{k_3} p\} & & \\
\vdots & &
\end{array}
$$

En este caso las cláusulas a comparar son D_{2a}, D_{7a} y D_{12a}, y conviene mirar cómo estas últimas se generan a partir de D_{1b}, D_6 y D_{11}.

Este ejemplo muestra cómo se pueden obtener infinitas apariciones de $@_i \langle r \rangle p$, y cómo cada una de ellas genera a su vez un nuevo nominal a un paso de distancia respecto de i.

3.2. Restringiendo la generación de nuevos nominales: la regla ($\langle r \rangle$)

En la sección anterior presentamos dos formas distintas en que el cálculo de la Figura 2 puede generar infinitos nominales. En una de ellas, se crean nuevos nominales cada vez más alejados; mientras que en la otra, infinitas apariciones de una misma fórmula generan infinitos nominales equidistantes. En esta sección presentamos una variación del cálculo que evita este último comportamiento.

Consideremos la cláusula $C = \{@_i \langle r \rangle p, @_j (p \wedge q)\}$ y supongámosla parte de un conjunto de cláusulas. Dependiendo de la función de selección que utilicemos, una derivación a partir de C comenzará simplificando $@_i \langle r \rangle p$ ó $@_j (p \wedge q)$. Veamos dos posibles derivaciones, una empezando en cada fórmula. Por un lado, tenemos:

$$\begin{array}{lll} C & = \{@_i \langle r \rangle p, @_j (p \wedge q)\} & \\ C_{1a} & = \{@_i \langle r \rangle k, @_j (p \wedge q)\} & (C, \text{ usando } (\langle r \rangle)) \\ C_{1b} & = \{@_k p, @_j (p \wedge q)\} & \\ C_{2a} & = \{@_i \langle r \rangle k, @_j p\} & (C_{1a}, \text{ usando } (\wedge)) \\ C_{2b} & = \{@_i \langle r \rangle k, @_j q\} & \\ C_{3a} & = \{@_k p, @_j p\} & (C_{1b}, \text{ usando } (\wedge)) \\ C_{3b} & = \{@_k p, @_j q\} & \end{array}$$

mientras que por el otro obtenemos:

$$\begin{array}{lll} C & = \{@_i \langle r \rangle p, @_j (p \wedge q)\} & \\ C'_{1a} & = \{@_i \langle r \rangle p, @_j p\} & (C, \text{ usando } (\wedge)) \\ C'_{1b} & = \{@_i \langle r \rangle p, @_j q\} & \\ C'_{2a} & = \{@_i \langle r \rangle k, @_j p\} & (C'_{1a}, \text{ usando } (\langle r \rangle)) \\ C'_{2b} & = \{@_k p, @_j p\} & \\ C'_{3a} & = \{@_i \langle r \rangle l, @_j q\} & (C'_{1b}, \text{ usando } (\langle r \rangle)) \\ C'_{3b} & = \{@_l p, @_j q\} & \end{array}$$

Si comparamos las últimas cuatro cláusulas de cada caso observaremos que sólo difieren en el hecho de que en la segunda derivación se generó un nominal nuevo (l) mientras que en la primera se usó en todos los casos el mismo nominal (k); i.e., la segunda derivación está considerando modelos más complicados que la primera (e.g. modelos en los que i se relaciona con más de un elemento).

De hecho, en el segundo caso estamos *perdiendo información*: las apariciones de $@_i \langle r \rangle p$ en C'_{1a} y C'_{1b} corresponden a la misma fórmula (puesto que ambas derivan de la aparición de $@_i \langle r \rangle p$ en C) y sin embargo ya no es posible relacionar las fórmulas que se derivan de ellas.

Lo que este ejemplo nos sugiere es que dada cualquier aparición de una fórmula $@_i \langle r \rangle \psi$, alcanza con tener un *único* nominal "testigo", que sea sucesor de i y que satisfaga ψ. Más formalmente, podemos demostrar lo siguiente.

Teorema 7. *Sea φ una fórmula de $\mathcal{H}^{NNF}(@)$ y sea j un nominal que no aparece en φ. φ es satisfacible si y sólo si para todo nominal i, toda relación r y toda fórmula $\psi \in \mathcal{H}^{NNF}(@)$, $\varphi[@_i \langle r \rangle \psi / (@_i \langle r \rangle j \wedge @_j \psi)]$ es satisfacible.*

Con este resultado, podemos proponer una regla alternativa para tratar diamantes. Para ello, dividamos NOM en dos conjuntos (infinitos) disjuntos NOM_i (*nominales iniciales*) y NOM_c (*nominales del cálculo*), y llamemos $\mathcal{H}_0^{NNF}(@)$ al conjunto de fórmulas de $\mathcal{H}^{NNF}(@)$ en el que sólo aparecen nominales de NOM_i. De aquí en más supondremos, sin perder generalidad, que el cálculo se utiliza sólo sobre fórmulas de $\mathcal{H}_0^{NNF}(@)$ y que todo nominal que aparece en $\mathit{ClSet}^*(\varphi)$ pero no en φ pertenece a NOM_c.

Definamos, además, $\mathcal{H}_{@_\Diamond}^{NNF}(@) = \{@_i \langle r \rangle \psi \mid i \in \mathsf{NOM}, r \in \mathsf{REL}, \psi \in \mathcal{H}_0^{NNF}(@), \psi \notin \mathsf{NOM}\}$, es decir, $\mathcal{H}_{@_\Diamond}^{NNF}(@)$ es el conjunto de fórmulas-@ de $\mathcal{H}^{NNF}(@)$ que pueden ser premisa de la regla ($\langle r \rangle$). Finalmente, sea $nom(x) : \mathcal{H}_{@_\Diamond}^{NNF}(@) \to \mathsf{NOM}_c$ una función *inyectiva* cualquiera. Definimos la regla alternativa ($\langle r \rangle'$) como:

$$(\langle r\rangle') \quad \frac{Cl \cup \{@_t \langle r\rangle \varphi\}}{\begin{array}{c} Cl \cup \{@_t \langle r\rangle n\} \\ Cl \cup \{@_n \varphi\} \end{array}} \quad \begin{array}{l} \text{con } n = nom(@_t \langle r\rangle \varphi) \\ \text{y } \varphi \notin \mathsf{NOM} \end{array}$$

El Teorema 7 garantiza la consistencia de esta regla. Además, se comprueba que la misma argumentación utilizada para la regla $(\langle r\rangle)$ en la demostración del Teorema 5 vale para la regla $(\langle r\rangle')$. Esto garantiza que al reemplazar $(\langle r\rangle)$ por esta regla se preserva la completitud refutacional del cálculo.

3.3. Restringiendo la generación de fórmulas repetidas: la regla (PARAM)

La regla $(\langle r\rangle')$ evita una derivación infinita como la del Ejemplo 4; sin embargo el uso de esta regla no modifica sustancialmente la derivación del Ejemplo 3, donde se siguen generando infinitos nominales cada vez más alejados de i.

Miremos más en detalle la cláusula D_6 de dicha derivación, que como ya mencionamos, cumple un rol principal en la derivación *cíclica* que genera un conjunto infinito de cláusulas. Esta cláusula se genera a partir de la paramodulación entre las cláusulas D_{4a} y D_{5a}. La fórmula $@_i \langle r\rangle k_2$, sintetizada en dicha operación, es el resultado de asumir que i y k representan lo mismo. Ahora bien, en $D_{5a} = \{@_k \langle r\rangle k_2\}$, k_2 es un nominal introducido bajo la suposición de que k se relaciona con él, y tal que en él p es verdadero. Aquí es importante observar que $nom(@_k \langle r\rangle p) = k_2$. Cuando sintetizamos D_6, k_2 pasa a cumplir un rol adicional: ahora también representa el elemento relacionado con i y tal que p es verdadero en él. Sin embargo, el cálculo tenía *reservado* otro nominal para cumplir esta función: $nom(@_i \langle r\rangle p) = k$.

En la sección anterior vimos que es posible asignar, usando la función nom, un *rol* preciso a cada nominal que se crea durante el cálculo y que esto permite que el cálculo pueda limitarse a considerar modelos más simples. El análisis que acabamos de ver muestra cómo al hacer paramodulación sobre fórmulas de la forma $@_i \langle r\rangle j$ donde alguno de los nominales i, j pertenece a NOM_c, también estamos considerando modelos potencialmente más complejos de lo necesario.

Una solución para este problema consiste en tratar las igualdades sobre fórmulas de la forma $@_i \langle r\rangle j$ (con i, j nominales) de manera tal de aprovechar lo que sabemos sobre los nominales que son introducidos por la aplicación de la regla $(\langle r\rangle')$. A este fin, incorporamos una regla especial de paramodulación sobre fórmulas como la de la cláusula C_6. En la Figura 3 presentamos el cálculo donde la regla (PARAM) ha sido reemplazada por (PARAM') y (PARAM-REL). La segunda regla es la que aplica paramodulación en forma especial sobre las fórmulas que acabamos de ver; (PARAM') difiere de (PARAM) únicamente en que no se aplica sobre las cláusulas que trata (PARAM-REL). Es importante notar que ya no nos alcanza con pedir que $nom(x)$ sea una función inyectiva; en este caso necesitamos que sea un tipo especial de función a la que llamamos *distribuidor de nominales*:

Definición 3.1. Una función $f : \mathcal{H}^{NNF}_{@\Diamond}(@) \to \mathsf{NOM}_c$ es un *distribuidor de nominales* si y sólo si

1. f es biyectiva,
2. la clausura transitiva de la relación dada por Rij si y sólo si $j = f(@_i \langle r\rangle \varphi)$ para algún $\langle r\rangle \varphi$ es un orden parcial, y
3. $i \succ j$ si y sólo si $nom(@_i \langle r\rangle \varphi) \succ nom(@_j \langle r\rangle \varphi)$

Las primeras dos condiciones son suficientemente generales y simplifican los argumentos acerca de este tipo de funciones. Lo que hacen es garantizar que todos los elementos de NOM_c puedan aparecer en $ClSet^*(\varphi)$: la primera condición garantiza sobreyectividad, mientras que la segunda evita *ciclos* como en $i = nom(@_i \langle r\rangle \psi)$. La tercer condición es necesaria para garantizar la completitud del cálculo.

Es necesario comprobar que los cambios introducidos a las reglas de paramodulación preservan consistencia y completitud. Para probar consistencia, se usa el hecho de que dado un modelo cualquiera

$$(\wedge)\quad \frac{Cl \cup \{@_t\,(\varphi_1 \wedge \varphi_2)\}}{\begin{array}{c} Cl \cup \{@_t\,\varphi_1\} \\ Cl \cup \{@_t\,\varphi_2\} \end{array}} \qquad (\vee)\quad \frac{Cl \cup \{@_t\,(\varphi_1 \vee \varphi_2)\}}{Cl \cup \{@_t\,\varphi_1, @_t\,\varphi_2\}}$$

$$(\text{RES})\quad \frac{Cl_1 \cup \{@_t\,\varphi\} \quad Cl_2 \cup \{@_t\,\neg\varphi\}}{Cl_1 \cup Cl_2}$$

$$([r])\quad \frac{Cl_1 \cup \{@_t\,\langle r\rangle s\} \quad Cl_2 \cup \{@_t\,[r]\varphi\}}{Cl_1 \cup Cl_2 \cup \{@_s\,\varphi\}} \qquad (\langle r\rangle')\quad \frac{Cl \cup \{@_t\,\langle r\rangle\varphi\}}{\begin{array}{c} Cl \cup \{@_t\,\langle r\rangle n\} \\ Cl \cup \{@_n\,\varphi\} \end{array}} \quad \begin{array}{l} \text{con } \varphi \notin \mathsf{NOM} \text{ y} \\ n = nom(@_t\,\langle r\rangle\varphi) \end{array}$$

$$(@)\quad \frac{Cl \cup \{@_t\,@_s\,\varphi\}}{Cl \cup \{@_s\,\varphi\}} \qquad (\text{REF})\quad \frac{Cl \cup \{@_t\,\neg t\}}{Cl} \qquad (\text{SYM})\quad \frac{Cl \cup \{@_s\,t\}}{Cl \cup \{@_t\,s\}} \quad \text{si } t \succ s$$

$$(\text{PARAM'})\quad \frac{Cl_1 \cup \{@_s\,t\} \quad Cl_2 \cup \{\varphi(s)\}}{Cl_1 \cup Cl_2 \cup \{\varphi(s/t)\}} \quad \begin{array}{l} \text{si } s \succ t,\ \varphi(s) \succ @_s\,t, \text{ y cuando} \\ \varphi(s) = @_i\,\langle r\rangle j, \text{ entonces } j \in \mathsf{NOM}_i, \\ \text{o } t \in \mathsf{NOM}_i \text{ y } j = s \end{array}$$

$$(\text{PARAM-REL})\quad \frac{Cl_1 \cup \{@_s\,t\} \quad Cl_2 \cup \{@_s\,\langle r\rangle n\}}{\begin{array}{c} Cl_1 \cup Cl_2 \cup \{@_t\,\langle r\rangle k\} \\ Cl_1 \cup Cl_2 \cup \{@_n\,k\} \end{array}} \quad \begin{array}{l} \text{si } s \succ t \text{ y } n \in \mathsf{NOM}_c, \text{ y donde} \\ \text{para algún } \varphi,\ n = nom(@_s\,\varphi), \text{ y} \\ k = nom(@_t\,\varphi) \end{array}$$

Restricciones:

- Si $C = C' \cup \{\varphi\}$ es la premisa principal, entonces o bien $S(C) = \{\varphi\}$ o, en caso contrario, $S(C) = \emptyset$ y $\{\varphi\} \succ C'$
- Si $D = D' \cup \{\psi\}$ es la premisa auxiliar, entonces $\{\psi\} \succ D'$ y $S(D) = \emptyset$
- $nom(x)$ es un *distribuidor de nominales*

(tanto φ como ψ representan la fórmula que participa en la inferencia)

Figura 3: Reglas de resolución para $\mathcal{H}(@)$ con orden, selección y control de nominales

de φ y un distribuidor de nominales $nom(x)$, se puede construir otro modelo de φ pero que además cumpla cierto criterio de compatibilidad con $nom(x)$. Esencialmente, si M es *compatible* con $nom(x)$, debe suceder que $M \models @_s\,\langle r\rangle\psi$ y $i = nom(@_s\,\langle r\rangle\psi)$ implique $M \models @_s\,\langle r\rangle i$ y $M \models @_i\,\psi$; y que si, además, $j = nom(@_t\,\langle r\rangle\psi)$ y $M \models @_s\,r$, entonces $M \models @_i\,j$.

Teorema 8. *Si $\varphi \in \mathcal{H}_0^{NNF}(@)$ es satisfacible,* $\mathrm{ClSet}^*(\varphi)$ *(saturado con las reglas de la Figura 3) también lo es. Por lo tanto, el cálculo de la Figura 3 es consistente.*

Lo primero que conviene observar para probar la completitud del nuevo cálculo es que la restricción sobre el orden de los nominales en la Definición 3.1 permite garantizar que los consecuentes que se obtienen al aplicar la regla (PARAM-REL) son menores que su premisa principal. Con esto, podemos adaptar la demostración del Teorema 5, a las reglas nuevas.

Teorema 9. *El cálculo de la Figura 3 es refutacionalmente completo.*

Como parte del trabajo de tesis también demostramos que la Definición 3.1 no es demasiado estricta, es decir, que puede ser satisfecha por alguna función concreta. Para ello, mostramos que para cualquier lenguaje híbrido podemos extender el conjunto de nominales de forma adecuada, de manera que la construcción de un distribuidor de nominales sea trivial.

3.4. Terminación

En esta sección veremos que si usamos un refinamiento de un orden admisible, el número de fórmulas distintas que el cálculo de la Figura 3 puede generar es finito. Con finitas fórmulas sólo es posible armar un conjunto finito de cláusulas distintas; con lo cual, el conjunto saturación debe ser necesariamente finito. Comencemos, entonces, con una observación simple.

Proposición 10. *Sea φ una fórmula de $\mathcal{H}_0^{NNF}(@)$; para toda fórmula $\psi \in \mathrm{ClSet}^*(\varphi)$ se cumple $size(\varphi) \geq size(\psi)$.*

Al igual que en la Definición 2.5, $size(\varphi)$ representa la cantidad de operadores (incluyendo los de aridad cero) de φ. Lo que esta proposición nos dice es que si $ClSet^*(\varphi)$ es infinito, esto no puede deberse a la presencia de fórmulas infinitamente grandes: es necesario que haya infinitos nominales distintos. Lo que veremos a continuación es que los dos ejemplos presentados en la Sección 3.1 ilustran todas las maneras en que pueden generarse infinitos nominales y que los mecanismos propuestos para remediar esto efectivamente funcionan. Para ello nos valemos de una medida de *distancia* entre un nominal cualquiera y algún nominal de NOM_i.

Definición 3.2. Dado un distribuidor de nominales $nom(x)$, la función $level(\cdot) : \mathsf{NOM} \to \mathbb{N}$ se define como:

$$level(i) = \begin{cases} 0 & \text{si } i \in \mathsf{NOM}_i \\ level(j) + 1 & \text{si } i = nom(@_j \langle r \rangle \varphi) \end{cases}$$

Claramente, $n \in \mathsf{NOM}_i$ si y sólo si $level(n) = 0$. Ahora podemos caracterizar fácilmente las dos maneras de generar infinitos nominales. La primera derivación se caracteriza por el hecho de que $level(n)$, cuando n es un nominal que aparece en $ClSet^*(\varphi)$, no está acotada; la segunda, por el hecho de que existe un conjunto infinito N de nominales que aparecen en $ClSet^*(\varphi)$ para el cual se cumple $level(n) = c$ si $n \in N$, para algún $c > 0$. Es claro que si no se cumple ninguna de estas dos condiciones, el conjunto de nominales no puede ser infinito.

Como ya dijimos, la función *level* nos da una medida de *profundidad* de nominales. Esta idea juega un papel importante, y de hecho influye en la noción de orden con la que trabajaremos.

Definición 3.3. Un orden $\succ$ y un distribuidor de nominales $nom(x)$ son *admisibles para terminación* si $\succ$ es un orden *admisible* y además verifica que para todo par de nominales i y j $level(i) > level(j)$ implica $i \succ j$, donde $level(x)$ está definido en función de $nom(x)$.

De aquí en más, asumimos que $\succ$ y $nom(x)$ son, respectivamente, un orden y un distribuidor de nominales de este tipo.

La prueba de terminación consiste en garantizar que, con el cálculo de la Figura 3, $level(n)$ está acotada cuando n aparece en $ClSet^*(\varphi)$ y que en cada *nivel* hay un número finito de nominales. Para ver la primera condición, utilizamos una noción de *profundidad modal* para fórmulas-@ que combina la profundidad meramente sintáctica (i.e., el concepto habitual de "profundidad modal") con la *profundidad* (según la función *level*) del prefijo de la fórmula-@. Para la segunda, nos valemos de una función que nos devuelve todos los nominales de un conjunto de cláusulas que pertenezcan a determinado nivel

Definición 3.4. Para toda fórmula $@_i \varphi$ definimos $d'(@_i \varphi) = level(i) + d(\varphi)$, donde $d(\varphi)$ es la profundidad modal de φ. Además, dado cualquier conjunto de cláusulas N, definimos $level_x(N) = \{i \mid i \in \mathsf{NOM} \wedge level(i) = x \wedge i \in S_f(N)\}$, donde $S_f(N)$ es el conjunto de todas las subfórmulas que aparecen en N.

Teorema 11. *Para toda fórmula φ, el conjunto de nominales distintos que aparecen en $\mathrm{ClSet}^*(\varphi)$ es finito. Por lo tanto, el cálculo de resolución de la Figura 3 constituye un método de decisión para la lógica $\mathcal{H}(@)$.*

La demostración de este teorema se sigue directamente de estas propiedades, que fueron demostradas. En lo que sigue, φ es una fórmula arbitraria de $\mathcal{H}_0^{NNF}(@)$:

1. Para toda cláusula $\{@_i \langle r \rangle j\} \cup C \in \mathit{ClSet}^*(\varphi)$, o bien $\mathit{level}(j) = 0$ o bien $\mathit{level}(j) = \mathit{level}(i)+1$.
2. Para toda fórmula ψ que aparezca en $\mathit{ClSet}^*(\varphi)$, si en ψ aparece algún $i \in \mathsf{NOM}_c$ (i.e. $\mathit{level}(i) > 0$), entonces ψ es de la forma: $@_i \psi'$, $@_n \langle r \rangle i$ ó $@_n i$ ($i \notin S_f(\psi')$ y $i \neq n$).
3. Si ψ aparece en alguna cláusula de $\mathit{ClSet}^*(\varphi)$, entonces $d'(\psi) \leq d(\varphi)$ y, además, si $\psi = @_i j$, entonces $\mathit{level}(j) \leq d(\varphi)$. Luego, para toda fórmula ψ, si i es un nominal que aparece en alguna fórmula de $\mathit{ClSet}^*(\psi)$, entonces $\mathit{level}(i) \leq d(\psi)$.
4. Para todo nominal i, el número de fórmulas distintas de la forma $@_i \langle r \rangle \psi$ (con $\psi \notin \mathsf{NOM}$) que aparecen en $\mathit{ClSet}^*(\varphi)$ es finito.
5. Para todo $x \in \mathbb{N}$, $\mathit{level}_x(\mathit{ClSet}^*(\varphi))$ es un conjunto finito.

4. Implementación y evaluación

HyLoRes es un demostrador para la lógica $\mathcal{H}(@, \downarrow)$ escrito en Haskell, de aproximadamente 5000 líneas de código, basado en el cálculo de resolución presentado en [2]. Es necesario aclarar que no se trata de una herramienta que pretenda competir seriamente con demostradores que representan el estado del arte en demostración automática para lógica de primer orden o DLs. Demostradores como RACER [10, 17] ó *SAT [9, 19] están especialmente afinados y cuentan con una batería de heurísticas y optimizaciones con las que consiguen resultados sobresalientes. En contraste, HyLoRes implementa un conjunto relativamente simple de optimizaciones y es todavía más una prueba de concepto que una aplicación que pueda ser usada en un ambiente real. Pero es justamente por ello que se trata de una entorno ideal para poner a prueba nuevas ideas y realizar una valoración empírica de las mismas.

Como parte de este trabajo se desarrolló una versión de HyLoRes que utiliza las reglas presentadas en la Figura 3 y se realizaron pruebas para comparar la performance de ambas versiones. En esta sección solamente presentamos algunos resultados de las pruebas que realizamos.

Hoy en día, el test estándar para la lógica modal básica es el denominado 3CNF$\Box_m$ aleatorio [16], que es una adaptación del test 3CNF aleatorio de la lógica proposicional [15]. Este tipo de tests se caracterizan por generar lotes de fórmulas aleatorias que se ajusten a determinadas restricciones (e.g., número de variables proposicionales, profundidad modal, número máximo de cláusulas). En general, cuanto mayor es el número de cláusulas de una de estas fórmulas, mayor es la probabilidad de que sea insatisfacible.

La definición estándar de CNF$\Box_m$ aleatorio genera fórmulas estrictamente modales. En [3] se presenta una extensión para trabajar con demostradores para lógicas híbridas. Esta extensión, hCNF$\Box_m$, genera formulas para cualquier sublenguaje de $\mathcal{H}(@, A, \downarrow)$. hGen, una herramienta que genera casos de test a partir de hCNF$\Box_m$, fue utilizada para realizar las pruebas.

Las variables que utilizamos en la generación de lotes de pruebas fueron: "cantidad de variables proposicionales (V)", "cantidad de nominales (N)", "máxima profundidad modal (D)" y "cantidad de cláusulas (L)". Fijados los valores de V, N y D, se generaron, para cada valor de L, lotes de 100 fórmulas, las cuales fueron dadas como entrada a los demostradores, con un timeout de 40 segundos por fórmula. En los gráficos mostramos, por cada uno, el porcentaje de fórmulas que se decidieron satisfacibles, el porcentaje de insatisfacibles y el de timeouts; también se muestra la mediana del tiempo de ejecución.

En la primera prueba, comparamos la performance de ambas versiones del demostrador utilizando fórmulas simples ($V = 2$, $N = 3$, $D = 1$). En este caso la performance de HyLoRes 2.0 fue claramente superior a la de la versión 1.0. En la Figura 4 se ve claramente cómo a HyLoRes 1.0 le cuesta resolver una fracción importante de los problemas más simples, mientras que HyLoRes 2.0 los resuelve todos. Es

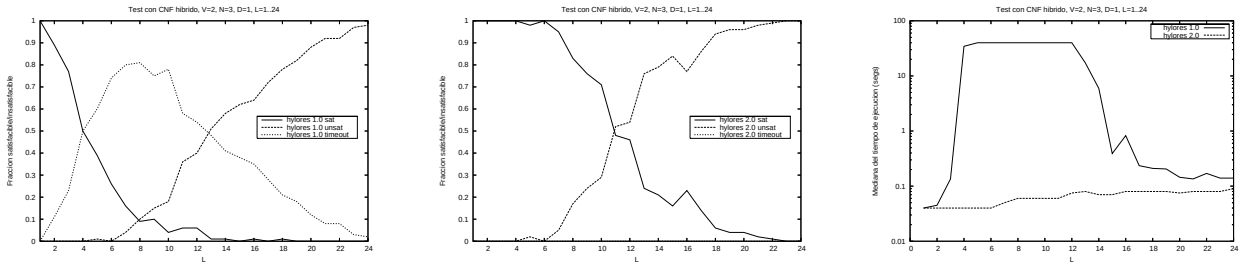

Figura 4: Evaluación con hCNF$\Box_m$ de HyLoRes 1.0 y 2.0 – Fórmulas simples.

interesante notar que HyLoRes 1.0 tiene la mayor proporción de *timeouts* en la región en que la mayoría de las fórmulas son satisfacibles. Se debe tener en cuenta que las restricciones de orden y selección aceleran el proceso de saturación, y que en la versión 1.0 no se contaba con ningún mecanismo de este tipo para la regla de paramodulación.

Como muestra la Figura 4, tener un número importante de *timeouts* afecta negativamente la representatividad de los valores graficados. La curva cortada en el gráfico de cláusulas generadas, y la forma poco usual de la curva del gráfico de tiempo de ejecución son una muestra elocuente de esto. Por otro lado, se puede ver que para valores chicos de L el tiempo de inicialización de HyLoRes 2.0 es mayor al tiempo que necesita para resolver el problema en sí.

Figura 5: Evaluación con hCNF$\Box_m$ de HyLoRes 2.0 – Fórmulas complejas, baja cantidad de nominales.

El número de casos que HyLoRes 1.0 no puede resolver dentro de un tiempo razonable cuando se aumenta la profundidad modal de las fórmulas generadas es demasiado grande como para que podamos tenerlo en cuenta. Es por ello que las últimas pruebas se realizaron sólo sobre distintas configuraciones de la versión 2.0.

En la Figura 5 se muestran los resultados para fórmulas en las que se aumenta la complejidad estrictamente modal: $V = 8$, $N = 3$ y $D = 7$. En este caso, el número de timeouts en la zona más difícil es de un 15 %, sin embargo, el tiempo medio de respuestas sigue sin superar los 2 segundos.

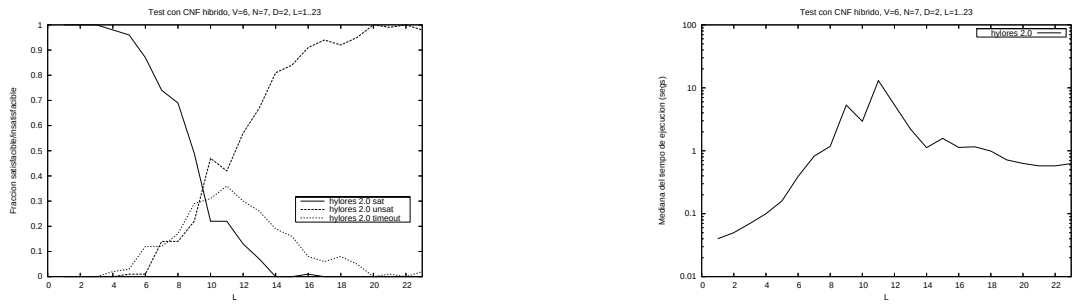

Figura 6: Evaluación con hCNF$\Box_m$ de HyLoRes 2.0 – Fórmulas medianas, alta cantidad de nominales.

Finalmente, en la Figura 6 se muestran los resultados obtenidos con fórmulas en las que aumenta el número de nominales manteniendo baja la profundidad modal: $V = 6$, $N = 7$, $D = 2$. Un número mayor de nominales implica una aplicación más frecuente de las costosas reglas de paramodulación. Como resultado de esto, HyLoRes 2.0 tiene un porcentaje más alto de timeouts con estas fórmulas, y el tiempo medio de ejecución en la zona más difícil supera los 10 segundos. Es importante notar que las fórmulas de la Figura 6 tienen el triple de variables proposicionales y más del doble de nominales que las de la Figura 4, que ya eran demasiado difíciles para HyLoRes 1.0.

Referencias

[1] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *Proceedings of the 8th Annual Conference of the EACSL*, pages 307–321, 1999.

[2] C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logic. *Journal of Logic and Computation*, 11(5):717–736, 2001.

[3] C. Areces and J. Heguiabehere. hGen: A random CNF formula generator for Hybrid Languages. In *Proceedings of Methods for Modalities 3*, Nancy, France, 2003.

[4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[5] L. Bachmair and H. Ganzinger. Equational reasoning in saturation-based theorem proving. In *Automated deduction—a basis for applications, Vol. I*, pages 353–397. Kluwer, Dordrecht, 1998.

[6] L. Bachmair and H. Ganzinger. Resolution theorem proving. In J. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 2, pages 19–99. Elsevier, 2001.

[7] S. Buss. On Herbrand's theorem. In *Logic and Computational Complexity*, number 960 in Lecture Notes in Computer Science, pages 195–209. Springer-Verlag, 1995.

[8] N. Dershowitz and J. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B: Formal Models and Sematics (B)*, pages 243–320. Elsevier and MIT Press, 1990.

[9] E. Giunchiglia, A. Tacchella, and F. Giunchiglia. SAT-based decision procedures for classical modal logics. *Journal of Automated Reasoning*, 28(2):143–171, 2002.

[10] V. Haarslev and R. Möller. RACER system description. In *IJCAR 2001*, number 2083 in LNAI, Siena, Italy, June 2001.

[11] J. Herbrand. *Recherches sur la théorie de la démonstrations*. PhD thesis, Sorbone, Paris, 1930. Reprinted in W. Goldfarb, editor, *Logical Writings*. Reidel, 1971.

[12] S. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24:1–14, 1959.

[13] S. Kripke. Semantic analysis of modal logics I, normal propositional calculi. *Zeitschrift fürr mathemathische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[14] S. Kripke. Semantical consideration on modal logics. *Acta Philosophica Fennica*, 16:83–94, 1963.

[15] D. Mitchell, B. Sleman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. of the 10th National Conference on Artificial Intelligence*, pages 459–465, 1992.

[16] P. Patel-Schneider and R. Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *Journal of Artificial Intelligence Research*, 18:351–389, May 2003.

[17] RACER System Description. `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`, 2004.

[18] J. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, Jan. 1965.

[19] *SAT Project. `http://www.mrg.dist.unige.it/~tac/StarSAT.html`, 2004.

[20] A. Voronkov. Algorithms, datastructures, and other issues in efficient automated deduction. In *IJCAR 2001*, number 2083 in LNAI, pages 13–28, Siena, Italy, June 2001.