Índice General

Índice de Tablas

Introducción

El objetivo principal del presente trabajo es la investigación, caracterización y la demostración de algunas propiedades de las lógicas intermedias, es decir las lógicas que están entre la lógica clásica y la lógica intuicionista, y para encontrar dichas caracterizaciones es necesario contar con una herramienta de software para el estudio de las diferentes axiomatizaciones, y formas excepcionales, debido a su utilidad en programación lógica.

Demostramos el teorema de completamiento para lógicas intuicionistas el cual no se encuentra en ninguna bibliografía referida y un lema de asignación sobre lógicas intermedias.

Este software llamado *'índ20"* verifica la independencia de un axioma con respecto al resto de axiomas de una teoría. Dando lugar a poder encontrar sus formas excepcionales, utilizando para ello una lógica con más valores de verdad que la original.

El material esta ordenado de la siguiente manera: Realizamos una presentación de las nociones básicas de lógica proposicional, con sus diferentes variaciones axiomáticas. Un brevario de las semánticas, además mostramos algunas propiedades de la lógica intuicionista, siguiendo con las lógicas intermedias y continuamos con la presentación de la negación fuerte y débil, Explicamos el concepto de independencia y excepcionalidad. A continuación

pasamos a la descripción del software y pruebas realizadas. y terminamos con una propuesta de paralelización de este software.

Capítulo 1

Preliminares

Damos una breve descripción del cálculo proposicional, ejemplos de diferentes axiomatizaciones y la noción de punto fijo. Para más detalles consúltese: [Men87] y [Dal89].

1.1 Cálculo proposicional

Definición 1 *Una teoría formal L está constituida por:*

1. Un conjunto numerable de símbolos Σ, llamados *"símbolos del lenguaje o alfabeto[HU96] de L"*. Una sucesión finita de símbolos de L, será una *expresión de L*.

2. Un subconjunto de expresiones de L que llamaremos *fórmulas bien formadas ("wfs")*.

3. Un subconjunto de fórmulas bien formadas que llamaremos *axiomas*.

4. Un conjunto finito $\{R_1, R_2, \ldots, R_n\}$ de relaciones entre las fórmulas bien formadas, que llamaremos reglas de inferencia. Para cada R_i existe

un único entero positivo j tal que, para cualquier conjunto de j fórmulas bien formadas y cada fórmula ψ se puede decidir si es que las j fórmulas bien formadas están R_i-relacionadas con ψ. Si este fuese el caso, se dice que ψ es una consecuencia directa de las fórmulas bien formadas dadas por medio de R_i.

Ejemplo 1 *El cálculo de proposiciones es un ejemplo de una teoría formal donde se tiene por alfabeto:*

- Símbolos proposicionales: $p_0, p_1, p_2, \ldots, p_n, \ldots$
- Conectivos: $\sim$ (negación), $\vee$ (disyunción), $\wedge$ (conjunción), $\Rightarrow$ (implicación), $\Leftrightarrow$ (doble implicación).
- Símbolos auxiliares: "(", ")"
- Fórmulas bien formadas: El conjunto de fórmulas bien formadas, wfs:
  1. $p_i \in wfs$ para cada $i \in N$.
  2. Si $p, q \in wfs$ entonces $p \vee q$, $p \wedge q$, $p \Rightarrow q$, $p \Leftrightarrow q \in wfs$.
  3. Si $p \in wfs$ entonces $\sim p \in wfs$.
- Adoptaremos la notación de p, q al referirnos a átomos y Φ, Ψ a wfs.
- Se puede demostrar que los conectivos $\sim$, $\Rightarrow$, son suficientes para expresar a los conectivos $\vee$, $\wedge$, $\Leftrightarrow$, por ejemplo, $p \vee q$ es lógicamente equivalente a $\sim p \Rightarrow q$[Men87].

Definición 2 *Al conjunto de proposiciones $PROP$ es el mínimo conjunto que cumple estas condiciones. Así que tenemos un mecanismo para formar proposiciones a partir de otras.*

Axiomas: Si p, q, r son cualesquiera fórmulas bien formadas, las siguientes expresiones son axiomas para el cálculo proposicional L.

- (A_1) $\psi \Rightarrow (\varphi \Rightarrow \psi)$
- (A_2) $((\psi \Rightarrow (\varphi \Rightarrow \phi)) \Rightarrow ((\psi \Rightarrow \varphi) \Rightarrow (\psi \Rightarrow \phi)))$
- (A_3) $(((\sim \varphi) \Rightarrow (\sim \psi)) \Rightarrow (((\sim \varphi) \Rightarrow \psi) \Rightarrow \varphi))$

Reglas de inferencia: La única regla de inferencia es MP (modus Ponens), esto es: q es consecuencia directa de p y de $(p \Rightarrow q)$.

Comentario 1 *El número de axiomas de esta lógica es infinito*

Notación 1 *Llamaremos* átomos *a los símbolos proposicionales.*

Comentario 2 *El axioma* (A_3) $(((\sim \varphi) \Rightarrow (\sim \psi)) \Rightarrow (((\sim \varphi) \Rightarrow \psi) \Rightarrow \varphi))$ *es equivalente a* $[(\sim p \Rightarrow p) \Rightarrow p)] \wedge [p \Rightarrow (\sim p \Rightarrow q)]$.

Ejemplo 2 *Sea el cálculo de proposiciones usual con los siguientes axiomas:*

1. $\psi \Rightarrow (\varphi \Rightarrow \psi)$
2. $((\psi \Rightarrow (\varphi \Rightarrow \phi)) \Rightarrow ((\psi \Rightarrow \varphi) \Rightarrow (\psi \Rightarrow \phi)))$
3. $(\sim \Phi \Rightarrow \Phi) \Rightarrow \Phi)$ Donde Φ es una wfs.

Definición 3 *Una* prueba en L de A *es una sucesión finita* $A_1, A_2, \ldots, A_n$ *tal que* $A_n = A$ *y cada* A_i *es un axioma o una consecuencia directa de algunas fórmulas bien formadas precedentes, obtenidas por medio de reglas de inferencia (ver [JYTL89]) . Sea* Γ *un conjunto de fórmulas bien formadas.* $\Gamma = \{\beta_1, \beta_2, \ldots, \beta_n\}$. *Diremos que* Γ *prueba a* A *y lo denotaremos* $\Gamma \vdash A$ *si existe una prueba para* A *basada en los axiomas y en las fórmulas* $\beta_i \in \Gamma$.

Definición 4 *Un teorema de L es una wfs A de L tal que existe una prueba cuya última wfs es A. (en este caso $\Gamma = \phi$)*

Definición 5 *p es una subfórmula de q cuando suceda cualquiera de los siguientes casos:*

1. $q = p$
2. $q = (q_1 \Rightarrow q_2)$ *donde* p *es subfórmula de* q_1 *o de* q_2.
3. $q = (\sim q_1)$ *y* p *es una subfórmula de* q_1.

En el aspecto semántico, dado un átomo, podemos asignarle un valor verdadero o falso (o respectivamente uno o cero). Definamos los valores de verdad para la proposición $p \Rightarrow q$ de la manera usual.

Definición 6 *Sea un mapeo* $V : PROP \longrightarrow \{0, 1\}$ *se llamará una valuación si*

1. $V(p \Rightarrow q) = 0$ *si y solo si* $V(p) = 1$ *y* $V(q) = 0$.
2. $V(\sim p) = 1 - V(p)$

Se puede probar que si V_1 es un mapeo del conjunto de átomos a $\{0, 1\}$, entonces existe una única valuación V tal que $V(p) = V_1(p)$ para el caso cuando p es un átomo[Dal89]. El autor prueba que el valor de $V(p)$ de p bajo V depende únicamente de los valores de V en las subfórmulas atómicas de p.

Teorema 1 *Si* $V(p_i) = V_1(p_i)$ *para todo* p_i *que ocurre en* Φ *entonces* $V(\Phi) = V_1(\Phi)$ *(ver [Dal89]).*

Resulta claro que $V(\ p \Rightarrow p) = 1$, independientemente del valor que tenga $V(p)$. Al subconjunto de $PROP$ que evalúen siempre a 1 se les llamará tautologías.

Definición 7 *p es una* tautología *si $V(p) = 1$. Denotaremos esto por $\models p$. Si Γ es un conjunto de proposiciones, $\Gamma \models p$ si y solo si para toda V se tiene que: $V(q) = 1$,$\forall q \in \Gamma$ implica que $V(p) = 1$.*

- Luego entenderemos que un conjunto de proposiciones Γ, valida a una proposición p si toda función de evaluación V que valida a cada una de las proposiciones en Γ, también valide a p. Denotaremos esto por $\Gamma \models p$ y diremos que p es una consecuencia lógica de Γ.

- En el caso de proposiciones, o de manera más general en el caso del cálculo de predicados, $\Gamma \models \Phi$ si y solo si $\Gamma \vdash \Phi$.

Definición 8 *Diremos que I es un modelo si su valuación y su valuación inducida (esto es $V(a) = 1, \forall a \in I$) es tal que $V(c) = 1$*

Definición 9 *Sea V una valuación y sea Φ una fórmula. V es un modelo para Φ en el caso que $V(\Phi) = 1$. Si V es una valuación y sea Γ un conjunto de wfs, diremos que V modela a Γ si $\forall \Phi \in \Gamma$, V modela a Φ*

Entenderemos por $\Phi\,[\Psi/p_i]$ a la proposición que obtenemos a partir de sustituir todas las ocurrencias de p_i en Φ por Ψ. Es claro que si en Φ no contiene alguna ocurrencia de p_i, entonces $\Phi\,[\Psi/p_i] = \Phi$. Adicionalmente se tienen los siguientes resultados:

- $\Phi\,[\Psi/p_i] = \Phi$ si Φ es un átomo y $\Phi \neq p_i$

- $\Phi\,[\Psi/p_i] = \Psi$ si $\Phi = p_i$
- $\Phi_1 \Rightarrow \Phi_2\,[\Psi/p_i] = \Phi_1\,[\Psi/p_i] \Rightarrow \Phi_2\,[\Psi/p_i]$
- $(\sim \Phi)\,[\Psi/p_i] = \sim \Phi\,[\Psi/p_i]$

Necesitaremos los siguientes teoremas:

Teorema 2 *Si* $\models \Phi_1 \Leftrightarrow \Phi_2$, *entonces* $\models \Psi\,[\Phi_1/p] \Leftrightarrow \Psi\,[\Phi_2/p]$ *para la prueba, véase [Dal89].*

Teorema 3 *(El teorema de deducción). Si* Γ *es un conjunto de fórmulas bien formadas, además* A, B *son fórmulas bien formadas y se tiene que* Γ, $A \models B$ *entonces:* $\Gamma \models A \Rightarrow B$. [1]

1.2 Programas definidos y programas normales

En este capítulo centramos nuestro estudio en ciertos conjuntos llamados programas definidos y programas normales, para los cuales proponemos una nueva axiomatización y examinaremos algunas propiedades de éstos, bajo esta nueva perspectiva.

Definición 10 *Llamaremos literal a un átomo o a la negación de un átomo.*

Definición 11 *Una cláusula es una wfs de la forma:* $A_1 \vee A_2 \vee \cdots \vee A_k \vee \sim B_1 \vee \sim B_2 \vee \cdots \vee \sim B_n$. *Donde* A_i, B_j, *son átomos. Por las leyes de De*

[1] Es de resaltar el hecho que en la prueba del teorema de deducción sólo se utilizan los axiomas A_1 y A_2, además la regla de inferencia MP. Esto significa que en cualquier sistema que tenga lo anteriormente señalado, vale el teorema de deducción[Men87].

Morgan, una cláusula se puede poner en la forma: $B_1 \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow A_1 \vee A_2 \vee \cdots \vee A_k$. *Por notación esto se expresará indistintamente como* $A_1 \vee A_2 \vee \cdots \vee A_k \Leftarrow B_1 \wedge B_2 \wedge \cdots \wedge B_n$, *como* $A_1, A_2, ..., A_k \Leftarrow B_1, B_2, ..., B_n$, *o como* $A_1, A_2, ..., A_k : -B_1, B_2, ..., B_n$ *Se le llamará a* $A_1, A_2, ..., A_k$ *la cabeza de la cláusula y a* $B_1, B_2, ...B_n$ *el cuerpo de la cláusula.*

Definición 12 *Una cláusula para un programa definido o una cláusula definida es una cláusula de la forma* $A \Leftarrow B_1, B_2, \ldots, B_n$ *donde* A, B_j *son átomos. De acuerdo a nuestra convención de utilizar únicamente los conectivos lógicos* $\sim$, *e* $\Rightarrow$, *una cláusula es una* wfs *de la forma*

$$B_1 \Rightarrow (B_2 \Rightarrow (\ldots (B_n \Rightarrow A)) \ldots)$$

Definición 13 *Un programa definido es un conjunto finito de cláusulas para programas definidos.*

Definición 14 *Una cláusula normal para un programa es una cláusula de la forma* $A \Leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_n$. *Donde* A *es un átomo y* L_i *son literales. Nuevamente esto se deberá entender como* $L_1 (\Rightarrow L_2 (\Rightarrow (\ldots (L_n \Rightarrow A) \ldots)$

Definición 15 *Una cláusula normal extendida es una cláusula de la forma* $A \Leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_n$. *Donde* A, L_i *son literales.*

1.3 Puntos Fijos

1.3.1 Relaciones y conjuntos parcialmente ordenados

Un orden parcial en un conjunto S es una relación que es reflexiva, transitiva y antisimétrica. Si S es una familia de conjuntos, la relación " $\subseteq$ " "es

subconjunto de" es un orden parcial en. Un conjunto parcialmente ordenado es un conjunto con un orden parcial definido en él. Escribamos esto como $(S, \text{“} \leq \text{”})$. Sea $(S, \text{“} \leq \text{”})$, si $x \leq y$ y no existe z tal que $z \leq x$, se dice que x es elemento mínimo de S. Un elemento a en un conjunto parcialmente ordenado es una cota inferior de un subconjunto E si $a \leq x$ para toda $x \in E$. Similarmente a es una cota superior de E en el caso que $x \leq a$ para todo $x \in E$. α es el supremo de E si es la mínima de las cotas superiores. β es el ínfimo de E si es la más grande de las cotas inferiores.

1.3.2 Lattices completos y mapeos monótonos

Sea $(L, \text{“} \leq \text{”})$ y supongamos que para todo $E \supseteq L$, E siempre tenga ínfimo y supremo. Entonces diremos que $(L, \text{“} \leq \text{”})$ es un lattice completo. Un lattice completo tiene un supremo e ínfimo sobre el mismo lattice. Sea S un conjunto, $(2^S, \text{“} \subseteq \text{”})$ es un lattice completo. El supremo de 2^S es S y el ínfimo es $\emptyset$. Podemos ahora considerar funciones o mapeos con dominio y rango en L. Sea $(L, \text{“} \leq \text{”})$ lattice completo, $T : L \to L$ mapeo. Diremos que T es monótona si siempre que $x \leq y$ implica que $T(x) \leq T(y)$. Sea $(L, \text{“} \leq \text{”})$ lattice completo y sea $X \subseteq L$. X se llamará un subconjunto dirigido si para cualquier subconjunto finito de X, admite una cota superior en X. Cuando T sea un mapeo que respete supremos sobre conjuntos dirigidos, diremos que el mapeo T es continuo. Sea $(L, \text{“} \leq \text{”})$ lattice completo. $T : L \longrightarrow L$. Llamaremos T es continua si dado cualquier subconjunto $X \subseteq L$ dirigido: $T(\sup\{X\}) = \sup\{T(X)\}$. Diremos que $a \in L$ es un punto fijo de T si $T(a) = a$. Si a es punto fijo y además para todo $b \in L$ también punto fijo se tiene que $a \leq b$ se dice que a es el menor punto fijo. Similarmente se define el

concepto de mayor punto fijo. Ahora enunciaremos un importante teorema sobre puntos fijos.

Teorema 4 *Sea* $(L, \text{“} \leq \text{”})$ *lattice completo. Sea* $T : L \longrightarrow L$ *monótona. Entonces*

1. El menor punto fijo de T es fijo igual a α, donde

$$\alpha = \inf \{x : T(x) = x\} = \inf \{x : T(x) \leq x\}$$

2. El mayor punto fijo de T es igual a β, donde

$$\beta = \sup \{x : T(x) = x\} = \sup \{x : x \leq T(x)\}$$

Para más detalles consultar [Llo87].

Capítulo 2

Algunas caracterizaciones de lógicas y semánticas

2.1 Lógicas no monótonas

La lógica clásica no otorga la expresividad suficiente para representar las formas de razonamiento que los seres humanos utilizamos habitualmente. Uno de estos tipos de razonamiento es el monótono o revisable, bosquejado en la sección anterior (ver [BDK97] y [ADO99]). Le llamamos monótono en el sentido siguiente: supongamos que de un conjunto de premisas S podemos inferir la aseveración F, si agregamos hechos a S es posible que ya no podamos inferir F. Esta situación es contraria al comportamiento de la lógica clásica. Para una explicación detallada de los problemas de representación del conocimiento que condujeron al estudio del razonamiento no monótono, y de las teorías que se han desarrollado, se sugiere revisar [BDK97], [Dix95], [DFN99] y [Llo87].

Aquí bosquejaremos cuatro lógicas distintas que se han desarrollado para

representar el razonamiento no monótono.

Observemos que para definir una lógica, dado el lenguaje (conjunto de fórmulas bien formadas) y el conjunto de axiomas, podemos proceder de dos formas: definir el conjunto de interpretaciones del lenguaje que modelan a el conjunto de axiomas, o definir un conjunto de reglas de inferencia. De cualquiera de estas formas se proporciona una manera de decidir que fórmulas se pueden deducir mediante dicha lógica, en otras palabras, cuales fórmulas son consecuencia lógica de los axiomas.

La primera lógica es muy sencilla, la *Lógica de la Asunción del Mundo Cerrado*, *CWA* (por sus siglas en inglés: Closed World Assumption). Ésta consiste en agregar a la lógica clásica la siguiente regla de inferencia: si no se puede inferir un átomo ground de la teoría, entonces se infiere la negación de dicho átomo [Gab91].

Lógica por Defecto da una manera de representar *hechos* y *defectos* (o prejuicios), a partir de éstos se definen las extensiones. Intuitivamente, un conjunto de fórmulas es una *extensión* si contiene a la cerradura lógica clásica del conjunto de hechos, si todos los defectos que son aplicables en el conjunto se han aplicado y si cada fórmula en el conjunto tiene una derivación a partir de los hechos y de los defectos aplicables. Una fórmula es una consecuencia lógica incrédula si pertenece a todas las extensiones de la teoría y es una consecuencia lógica crédula si pertenece a alguna de las extensiones.

Lógica Autoepistémica (ver [Gel87])es una lógica modal y de creencias. Se amplía el lenguaje de primer orden con la inclusión de un nuevo operador unario B. Éste representa a un agente de razonamiento que tiene creencias propias. Se introducen las reglas de inferencia: *introspección negativa* (si φ no es creída por el agente, $\neg B\varphi$ es creída) e *introspección positiva* (si

φ es creída por el agente, $B\varphi$ es creída). A partir de esto, se definen las *expansiones*. Una fórmula es una *consecuencia lógica incrédula* si pertenece a todas las expansiones de la teoría y es una *consecuencia lógica crédula* si pertenece a alguna de las expansiones.

Por último, la lógica definida por *Circunscripción*. La Circunscripción es un método para definir la cerradura lógica de una teoría mediante la restricción de sus modelos a aquellos que tienen una extensión mínima de algunos predicados previamente selectos, es un caso particular de las llamadas Lógicas Preferenciales. La manera de elegir a los *modelos preferidos* es la siguiente. Se define un orden $\prec$ en las interpretaciones del lenguaje, una interpretación I satisface preferencialmente a una sentencia A si y sólo si satisface a A y no existe $I^{'}$ que satisface A con $I^{'} \prec I$. En este caso, I es un modelo preferido de A. Como vimos antes, F será una consecuencia lógica de la teoría si es verdadera para todo modelo preferido.

2.2 Razonamiento revisable

El razonamiento no monótono o revisable tiene como característica principal el hecho de que, habiendo obtenido una conclusión a partir de un conocimiento dado, esa conclusión puede ser refutada mediante la obtención de nuevo conocimiento.

Daremos aquí algunos ejemplos clásicos de este tipo de razonamiento. Para una explicación más detallada se puede consultar [BDK97] y [DFN99].

Razonamiento del tipo CWA (Closed World Assumption). Supongamos que nos enteramos que un amigo ha comido hoy a la una de la tarde. Si alguien nos pregunta si nuestro amigo comió a las dos, aunque nadie nos haya

dicho que el amigo en cuestión no comió a esa hora, nosotros contestaremos negativamente, ¿Cómo sabemos que, en efecto, no comió a las dos?, lo hemos deducido, sin embargo; si después nos enteramos de que el pobre chico, después de haber comido con su novia a la una, se encontró con su otra novia y comió también con ella a las dos (para despistar), se habrá refutado nuestra deducción mediante la obtención de más conocimiento.

Razonamiento del tipo Abductivo. En el razonamiento lógico clásico, si sabemos que una situación B es consecuencia de una situación A y nos enteramos de que A ha ocurrido, podemos deducir B $\left(\frac{A, A \supset B}{B}\right)$. En el diagnóstico médico, si se sabe que una enfermedad A ocasiona los síntomas B y se sabe que un paciente tiene los síntomas B, se deduce que la enfermedad que padece es A $\left(\frac{B, A \supset B}{A}\right)$, éste es llamado razonamiento abductivo, es revisable porque si el médico se entera de que el paciente tiene más síntomas, o de información que el paciente le había ocultado (tal vez involuntariamente), puede darse cuenta de que su diagnóstico ha sido incorrecto.

Razonamiento por defecto. Supongamos que un amigo nos presentará mañana a una chica nacida en Tabasco, por cierto prejuicio (por defecto), nosotros deducimos que la muchacha es morena y de formas turgentes (así como nos gustan) y nos preparamos muy bien para el encuentro. Es claro que nuestra conclusión puede ser refutada por el nuevo conocimiento que obtendremos mañana. Así, estamos ante un razonamiento revisable.

2.3 LP-Semánticas

Describiremos a continuación los conceptos básicos que involucran las semánticas, como modelo mínimo, interpretación de Herbrand, Validez, etc.

2.4 LP-Semánticas

Proposición 1 *Sea P un programa definido, sea $MOD(P)$ el conjunto de modelos de Herbrand bivaluados de P (como sabemos cada modelo se puede ver como un subconjunto de B_P). La intersección de $MOD(P)$, denotada por M_P, siempre existe y es también un modelo de P. Le llamaremos* el Modelo Mínimo de Herbrand de P.

Proposición 2 *Sea P un programa definido. Sucede que $M_P = \{A \in B_P : A$ es una consecuencia lógica de $P\}$.*

Definición 16 *Sea P un programa definido. El mapeo $\mathbf{T}_P : 2^{B_P} \to 2^{B_P}$ es definido como sigue. Sea I una interpretación de Herbrand. Entonces $\mathbf{T}_P(I) = \{A \in B_P : A \leftarrow A_1, ..., A_n$ es una instancia ground de una cláusula en P y $\{A_1, ..., A_n\} \subseteq I\}$. Se puede ver que T_P es continuo.*

Proposición 3 *Sea P un programa definido. Sucede que $M_P = lfp(T_P) = T_P \uparrow \omega$.*

Definición 17 *Sea P un programa definido y G una consulta definida. Una* respuesta *para $P \cup \{G\}$ es una sustitución para algunas de las variables de G. Si G es de la forma $\leftarrow A_1, ..., A_k$, se dice que θ es una* respuesta correcta *para $P \cup \{G\}$ si $\forall((A_1, ..., A_k)\theta)$ es una consecuencia lógica de P.*

Tenemos entonces que M_P es la semántica natural de un programa P. La manera computacional de encontrar los elementos de este modelo es el bien conocido procedimiento llamado Derivación-SLD. Para los detalles de este método se puede consultar [Llo87]. Dados un programa y una meta definidos, un sistema de programación lógica que utilice la Derivación-SLD

podrá entregar una respuesta a la que llamaremos respuesta computada. Los resultados teóricos más interesantes aquí son los de validez y completez de la Derivación-SLD.

Teorema 5 *(Validez) Sea P un programa definido y G una meta definida. Sucede que una respuesta computada de $P \cup \{G\}$ es también una respuesta correcta de $P \cup \{G\}$.*

Teorema 6 *(Completez) Sea P un programa definido y G una meta definida. Para cada respuesta correcta θ para $P \cup \{G\}$, existe una respuesta computada σ para $P \cup \{G\}$ y una sustitución γ tal que $\theta = \sigma\gamma$.*

Veamos como podemos inferir información negativa a partir de programas definidos.

Definición 18 *(*CWA*) Sea P un programa definido y $A \in B_P$. Si A no es una consecuencia lógica de P podemos inferir $\neg A$.*

Definición 19 *(*Regla de Herbrand*) Sea P un programa definido y $A \in B_P$. Si $comp(P) \cup \{A\}$ no tiene un modelo de Herbrand podemos inferir $\neg A$.*

Definición 20 *(*Negation as Failure*) Sea P un programa definido y $A \in B_P$. Si $P \cup \{\leftarrow A\}$ tiene un árbol SLD finito y fallido podemos inferir $\neg A$.*

Consideremos ahora los programas normales.

Para estos programas, la definición de las semánticas está basada en el completamiento de Clark de un programa, definido en el capítulo anterior.

Definición 21 *Sea P un programa normal y G una consulta normal. Una* respuesta *para $P \cup \{G\}$ es una sustitución para algunas de las variables de G. Si G es de la forma $\leftarrow L_1, ..., L_k$, se dice que θ es una* respuesta correcta *para $comp(P) \cup \{G\}$ si $\forall((L_1, ..., L_k)\theta)$ es una consecuencia lógica de $comp(P)$.*

El método computacional para encontrar las respuestas para una consulta normal es el procedimiento llamado Derivación-SLDNF. Para los detalles de este método se puede consultar [Llo87]. Dados un programa y una meta definidos, un sistema de programación lógica que utilice la Derivación-SLDNF podrá entregar una respuesta a la que llamaremos respuesta computada. El resultado de validez de la Derivación-SLDNF se da a continuación, la completez sólo ocurre bajo ciertas condiciones (véase [Llo87], [BD94a] y [BD94b]).

Teorema 7 *(Validez) Sea P un programa normal y G una meta normal. Sucede que una respuesta computada de $P \cup \{G\}$ es también una respuesta correcta de $P \cup \{G\}$.*

Sumarizando, hemos definido una semántica para programas definidos mediante la exhibición de un modelo, la semántica del modelo mínimo. Hemos definido una semántica para programas normales mediante la exhibición de una regla de inferencia (SLDNF), esta semántica consiste de el conjunto de consecuencias lógicas del completamiento del programa y le denotamos por *COMP*. Sin embargo, esta última presenta algunos inconvenientes. Primero, el completamiento de un programa puede ser inconsistente. Segundo, en el procedimiento SLDNF se puede producir un estancamiento (floundering), es decir; es posible que en la secuencia de consultas generada por el proceso, exista una consulta en la cual ninguna literal pueda ser seleccionada. Por esto se ha definido una semántica basada en modelos trivaluados que resuelve el problema de inconsistencia, la llamada $COMP_3$ (la lógica trivaluada que se considera en estos modelos es la de Lukasiewicz). Esta semántica se define como un punto fijo del siguiente operador:

Definición 22 *Sea P un programa normal, definimos el mapeo $\phi_P : 3^{B_P} \rightarrow$*

3^{B_P} *(3^{B_P} es el conjunto de interpretaciones de Herbrand trivaluadas para L_P) de la siguiente manera.*

$$\mathbf{\Phi}_P(I)(A) = \begin{cases} \text{t}, \ \textit{si existe } A \leftarrow L_1, ..., L_n \textit{ tal que es instancia ground} \\ \textit{de una cláusula en } P \textit{ y } \forall i \leq n \textit{ sucede que } I(L_i) = t \\ \text{f}, \ \textit{si para toda } A \leftarrow L_1, ..., L_n \textit{ tal que es instancia ground} \\ \textit{de una cláusula en } P \textit{ sucede que } \exists i \leq n \textit{ con } I(L_i) = f \\ \text{u}, \ \textit{de otra manera.} \end{cases}$$

donde t *es verdadero (1),* f *es falso (0) y* u *es indefinido (*1/2*).*

Definición 23 *Sean I y J modelos trivaluados de un programa P. Decimos que $\mathbf{I} \leq_k \mathbf{J}$ si $True(I) \subseteq True(J)$ y $False(I) \subseteq False(J)$. Decimos que $\mathbf{I} \leq_t \mathbf{J}$ si $True(I) \subseteq True(J)$. Donde $True(I)$ es el conjunto de átomos ground a los que el modelo I evalúa verdaderos y $False(I)$ es el conjunto de átomos ground a los que el modelo I evalúa falsos .*

Definición 24 *Sean SEM_1 y SEM_2 dos semánticas, decimos que $SEM_1 \leq_k SEM_2$ si para todos los programas P y todas las literales l lo siguiente sucede: l es verdadero en cada uno de los modelos que conforman $SEM_1(P)$ implica que l es verdadero en cada uno de los modelos que conforman $SEM_1(P)$.*

Proposición 4 *Φ_P es $\leq_k$-monótono, luego tiene un menor punto fijo. Este punto fijo es un $\leq_k$-modelo mínimo trivaluado de comp(P). Definimos* $\text{COMP}_3(P) = lfp(\Phi_P)$.

Capítulo 3

Hacia las lógicas Intermedias

El matemático holandés L.E.J. Brouwer (1881-1966) en principios del siglo 20 tenía la visión fundamental que se evitarán los argumentos del no constructivismo, abandonando un principio de lógica clásica que está detrás de las leyes de Morgan. Éste es el principio del tercero excluido (o medio excluido) que afirma que, para cada proposición ψ, ψ o no ψ; y equivalentemente que, para cada ψ, $\neg\neg\psi$ implica ψ. Este principio es básico en la lógica clásica y ya se había enunciado por Aristóteles, aunque con algunas reservaciones, cuando él señaló que la proposición "habrá una batalla en el mar del mañana"es ni verdadero ni falso.

Brouwer no pidió que el principio del tercero excluido siempre fallara, sólo que puede fallar en la presencia de conjuntos infinitos. De dos números naturales x y y que uno siempre puede decidir si $x = y$ o $x \neq y$, pero de dos números reales esto no puede ser posible, como uno sabe, puede tener un número infinito de dígitos sus expansiones decimales. Las objeciones similares aplican a las leyes de Morgan, una consecuencia del principio del tercero excluido.

Arend Heyting (1898-1980) discípulo de Brouwer iniciaron un lenguaje formal para la aritmética intuicionista de primer-orden. Posteriormente algunos de los seguidores de Brouwer estudian la teoría de tipos intuicionista que sólo difiere de la teoría de tipos clásica por la ausencia de un solo axioma (la negación doble):

$$\neg\neg\psi \rightarrow \psi$$

La forma moderada de intuicionismo considerada por el constructivismo de Kronecker no considera la posición más extrema de finitismo. Según este punto de vista que regresa a Aristóteles, los conjuntos infinitos no existen, exceptuando potencialmente. De hecho, precisamente está en la presencia de conjuntos infinitos que los intuicionistas dejan caer el principio clásico del tercero excluido.

3.1 Lógica intuicionista

Exponemos brevemente algunas propiedades que se cumplen en lógica intuicionista, es importante observar el uso de la *"La negación como fallo"* en lógica intuicionista.

Axiomas: Si ψ, φ, ϕ son cualesquiera fórmulas bien formadas, las siguientes expresiones son axiomas para la lógica intuicionista I.

(A_1) $\psi \Rightarrow (\varphi \Rightarrow \psi)$

(A_2) $((\psi \Rightarrow (\varphi \Rightarrow \phi)) \Rightarrow ((\psi \Rightarrow \varphi) \Rightarrow (\psi \Rightarrow \phi)))$

(A_{3i}) $\psi \wedge \varphi \Rightarrow \psi$

(A_{4i}) $\psi \wedge \varphi \Rightarrow \varphi$

(A_{5i}) $\psi \Rightarrow (\varphi \Rightarrow (\psi \wedge \varphi))$

(A_{6i}) $\psi \Rightarrow (\psi \vee \varphi)$

(A_{7i}) $\varphi \Rightarrow (\psi \vee \varphi)$

(A_{8i}) $(\psi \Rightarrow \phi) \Rightarrow ((\varphi \Rightarrow \phi) \Rightarrow (\psi \vee \varphi \Rightarrow \phi))$

(A_{9i}) $(\psi \Rightarrow \varphi) \Rightarrow ((\psi \Rightarrow \sim \varphi) \Rightarrow \sim \psi)$

(A_{10i}) $\sim \psi \Rightarrow (\psi \Rightarrow \varphi)$

Reglas de inferencia: La única regla de inferencia es MP (modus Ponens), esto es: φ es consecuencia directa de ψ y de $(\psi \Rightarrow \varphi)$.

Definición 25 "*a* prueba a φ"*, donde el concepto de prueba se comprende como una construcción. Las pruebas de declaraciones compuestas dependen de las pruebas de sus partes.*

- *a prueba* $\varphi \wedge \psi :=$ *a es un par* $< b, c >$ *tal que b prueba* φ *y c prueba* ψ
- *a prueba* $\varphi \vee \psi :=$ *a es un par* $< b, c >$ *tal que b es un numero natural y si* $b = 0$ *entonces c prueba* φ*, si* $b \neq 0$ *entonces c prueba* ψ*.*
- *a prueba* $\varphi \Rightarrow \psi :=$ *a es una construcción que convierte cualquier prueba p de* ϕ *dentro de una prueba* $a(p)$ *de* ϕ
- *no a prueba* $\perp$

Ejemplo 3 $\varphi \wedge \psi \Rightarrow \psi$ *es verdadera, para* $< a, b >$ *una prueba de* $\varphi \wedge \psi$*, entonces la construcción c con* $c(a, b) = a$ *convierte una prueba de* $\varphi \wedge \psi$ *a una prueba de* φ*. Así "c"prueba* $(\varphi \wedge \psi \Rightarrow \varphi)$*.*

En lógica intuicionista se cumplen las siguientes propiedades, algunas demostradas en las siguiente sección:

$\vdash_I \varphi \wedge \psi \Leftrightarrow \psi \wedge \varphi$

$\vdash_I \varphi \vee \psi \Leftrightarrow \psi \vee \varphi$

$\vdash_I (\varphi \wedge \psi) \wedge \sigma \Leftrightarrow \varphi \wedge (\psi \wedge \sigma)$

$\vdash_I (\varphi \vee \psi) \vee \sigma \Leftrightarrow \varphi \vee (\psi \vee \sigma)$

$\vdash_I \varphi \vee (\psi \wedge \sigma) \Leftrightarrow (\varphi \vee \psi) \wedge (\varphi \vee \sigma)$

$\vdash_I \varphi \wedge (\psi \vee \sigma) \Leftrightarrow (\varphi \wedge \psi) \vee (\varphi \wedge \sigma)$

$\vdash_I \varphi \Rightarrow \neg\neg\varphi$

$\vdash_I (\varphi \Rightarrow (\psi \Rightarrow \sigma) \Leftrightarrow \varphi \wedge \psi \Rightarrow \sigma)$

$\vdash_I \varphi \Rightarrow (\psi \Rightarrow \varphi)$

$\vdash_I \varphi \Rightarrow (\neg\varphi \Rightarrow \psi)$

$\vdash_I \neg(\varphi \vee \psi) \Leftrightarrow \neg\varphi \wedge \neg\psi$

$\vdash_I \neg\psi \vee \neg\varphi \Rightarrow \neg(\psi \wedge \varphi)$

$\vdash_I (\neg\varphi \vee \psi) \Leftrightarrow (\varphi \Rightarrow \psi)$

$\vdash_I (\varphi \Rightarrow \psi) \Rightarrow (\neg\psi \Rightarrow \neg\varphi)$

$\vdash_I (\varphi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \sigma) \Rightarrow (\varphi \Rightarrow \sigma))$

3.2 Teorema de extensión para teorías intuicionistas

Lema 1 $\vdash_I \psi \Rightarrow \psi$

Demostración.

(1)	$(\psi \Rightarrow ((\psi \Rightarrow \psi) \Rightarrow \psi)) \Rightarrow ((\psi \Rightarrow (\psi \Rightarrow \psi)) \Rightarrow (\psi \Rightarrow \psi))$	axioma A_2
(2)	$\psi \Rightarrow ((\psi \Rightarrow \psi) \Rightarrow \psi)$	axioma A_1
(3)	$(\psi \Rightarrow (\psi \Rightarrow \psi)) \Rightarrow (\psi \Rightarrow \psi)$	MP (1), (2)
(4)	$\psi \Rightarrow (\psi \Rightarrow \psi)$	axioma A_1
(5)	$(\psi \Rightarrow \psi)$	MP (3), (4)

■

Lema 2 *(teorema de deducción intuicionista)*

Demostración. La demostración del teorema de deducción de lógica clásica es válido en lógica intuicionista, porque solo utiliza los axiomas A_1 y A_2, que se encuentran también en lógica intuicionista. ■

Lema 3 $\vdash_I \psi \Rightarrow \sim\sim \psi$

Demostración. Por teorema de deducción intuicionista, si $\psi \vdash_I \sim\sim \psi$ entonces $\vdash_I \psi \Rightarrow \sim\sim \psi$

(1)	ψ	hip.
(2)	$(\sim \psi \Rightarrow \psi) \Rightarrow ((\sim \psi \Rightarrow \sim \psi) \Rightarrow \sim\sim \psi)$	axioma A_9
(3)	$\psi \Rightarrow (\sim \psi \Rightarrow \sim \psi)$	axioma A_1
(4)	$\sim \psi \Rightarrow \psi$	MP (1),(3)
(5)	$(\sim \psi \Rightarrow \sim \psi) \Rightarrow \sim\sim \psi$	MP (4),(2)
(6)	$\sim \psi \Rightarrow \sim \psi$	Lema 1
(7)	$\sim\sim \psi$	MP (6),(5)

■

Lema 4 $\vdash_I \varphi \Rightarrow (\sim \varphi \Rightarrow \psi)$

Demostración. Por teorema de deducción intuicionista, si $\psi, \sim \psi \vdash_I \varphi$ entonces $\vdash_I \varphi \Rightarrow (\sim \varphi \Rightarrow \psi)$

(1)	ψ	hip.
(2)	$\sim \psi$	hip.
(3)	$\sim \psi \Rightarrow (\psi \Rightarrow \varphi)$	axioma A_{10}
(4)	$\psi \Rightarrow \varphi$	MP (2), (3)
(5)	φ	MP (1), (4)

■

Lema 5 $\vdash_I (\psi \Rightarrow \sim \psi) \Rightarrow \sim \psi$

Demostración.

(1)	$(\psi \Rightarrow \psi)$	por Lemma 1
(2)	$(\psi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \sim \psi) \Rightarrow \sim \psi)$	por axioma A_9 ■
(3)	$(\psi \Rightarrow \sim \psi) \Rightarrow \sim \psi$	MP (1), (2)

Lema 6 *Si una ws* $\sim \psi$ *de la teoría intuicionista* K *no es demostrable en* K*, entonces la teoría intuicionista* K' *obtenida de* K *agregando* ψ *como un axioma, es consistente.*

Demostración. (por contradicción)

Suponemos que K' es inconsistente. Entonces, para alguna wf φ, $\vdash_{K'} \varphi$ y $\vdash_{K'} \sim \varphi$.

(1)	$\vdash_{K'} \varphi$	hip.
(2)	$\vdash_{K'} \sim \varphi$	hip.
(3)	$\vdash_{K'} \varphi \Rightarrow (\sim \varphi \Rightarrow \sim \psi)$	Lema 4
(4)	$\vdash_{K'} \sim \varphi \Rightarrow \sim \psi$	M.P. (1), (3)
(5)	$\vdash_{K'} \sim \psi$	M.P. (2), (4)
(6)	$\psi \vdash_K \sim \psi$	hip. construcción de K
(7)	$\vdash_K \psi \Rightarrow \sim \psi$	Lema 2
(8)	$\vdash_K (\psi \Rightarrow \sim \psi) \Rightarrow \sim \psi$	Lema 5
(9)	$\vdash_K \sim \psi$	M.P. (7), (8)

Por lo tanto $\vdash_K \sim \psi$ contradiciendo nuestra hipótesis.
(Similarmente, si ψ no es demostrable en K, entonces la nueva teoría obtenida agregando $\sim \psi$ como un axioma de K es consistente).

■

Lema 7 *Si K es una teoría intuicionista consistente, entonces hay una extensión de K completa y consistente.*

Demostración. Sea $\psi_1, \psi_2, \ldots,$ una enumeración de todas las wfs de K. Definimos una secuencia $J_0, J_1, J_2, \ldots$ de teorías como sigue. J_0 es K.
Suponemos que J_n esta definido con $n \geq 0$.

- Si no es el caso que $\vdash_{J_n} \sim \psi_{n+1}$ entonces J_{n+1} es obtenida de J agregando ψ_{n+1} como un axioma adicional

- Si $\vdash_{J_n} \sim \psi_{n+1}$ entonces $J_{n+1} = J_n$

Sea J la teoría obtenida tomando como sus axiomas, todos los axiomas de todas las J_i. Donde J_{n+1} es una extensión de J_n y J es una extensión de todas las J_i, incluyendo $J_0 = K$.

consistencia Para mostrar que J es consistente, es suficiente con demostrar que todas las J_i son consistentes. La demostración de la consistencia de J_i la haremos por inducción.
Por Hipótesis, $J_0 = K$ es consistente.
Supongamos que J_i es consistente.

Si $J_{i+1} = J_i$ entonces J_{i+1} es consistente.

Si $J_{i+1} \neq J_i$ entonces por la definición de J_{i+1}, $\sim \psi_{i+1}$ no es demostrable en J_i, entonces por lema 6, J_{i+1} también es consistente.

Aquí J_{i+1} es consistente si J_i lo es, por lo tanto J es consistente.

completitud Demostremos que J es completo, sea φ cualquier wf de K.

Entonces $\varphi = \psi_{j+1}$ para alguna $j \geq 0$.

Ahora, $\vdash_{J_j} \sim \psi_{j+1}$ o $\vdash_{J_{j+1}} \psi_{j+1}$, entonces, si no $\vdash_{J_j} \sim \psi_{j+1}$, entonces agregamos ψ_{j+1} como un axioma en J_{j+1}.

Por lo tanto, cualquiera $\vdash_{J_{j+1}} \sim \psi_{j+1}$ o $\vdash_{J_j} \psi_{j+1}$.

Así J es una extensión completa y consistente de K ■

3.3 Negación-Estable en programación lógica

Aquí establecemos los conceptos de negación entre los extremos de las lógicas intermedias, por un lado la negación fuerte (lógica clásica) y por otro la negación débil o negación como falla (lógica intuicionista).

En programación lógica, entendido en el sentido amplio para que también abarque la representación del conocimiento y los vagos formalismos del razonamiento basado en un tipo de lenguaje de programación, resulta que los dos estilos de negación son ubicuos.

- El primer estilo más común de negación, presente en una forma u otra en casi cada sistema, normalmente se llama *negación-como-falla*; esta etiqueta denota el hecho que $\neg\, \phi$ se piensa que significa algo como *los ϕ no pueden ser probados.*

- El segundo estilo de negación, de orígenes más recientes en tipos de programación lógica, llamado *negación fuerte*: pero a veces es llamado *negación explícita'*, o incluso *negación clásica*. Esta negación que es verdadera si y sólo si la proposición que se niega es falsa. Denotada por '$\sim$'.

Si estamos de acuerdo que, generalmente hablando, lo que es falso no puede demostrarse verdadero, entonces el nombre de negación *'fuerte'* parece bastante apropiada, desde que $\neg\, \phi$ debe ser reducible de $\sim \phi$ en cualquier sistema lógico razonable. La inferencia en la otra dirección, de '$\neg$ ' a '$\sim$', generalmente es verdadera sólo bajo la suposición cuestionable que el no demostrable implica falso.

Hay otras maneras de distinguir la negación-como-falla, '$\neg$', de la negación fuerte, '$\sim$'. En programación lógica, '¬ñormalmente recibe un significado contextual o significado 'local': fracaso o no demostrable se entiende en el contexto de un programa particular. Agrandando el programa, y una meta o consulta que previamente tuvo éxito puede fallar ahora, es decir, el razonamiento es revisable. En contraste, la negación fuerte tiene un carácter más global y estable: si $\sim \phi$ se declara como un hecho en el programa o banco de datos, entonces ese hecho nunca será meramente reversible al extender la base de datos. Sólo si la cadena de inferencia a $\sim \phi$ en el programa depende del fracaso de alguna otra proposición puede cambiar el valor de verdad cuando el programa se aumenta. Sin embargo, en los programas y sus extensiones donde solo '$\sim$'se permite aparecer, la inferencia será monotónica (o de razonamiento revisable).

Podríamos seguir fácilmente más allá la discusión de las propiedades de

estos dos estilos de negación, pero pronto estaremos obligados a que declaremos lo que queremos decir por "no demostrable"y "falso". Casi cada semántica diferente dada por programación lógica nos da una receta diferente para entender estos dos estilos de negación.

Debemos seleccionar un enfoque particular a la semántica de programación lógica, que de modelos estables, para analizar su tratamiento de negación desde un punto de vista completamente lógico. Es bastante extra´no que, el punto de vista lógico es a menudo el suprimido en programación lógica. La razón es que una semántica se define habitualmente especificando una cierta clase de modelos intencionales (por ejemplo los modelos mínimos de Herbrand, modelos estables, modelos bien-fundados), pero los modelos en cuestión son meramente entidades de conjuntos teóricos usados para interpretar el vocabulario del programa. La pregunta: ¿Para qué lógica estas entidades corresponden a modelos en el sentido usual? se suprime a menudo. Algunos casos son relativamente bien definidos, en otros menos. Por ejemplo, en algunos enfoques es supuesto claramente que uno está programando en lógica clásica, aunque quizás sólo una subclase propia del sentido usual que normalmente no se hace explícito. Los modelos del caso son particularmente sutiles e interesantes. Su presentación normal apenas los distingue de los modelos clásicos (cuando la negación fuerte está ausente en el idioma), llevando a la mayoría de los investigadores a considerar el razonamiento de modelos estables como su fortaleza del razonamiento clásico[Pea97].

Brevemente, podemos decir que una relación de inferencia $\mid\!\sim$ es *negación-estable* en una lógica L si para cualquier teoría T y la sentencia ψ, $T \mid\!\sim \psi$ si

y sólo si $\psi \in T'$, para todo T' tal que

$$T' = C_L(T \cup \{\neg\psi : \psi \notin T'\})$$

donde C_L es el funcionamiento de la consecuencia de L. Más bien, hablando informalmente podemos decir entonces que una negación $\neg$ (entendiendo wrt como alguna relación de inferencia $\mid\sim$ es estable sobre L si $\mid\sim$ es ella misma negación-estable en L, una característica del razonamiento clásico es que la lógica clásica es negación-estable sobre sí misma; la lógica intuicionista no lo es. Así nosotros podríamos decir que la negación clásica es clásicamente estable, aunque la negación de Heyting no es estable intuicionistamente. La novedad principal según Pearce [Pea97], es mostrar esa negación-como-falla entendido dentro del razonamiento del modelo estable es estable sobre la lógica intermedia de *áquí-y-allí"*. Éste es un rasgo distinguido de la semántica. Por ejemplo, es fácil ver esa negación-como-falla entendido en el paradigma del rival de semántica bien-fundada no es estable sobre cualquier lógica intermedia (aunque bien-fundada extiende la inferencia de la misma lógica de aquí-y-allí).

3.4 Lógica Intermedia

Una lógica intermedia se caracteriza por mantener los axiomas de la lógica intuicionista agregándole principalmente el siguiente axioma:

$$(\psi \Rightarrow \neg\varphi) \Rightarrow (((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi)$$

esto nos lleva a tener una lógica menos constructivista, dando lugar a las bases del razonamiento del equilibrio[Pea97].

Definición 26 *En una lógica (n+1)-multivaluada los conectivos* $\sim, \wedge, \vee, \Rightarrow$ *son evaluados como sigue:*

$$\sim \psi \; es \begin{cases} 0 & si \; \psi \; es \; n \\ n & en \; cualquier \; otro \; caso \end{cases}$$

$$\psi \wedge \varphi \; es \; max(\psi, \varphi)$$

$$\psi \vee \varphi \; es \; min(\psi, \varphi)$$

$$\psi \Rightarrow \varphi \; es \begin{cases} 0 & si \, \psi \geq \varphi \\ \varphi & si \, \psi < \varphi \end{cases}$$

Ejemplo 4 *Sea* $\psi = \sim (\sim A_2 \Rightarrow A_5)$ *es una lógica 3-valuada, entonces:*

Tabla 3.1: Tabla de valores de verdad de $\sim (\sim A_2 \Rightarrow A_5)$

A_2	A_5	$\sim (\sim A_2 \Rightarrow A_5)$
0	0	0
0	1	0
0	2	0
1	0	0
1	1	0
1	2	0
2	0	0
2	1	1
2	2	2

Propongamos y demostremos el siguiente lema para lógicas multivaluadas, como una generalización de la demostración en lógica clásica. Útil para demostrar propiedades de extensiones de teorías sobre lógicas intermedias.

Definición 27 *Una fórmula bien formada (wf) en una lógica multivaluada es definida inductivamente como sigue:*

1. *Si p es un símbolo proposicional, entonces p es una wf.*

2. *Si ψ y ϕ son wfs, entonces también lo son $(\sim \psi)$ y $(\sim \psi \longrightarrow \phi)$.*

Lema 8 *Sea ψ una wf en una lógica multivaluada y sea $B_1, \ldots, B_n$ letras proposicionales que ocurren en ψ. Entonces $B'_1, \ldots, B'_n \vdash \psi'$*[1]*, donde*

$$B'_i \text{ es} \begin{cases} \sim\sim B_i & \text{si } B_i \text{ no es falso} \\ \sim B_i & \text{si } B_i \text{ es falso} \end{cases}$$

$$\psi' \text{ es} \begin{cases} \sim\sim \psi_i & \text{si } \psi_i \text{ no es falso} \\ \sim \psi_i & \text{si } \psi_i \text{ es falso} \end{cases}$$

Demostración. Por inducción (ver [Joh]) en el número n de apariciones de los conectivos primitivos en ψ.

$(n = 0)$: ψ es B_1 entonces $\sim\sim B_1 \vdash \sim\sim B_1$ y $\sim B_1 \vdash \sim B_1$ es válido ya que $\alpha \vdash \beta$

$(j < n)$: Suponemos que es válido para $j < n$ por demostrar que es válido para n.

[1] En esta sección suponemos que trabajamos con una lógica intermedia, así usaremos $\vdash$ en lugar de $\vdash_{Int}$

Caso $\sim$: ψ es $\sim \varphi$. Entonces φ contiene $n-1$ conectivos lógicos primitivos.

1. φ no es *Falso* [2]. Entonces ψ es $Falso$.

   Por hipótesis de inducción sobre φ, $B'_1, \ldots, B'_n \vdash \varphi'$ con $\varphi' = \sim\sim \varphi$ implica $B'_1, \ldots, B'_n \vdash \psi'$, donde ψ' es $\sim \psi$, es decir, ψ' es $\sim\sim \varphi$

2. φ es *Falso*. Entonces $\sim\sim \psi$ es verdadero Por hipótesis de inducción sobre φ, $B'_1, \ldots, B'_n \vdash \sim \varphi$

   por lema ($\vdash \alpha \Rightarrow \sim\sim \alpha$), $B'_1, \ldots, B'_n \vdash \sim\sim\sim \varphi$ entonces $B'_1, \ldots, B'_n \vdash \psi'$ ya que ψ' es $\sim\sim \psi$, ψ' es $\sim\sim \varphi'$, φ' es $\sim \varphi$ y ψ' es $\sim\sim\sim \varphi$.

Caso $\Rightarrow$: ψ es $(\varphi \Rightarrow \phi)$. Entonces φ y ϕ tienen menos de n ocurrencias.

por hipótesis inductiva $B'_1, \ldots, B'_n \vdash \varphi'$ y $B'_1, \ldots, B'_n \vdash \phi'$

1. φ es $falso$ φ' es $\sim \varphi$, ψ es $\sim\sim \psi$

   así $B'_1, \ldots, B'_n \vdash \sim \varphi$ por hip. inductiva

   $B'_1, \ldots, B'_n \vdash \varphi \Rightarrow \phi$ por axioma (10) $\sim \alpha \Rightarrow (\alpha \Rightarrow \beta)$

   $B'_1, \ldots, B'_n \vdash \sim\sim (\varphi \Rightarrow \phi)$ por lemma $\alpha \Rightarrow \sim\sim \alpha$, pero $\sim\sim (\varphi \Rightarrow \phi)$ es ψ'.

2. ϕ no es $falso$ Entonces $\sim\sim \psi$ es $verdadero$, ϕ' es $\sim\sim \phi$

   ψ' es $\sim\sim \psi$, por lo tanto ψ' es $\sim\sim (\varphi \Rightarrow \phi)$

   $B'_1, \ldots, B'_n \vdash \sim\sim \phi$

   $B'_1, \ldots, B'_n \vdash \sim\sim (\varphi \Rightarrow \phi)$ por lema $\vdash \sim\sim \alpha \Rightarrow \sim\sim (\beta \Rightarrow \alpha)$

3. φ no es $verdadero$ y ϕ es $falso$

[2] bajo un asignamiento de valor de verdad dado

ψ toma el valor de $falso$, φ' es $\sim\sim \varphi$ ϕ' es $\sim \phi$ ψ' es $\sim \psi$

$B'_1, \ldots, B'_n \vdash \sim\sim \varphi$

$B'_1, \ldots, B'_n \vdash \sim \phi$

$B'_1, \ldots, B'_n \vdash \sim (\varphi \Rightarrow \phi)$ por lema $\vdash \sim\sim \alpha \Rightarrow (\sim \beta \Rightarrow \sim (\alpha \Rightarrow \beta))$

■

Capítulo 4

Independencia y Excepcionalidad

David Pearce[Pea97] han dado una nueva pauta a la programación lógica, proporcionando una nueva caracterización de las lógicas intermedias o lógicas de aquí y allí. es decir, lógicas que están entre lógica clásica y lógica intuicionista.

Para la investigación de las diferentes extensiones que se han realizado en las lógicas intermedias, es necesario proponer diferentes axiomatizaciones que nos ayuden a la transformación de programas lógicos. Pero para cada conjunto de axiomatizaciones propuesto es necesario verificar su independencia y formas excepcionales.

Hemos desarrollado este software llamado *"índ20"* que verifica la independencia de un axioma con respecto al resto de axiomas de una teoría. Dando lugar a encontrar sus formas excepcionales, utilizando para ello una lógica con más valores de verdad que la original y con la posibilidad de usar valores designados $0, 1, \ldots, m$ siendo $0 \leq m < n$.

4.1 Formas declarativas excepcionales

Aquí establecemos las nociones para verificar la independencia de axiomas y la obtención de formas declarativas excepcionales, a través de sus valores designados.

Definición 28 *Sean los números* $0, 1, 2, \ldots, n$ *"valores de verdad", y sea* $0 \leq m < n$. *Los números* $0, 1, 2, \ldots, n$ *son llamados:* "valores designados".

Si tomamos un número finito de "tablas de verdad"representando funciones de los conjuntos de la forma $\{0, 1, \ldots, n\}^k \longrightarrow \{0, 1, \ldots, n\}$. Para cada tabla de verdad, introducimos una señal, llamada: el *conectivo* correspondiente. Usando estos conectivos y letras declarativas, nosotros podemos construir "formas declarativas", y cada tal forma declarativa que contiene las j distintas letras definidas a una "función de verdad"de $\{0, 1, \ldots, n\}^j \longrightarrow \{0, 1, \ldots, n\}$.

Definición 29 *Se dice que una forma declarativa cuya función de verdad correspondiente sólo da valores designados es* éxcepcional".

Definición 30 *Los números* m, n *y las tablas de verdad básicas definen una* lógica multi-valuada (finita) M.

Definición 31 *Una teoría axiomática que involucra las letras declarativas y los conectivos de* M *se dicen ser* convenientes *para* M *si y sólo si los teoremas de la teoría coinciden obviamente con las formas de declaración excepcionales de* M.

Todas estas nociones pueden generalizarse al caso de un número infinito de valores de verdad. Si $n = 1$ y $m = 0$, y las tablas de verdad son aquéllas dadas por $\sim$ y $\supset$ en el ejemplo (1).

Definición 32 *Los wfs excepcionales en este caso son llamadas* tautologías.

4.2 Independencia de axiomas

Definición 33 *Dada una teoría axiomática, un subconjunto Ξ de axiomas se dice ser* independientes *si para alguna fórmula bien formada en Ξ no puede ser probada a través de las reglas de inferencia de la teoría, desde el resto de formulas de Ξ.*

La verificación de independencia de algún axioma $\phi \in \Xi$, bajo una lógica de n valores de verdad, lo realizamos utilizando una lógica de $n+1$ valores. Dando las propuestas posibles de tablas de verdad para cada conectivo lógico, seleccionamos desde $\{0\}$, $\{0,1\}$, $\{0,1,2\}$ hasta $\{0,1,\ldots,n\}$ ser los valores de verdad *"designados"*. Para cada una de estas propuestas buscamos las asignaciones de valores de verdad a conectivos, tal que para $\Xi - \{\phi\}$ sean *"designados"*, además de verificar que MP también mantenga los valores designados.

Ejemplo 5 *Sea la teoría formal expuesta en (Capítulo. 2), consideremos las siguientes tablas.*

Tabla 4.1: Tabla de valores de verdad para el conectivo $\sim$

A	$\sim A$
0	1
1	1
2	2

Tabla 4.2: Tabla de valores de verdad para el conectivo $\rightarrow$

A	B	$A \rightarrow B$
0	0	0
1	0	0
2	0	0
0	1	2
1	1	2
2	1	0
0	2	1
1	2	0
2	2	0

Tomando a 0 como el valor *"designado"*, vemos que los axiomas (A_2) $((\psi \Rightarrow (\varphi \Rightarrow \phi)) \Rightarrow ((\psi \Rightarrow \varphi) \Rightarrow (\psi \Rightarrow \phi)))$ y (A_3) $(((\sim \varphi) \Rightarrow (\sim \psi)) \Rightarrow (((\sim \varphi) \Rightarrow \psi) \Rightarrow \varphi))$ cumplen al ser designados (es decir, para cualquier valor de verdad obtenemos el valor "designado", por otro lado también MP preserva el valor designado, pero no así el axioma (A_1) $\psi \Rightarrow (\varphi \Rightarrow \psi)$. Por lo tanto el axioma (A_1) es independiente de (A_2) y (A_3) con MP como inferencia.

4.3 Descripción y ejemplos del software

En este trabajo realizamos la implementación de un algoritmo secuencial, que valide la independencia de axiomas basado en los antecedentes presentados en la sección (4.2).

Este software fue realizado en Delphi, bajo un entorno de programación de windows.

4.3.1 Algoritmo de independencia

Utilizamos el método de fuerza bruta, es decir, dado el conjunto de axiomas originales por el usuario y el axioma que se desea verificar su independencia (esto en forma infija), con el uso de la notación polaca inversa, primero generamos las tablas que validen los axiomas obteniendo el valor de designado todos ellos excepto el axioma que se desea su independencia, posteriormente verificamos que MP (Modus Ponens) preserve los valores designados obtenidos. Y si es así hemos logrado la independencia, y si no podemos inferir que son dependientes, para esto es necesario utilizar los métodos tradicionales de verificación de dependencia que consiste en demostrar a través de el resto de los axiomas el axioma dependiente. Recordemos que en una teoría esta constituida solo con axiomas dependientes.

4.3.2 Ejemplos

Ahora presentamos algunos ejecuciones realizadas por *ind20*.

Ejemplo 6 *En este ejemplo presentamos la independencia del axioma* (A_1) *de los axiomas* (A_2) *y* (A_3).

Número de Valores Lógicos : 3
Valores Característicos : 0

$$¿\ A_1 : a \to (b \to a) \text{ es Independiente?}$$

de

$$A_2 : (a \to (b \to c)) \to ((a \to b) \to (a \to c))$$

$$A_3 : (\ b \rightarrow\ \ a) \rightarrow ((\ b \rightarrow a) \rightarrow b)$$

Tabla 4.3: Tabla de Negación de ejemplo 6

A	$\sim A$
0	2
1	0
2	0

El Axioma A_0 si es Independiente

Tabla 4.4: Tabla de implicación del ejemplo 6

A	B	$A \rightarrow B$
0	0	0
0	1	2
0	2	2
1	0	2
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

Ejemplo 7 *En este ejemplo presentamos la independencia del axioma* (A_2) *de los axiomas* (A_1) *y* (A_3).

Número de Valores Lógicos : 3

Valores Característicos : 0

$$¿\ A_2 : (a \to (b \to c)) \to ((a \to b) \to (a \to c)) \text{ es Independiente ?}$$

$$A_1 : (\ b \to\ a) \to ((\ b \to a) \to b)$$

$$A_3 : a \to (b \to a)$$

Tabla 4.5: Tabla de Negación de ejemplo 7

A	$\sim A$
0	2
1	0
2	0

El Axioma A_0 si es Independiente

Ejemplo 8 *En este ejemplo presentamos la independencia del axioma* (A_3) *de los axiomas* (A_1) *y* (A_2).

Número de Valores Lógicos : 3

Valores Característicos : 0

$$¿\ A_3 : (\ b \to\ a) \to ((\ b \to a) \to b) \text{es Independiente ?}$$

Tabla 4.6: Tabla de implicación del ejemplo 7

A	B	$A \to B$
0	0	0
0	1	2
0	2	2
1	0	0
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

$$A_1 : a \to (b \to a)$$

$$A_2 : (a \to (b \to c)) \to ((a \to b) \to (a \to c))$$

Tabla 4.7: Tabla de Negación de ejemplo 8

A	$\sim A$
0	2
1	0
2	0

El Axioma A_3 si es Independiente

Ejemplo 9 *En este ejemplo presentamos la independencia del axioma* (A_n) : $((\sim A \to A) \to A)$ *de los axiomas* (A_1) *y* (A_2). *No encuentra ninguna asignación de valores de verdad que verifiquen la independencia, sin embargo no hay garantía que sea dependiente.*

Número de Valores Lógicos : 3

Valores Característicos : 0

Tabla 4.8: Tabla de implicación del ejemplo 8

A	B	$A \rightarrow B$
0	0	0
0	1	1
0	2	2
1	0	0
1	1	0
1	2	2
2	0	0
2	1	0
2	2	0

¿ $A_n : (\ a \rightarrow a) \rightarrow a$ es Independiente ?

$$A_1 : a \rightarrow (b \rightarrow a)$$

$$A_2 : (a \rightarrow (b \rightarrow c)) \rightarrow ((a \rightarrow b) \rightarrow (a \rightarrow c))$$

Tabla 4.9: Tabla de Negación de ejemplo 9

A	$\sim A$
0	2
1	0
2	0

A_n no es independiente

4.4 Paralelización del software

En esta sección damos una propuesta para paralelizar este algoritmo de fuerza bruta.

En la sección anterior describimos el desarrollo de un programa secuencial, donde notamos que el tiempo de respuesta para lógicas con muchos valores de verdad es tardado, eso nos lleva a proponer un algoritmo paralelo que resuelva dicho problema.

Podemos observar que el número de posibles asignaciones a un operador unario de n-valores de verdad es: n^n y para un operador binario es: n^{n^2}. En general el número de posibles asignaciones a un operador m-ario de n-valores de verdad es: n^{n^m}.

Esto explica la razón de la demora en el tiempo de respuesta. Por lo tanto seria conveniente paralelizar dicho algoritmo.

Proponemos utilizar la técnica de divide y vencerás, donde podemos repartir los n^{n^m} posibles valores de verdad de cada tabla de verdad entre p procesadores, logrando con esto una carga balanceada y tomando en cuenta que el tiempo de comunicación es mínimo. Lograríamos una *eficiencia*[KGGK94] casi de 1 (100%).

$$eficiencia = \frac{t_{\frac{secuencial}{t_p}}}{p},$$

donde t es tiempo y p es el número de procesadores.

Podemos realizar una implementación económica de este algoritmo paralelo, utilizando para ello un cluster basado en algún entorno paralelo como PVM o MPI.

Conclusión

En este trabajo hemos realizado la demostración de 2 lemas importantes que no se encuentran en la bibliografía actual, como es expresado por David Pearce. La extensión completa y consistente de teorías bajo lógica intuicionista, nos dice que toda teoría intuicionista es completable, bien es sabido que si se completa con literales positivas llegamos a la semántica estable, pero este resultado va más allá, ya que en las lógicas intermedias podemos completar no solo con literales positivas, sino también agregando literales negadas, esto nos establece nuevas líneas de investigación en programación lógica. Por lo tanto la búsqueda de nuevas axiomáticas intermedias plantea un enfoque nuevo de atención de los investigadores en el área.

Bibliografía

[ADO99] J. Arrazola, J. Dix, and M. Osorio. Confluent rewriting systems in non-monotonic reasoning. *Computacion y Sistemas*, 1999.

[BD94a] S. Brass and J. Dix. *Characterizations of the Disjunctive Stable Semantics by Partial Evaluation.* Elsevier Science, 1994.

[BD94b] S. Brass and J. Dix. *Semantics of Disjunctive Logic Programs based on Partial Evaluation.* Elsevier Science Inc, 1994.

[BDK97] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic Reasoning, An Overview.* CSLI Publications, 1997.

[Dal89] Dirk Van Dalen. *Logic and Structure.* Springer-Verlag, 1989.

[DFN99] J. Dix, U. Furbach, and I.Ñiemela. *Nonmonotonic Reasoning: towards efficient calculi and Implementations.* Elsevier Science Publishers B.V., 1999.

[Dix95] J. Dix. *Semantics of Logic Programs: Their Intuitions and Formal Properties.* André Fuhrmann and Hans Rott, Eds., 1995.

[Gab91] D. M. Gabay. *Modal provability foundation for negation by failure.* Springer-Verlag, 1991.

[Gel87] M. Gelfond. *On stratified auto-epistemic theories*. In proceeding of AAAI-87, 1987.

[HU96] J. E. Hopcroft and J. D. Ullman. *Introducción a la teoria de autómatas, lenguajes y computación*. ed. cecsa, 1996.

[Joh] R. Johnsonbaugh. *Matemáticas Discretas.*

[JYTL89] Girard Jean-Yves, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, Cambridge, 1989.

[KGGK94] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing*. The bejamin/cumming publishing company, inc, 1994.

[Llo87] Lloyd. *Logic Programming*. Springer-Verlag, Berlín, 1987.

[Men87] Elliott Mendelson. *Introduction to Mathematical logic*. Wadsworth and Brooks/Cole Advanced Books, 1987.

[Pea97] David Pearce. *From here to there: Stable negation in logic programming*. D Gabbay y H Wansing, 1997.