VNIVERSIDAD NACIONAL
AVTÓNOMA DE
MEXICO

Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería en Computación.

Deducción Natural en Lógica Modal: una implementación en Coq.

T E S I S

Que para optar por el grado de:
Maestra en Ciencias (Computación)

P r e s e n t a:

Pilar Selene Linares Arévalo

Directora de tesis:
Dra. Atocha Aliseda Llera
Instituto de Investigaciones Filosóficas, UNAM.

Codirector de tesis:
Dr. Favio Ezequiel Miranda Perea
Facultad de Ciencias, UNAM.

México, D.F Mayo 2015

Agradecimientos

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la Universidad Nacional Autónoma de México, en el marco del proyecto:

"Lógicas del descubrimiento, heurística y creatividad en las ciencias"
(PAPIIT, IN400514)

agradezco a la DGAPA-UNAM por la beca recibida.

También agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca que me otorgó de 2012 a 2014.

Índice general

Introducción

Un ejemplo excepcional del razonamiento matemático tiene lugar en las pruebas[1] matemáticas. La formalización de este tipo de razonamiento ha motivado el uso de las computadoras para modelarlo y automatizarlo en la medida de lo posible. Sin embargo, esta tarea requiere de la descripción explícita y precisa de los procedimientos involucrados en el razonamiento matemático. Esto es, todos los pasos y reglas empleadas en los procedimientos deben ser explícitos y absolutamente especificados, sin que quede oculto o sobreentendido ningún elemento. En principio, todos los enunciados matemáticos pueden ser escritos en algún sistema formal, y la comprobación de la validez de sus demostraciones si bien debería ser una verificación sintáctica simple, puede resultar tediosa. Es por esto que la formalización de las matemáticas se ha apoyado en la computación, intentando dejar esa labor de verificación a sistemas computacionales, que sabemos son adecuados para ejecutar de forma rápida rutinas sencillas, monótonas y fastidiosas. En este momento es donde entran en juego tanto los sistemas de demostración automática como los asistentes de pruebas, que si bien todos ellos persiguen el mismo objetivo de la demostración de teoremas y la construcción de sus pruebas matemáticas, los mecanismos de funcionamiento para cada uno de ellos son distintos.

Por un lado, los demostradores automáticos de teoremas tienen como objetivo construir pruebas de enunciados de forma automática y autónoma, y proporcionan un método rápido y fiable para la verificación de la validez de un argumento. Sin embargo, lo que sigue siendo difícil es convertir los argumentos escritos en papel al lenguaje formal que emplea el demostrador, además de que la tarea de decidir si el argumento es válido puede variar de ser un trabajo trivial a uno imposible.

Por otro lado, los asistentes de pruebas son herramientas diseñadas para desarrollar demostraciones formales a través de la colaboración entre usuario y el programa. Estos asistentes de pruebas contienen un verificador automático en el núcleo de su implementación, el resto del software da soporte al desarrollo interactivo de pruebas formales, administrando los enunciados que deben ser demostrados y las hipótesis disponibles para ello; también proporcionan herramientas para resolver de forma automática o simplificar ciertas metas. Algunos ejemplos de formalización y construcción de demostraciones matemáticas utilizando asistentes de pruebas, incluyen la verificación del teorema de la curva de Jordan y la demostración de la conjetura de Kepler por Thomas Hales, la formalización del Teorema fundamental del Álgebra por Geuvers, Pollack, Wiedijk y Zwanenburg, la formalización constructiva del Teorema fundamental del Cálculo por Cruz-Filipe, y la demostración del Teorema de los cuatro colores por George Gonthier.

Esta tesis se sitúa en el campo de la lógica computacional y en este sentido su objetivo es realizar una aportación a la formalización de las matemáticas -en particular en la Lógica- ofreciendo una formalización de un sistema de deducción natural para la lógica modal intuicionista, así como la demostración de diversas propiedades que éste cumple, todo esto utilizando precisamente un asistente de pruebas. La decisión de abordar la lógica modal intuicionista está motivada por su interpretación

[1] En esta tesis se emplea la palabra *prueba* como traducción de la palabra en inglés *proof*.

computacional y sus aplicaciones (que mencionaré en breve). El objetivo de los siguientes párrafos es ofrecer un breve panorama tanto del asistente de pruebas que utilizo en esta investigación, como del sistema lógico del que me ocupo.

El asistente de pruebas empleado en esta investigación es Coq [29]. Este sistema pertenece a una gran familia de herramientas computacionales cuyo propósito es proporcionar asistencia en la demostración de teoremas. Coq es un asistente de pruebas en el cual se pueden expresar especificaciones y desarrollar programas que las satisfagan; también es un sistema en el que se pueden construir demostraciones matemáticas sobre una lógica de orden superior. Estas pruebas se construyen de forma interactiva con la ayuda de herramientas de búsqueda automática cuando es posible. Además, Coq puede ser utilizado como un *marco lógico* [2], en el que se definen axiomas y reglas de inferencia de nuevas lógicas para desarrollar pruebas de teoremas sobre éstas.

Antes de entrar en la descripción del sistema lógico con el que trabajo, dedico unas líneas al intuicionismo, una de las tres principales corrientes identificadas en la filosofía de las matemáticas y que es considerado como una forma de implementar el constructivismo en matemáticas. El constructivismo es el enfoque que toma la postura según la cual todo objeto matemático es una construcción mental, por lo que debe existir una forma o método para construirlo. Es por esto que el intuicionismo es más restrictivo que el razonamiento clásico, por ejemplo, probar que la no existencia de un objeto deriva un absurdo $(\varphi \wedge \neg\varphi)$, no equivale a probar la existencia del mismo; también rechaza la ley del tercero excluido $(\varphi \vee \neg\varphi)$, pues ésta garantiza que alguna de las fórmulas φ o $\neg\varphi$ es verdadera, aún cuando no sea posible verificar cuál de las dos fórmulas se puede construir. Por lo tanto, en la lógica intuicionista más allá de afirmar que un objeto existe, se está obligado a construirlo o al menos mostrar una prueba directa de él. Estas pruebas pueden ser formalizadas en algún sistema axiomático de deducción formal o en algún sistema de deducción natural. En contraste con los sistemas deductivos en estilo Hilbert, caracterizados por un conjunto de axiomas y pocas reglas de inferencia, los sistemas de deducción natural consisten en un único axioma y varias reglas de inferencia (también llamadas reglas de derivación); en esta tesis trabajaremos con este último tipo de sistema deductivo.

La Deducción Natural fue introducida por Gerhard Gentzen (para ahondar en la historia de la Deducción Natural se puede consultar [22]), con la motivación de capturar formalmente la estructura lógica que tienen las argumentaciones deductivas que se llevan a cabo en el razonamiento humano, en particular, en las demostraciones matemáticas. La esencia de este enfoque consiste en considerar cada conectivo lógico por separado, proporcionándole un significado[3] intrínseco.

A continuación, describo brevemente la lógica modal intuicionista (LMI). Inicio con unas palabras sobre la noción de modalidad. Una expresión u operador modal (por ejemplo: "Es posible que llueva esta noche","Necesariamente, o el café está frío o no lo está", "Ellas se reencontrarán en el futuro" y "El investigador sabe quién cometió el crimen") se utiliza para caracterizar la verdad de un juicio o una actitud proposicional ("Yo creo que hoy es Viernes"). En general, el término *lógica modal* se utiliza para referirse a cada uno de los miembros de una familia de lógicas que comparten reglas de inferencia (como la regla de Necesitación y el Modus ponens) y que capturan el comportamiento de algún grupo de operadores modales. Ejemplo de estas lógicas, cada una abordando diferentes operadores modales son: la lógica deóntica que cubre las modalidades concernientes a los conceptos de obligación y per-

[2]Es importante aclarar que cuando se habla de marco lógico, hacemos referencia a lo que en inglés se conoce como logical framework. El concepto de marco lógico que se emplea en este trabajo no debe confundirse con marcos de Kripke

[3]Intuitivamente cada conectivo $\star$ está definido de la siguiente forma: por la manera en que se prueba que una fórmula es verdadera si su conectivo principal es $\star$ (información que se refleja y corresponde a las reglas de introducción), y por los elementos o información que se obtienen al descomponer dicha fórmula (lo cual corresponde a las reglas de eliminación).

miso, la lógica temporal que representa y razona sobre proposiciones calificadas en términos de tiempo, y la lógica epistémica que da un tratamiento formal a las nociones de conocimiento y creencia. La lógica modal estrictamente hablando, es el estudio del comportamiento deductivo de las expresiones "es necesario que" y "es posible que". Esta tesis trata exclusivamente sobre este sistema en donde los símbolos $\Box$ y $\Diamond$ representan los operadores modales de necesidad y posibilidad.

La lógica modal intuicionista (LMI) es entonces una extensión de la lógica intuicionista (constructiva) que estudia el comportamiento de alguna interpretación de los operadores modales. Las lógicas modales constructivas junto con las teorías de tipos se han vuelto relevantes para las ciencias de la computación, tanto en sus fundamentos teóricos como en sus aplicaciones.

Algunos ejemplos de aplicaciones de la lógica modal intuicionista en ciencias de la computación se encuentran en los siguientes trabajos: el desarrollo de la captura de una noción de *bisimilitud* de procesos divergentes por Stirling en [26], el empleo de una lógica modal intuicionista para razonar sobre el comportamiento de circuitos lógicos por Fairtlough y Mendler en [11], y el uso de una versión intuicionista de la lógica modal S4 para definir un lenguaje de programación con constructores de tiempo ligados por Davies y Pfenning en [23].

Como es sabido, existen diferentes concepciones formales de la lógica [1] por ejemplo: la sintáctica, la semántica y la estructural. La primera de ellas es la que constituye hoy en día lo que conocemos como Teoría de la demostración, la segunda conforma la teoría de modelos y la tercera es un enfoque que pretende caracterizar un sistema formal por medio de las reglas estructurales que cumple. Este último enfoque está inspirado en los trabajos de consecuencia lógica de Tarski y aquellos sobre deducción natural de Gentzen.

Existe una estrecha relación entre la formalización de las matemáticas, las pruebas constructivas y la teoría de la demostración, esta última vista como el estudio de los sistemas deductivos y sus propiedades. Una rama de la teoría de la demostración, conectada estrechamente con la tercera concepción de la lógica citada en el párrafo anterior, la llamada *teoría de la prueba estructural*, se encarga del estudio de propiedades estructurales de sistemas deductivos; esto es relevante en áreas de la computación como la deducción automática. Es también bajo este enfoque de la teoría de la demostración, en que se desarrolla esta tesis, por lo que el planteamiento y tratamiento de los conceptos son de naturaleza sintáctica.

La presente investigación se sitúa por lo tanto, en el centro de la teoría de la prueba estructural y desde esta perspectiva aborda una lógica modal intuicionista a través de un sistema de deducción natural, su formulación mediante secuentes y la demostración de sus propiedades. En particular se trata de una lógica minimal en la que, como se observará, el operador de negación no es necesario. En el artículo de Pfenning y Davies [24] se hace un análisis de los fundamentos de la lógica modal intuicionista, empleando la metodología de Martin-Löf, que consiste en hacer una distinción entre *juicios* y *proposiciones*. Esto da pie a un sistema de deducción natural para la lógica modal intuicionista (al que he nombrado DLMI) y el cual es la base de este trabajo.

El objetivo principal de esta tesis es ofrecer una formalización y demostración de propiedades del sistema DLMI en el asistente de pruebas COQ. Cabe mencionar que tanto la formalización como la demostración de propiedades no son un trabajo trivial. En primer lugar se debe proponer una representación de la lógica modal intuicionista en el lenguaje del asistente de pruebas. En segundo lugar hay que construir la demostración de propiedades del sistema de deducción. En particular, demostrar propiedades estructurales como *contracción, debilitamiento* e *intercambio*, conlleva a verificar

propiedades básicas que se dan por entendidas en las demostraciones en papel, pero que en la formalización en el asistente de pruebas deben ser enunciadas y formalizadas. Ejemplo de éstas son las propiedades de equivalencia de contextos, propiedades de conjuntos e igualdad de elementos.
En [24], Pfenning y Davies no especifican cuál es la lógica subyacente al sistema de deducción con el que se trabaja. Así uno de los retos de este trabajo es identificar a través de la demostración de diversas propiedades y axiomas, de qué lógica se trata.

Esta tesis está organizada de la siguiente forma: en el primer capítulo expongo dos temas que permiten apreciar la dificultad de formalizar de forma directa, la lógica modal (en nuestro caso particular, intuicionista) sobre un marco lógico (Coq). Presento un breve panorama de los marcos lógicos, en donde se aclara que éstos son adecuados para la codificación de formalismos de deducción natural, por lo que resulta complicado implementar de forma directa, una lógica como la modal intuicionista. Después abordo el tema de la lógica modal y el teorema de la deducción, indico que hablar de la invalidez de este teorema en lógica modal es incorrecto, pues los argumentos a favor de la invalidez están basados en una interpretación incorrecta de este teorema.

En el segundo capítulo desarrollo el sistema de deducción natural para la lógica modal intuicionista. Se inicia con la formulación de los fundamentos de la LMI, utilizando la metodología de Martin-Löf; se presentan los conceptos de *juicio hipotético*, *juicio categórico*, *derivación o prueba* y la definición de las modalidades de *necesidad* y *posibilidad*. Se presenta el sistema de deducción y se demuestra que cumple las propiedades estructurales usuales de contracción, debilitamiento e intercambio, así como un teorema de sustitución.

En el capítulo cuatro, se encuentra el desarrollo de la formalización de la lógica de la que se ha hablado, en el asistente de pruebas Coq. Primero se presentan los aspectos generales de los asistentes de prueba para después hacer una breve introducción a Coq y sus características. Se continúa con el análisis de la formalización del sistema de deducción, presentando fragmentos de código de la implementación, con los cuales se va explicando la forma en que trabaja el asistente de pruebas; por supuesto, se abordan propiedades que cumple el sistema y presento sus demostraciones.

Para finalizar esta investigación, presento una sección de conclusiones y trabajo a futuro. En este capítulo se encuentran de forma resumida los resultados obtenidos, en particular, se responde a la pregunta de a qué lógica corresponde el sistema de deducción con el que se ha trabajado. También se pone sobre la mesa la semántica de la lógica que subyace al sistema de deducción. Dicha semántica causó interés durante el desarrollo de esta tesis, y se plantea como una de las posibles líneas de investigación sobre trabajo futuro.

Capítulo 1

Preliminares

En este capítulo se presentan, antes de entrar de lleno al sistema de deducción natural y su formalización, dos temas importantes: Marcos lógicos y el Teorema de la Deducción en lógica modal. En el capítulo anterior se ha dicho que el objetivo de este trabajo es desarrollar una formalización de un sistema de deducción natural para la lógica modal intuicionista en Coq; lo que significa que se empleará este asistente de pruebas como un marco lógico, en el que se codificará la especificación del sistema deductivo para así poder construir pruebas de teoremas y propiedades del mismo.

A través de las siguientes secciones, se busca exponer los motivos por los cuales la lógica modal se considera difícil de formalizar de forma directa en un marco lógico.

1.1. Marcos lógicos

Un marco lógico[1] se define como un metalenguaje para la especificación y verificación de sistemas deductivos. Los sistemas deductivos son sistemas formales dados a través de axiomas y reglas de inferencia que definen juicios derivables. Son herramientas comunes en lógica matemática y ciencias de la computación y se utilizan para especificar, desde diferentes lógicas, teorías lógicas, hasta aspectos de lenguajes de programación como los sistemas de tipos, la semántica operacional, máquinas abstractas o compilación. Los marcos lógicos están sujetos a los mismos principios de diseño que los lenguajes de programación. Deben ser simples y uniformes, proporcionando medios concisos para expresar los conceptos. También deben poder estructurar grandes especificaciones, verificando que sus componentes sean concordantes. Finalmente deben detectar de forma estática, expresiones que no están en el lenguaje y carecen de sentido. Pero también hay aspectos que se deben garantizar y que los caracterizan como marcos lógicos, dos de los más importantes son: en primer lugar, una implementación debe ser capaz de verificar deducciones de validez respecto a la especificación de un sistema deductivo; en segundo lugar, debe ser posible demostrar que las representaciones de los sistemas deductivos en el marco lógico, son suficientes y adecuadas, de tal forma que se pueda garantizar que las derivaciones formales son correctas.

Históricamente, la primera implementación de un marco lógico fue *Automath* [20] y sus diversos lenguajes, que se desarrollaron a finales de los años sesenta y principios de los setenta. El objetivo del proyecto era proporcionar una herramienta para la formalización y automatización de las matemáticas que fuera lo más general posible, es decir, que estuviera ligada lo menos posible a cualquier conjunto de reglas para la lógica o fundamentos de las matemáticas. También se buscaba que la forma en la que

[1] Es importante aclarar que cuando se habla de *marco lógico*, hacemos referencia a lo que en inglés se conoce como *logical framework*. El concepto de marco lógico que se emplea en este trabajo no debe confundirse con marcos de Kripke.

el material matemático fuera presentado en el sistema, correspondiera a la forma usual en la que se escriben las matemáticas. Durante el progreso de este proyecto, se observó que la lógica que subyace a un desarrollo matemático particular es parte integral de su formalización.

Muchas de las ideas de los lenguajes desarrollados para Automath se pueden ver plasmadas en los sistemas modernos. A principios de los años ochenta, la importancia de las teorías constructivas de tipos para las ciencias de la computación fue reconocida a través del trabajo del pionero Martin Löf [19]. Por un lado, esto condujo a una serie de sistemas para las matemáticas constructivas y la extracción de programas funcionales a partir de las demostraciones constructivas; por otro lado, influyó fuertemente el diseño del marco lógico Edinburgh, que se desarrollaba a la par de marcos lógicos (en forma de demostradores de teoremas) basados en lógica de orden superior y resolución binaria, así como lenguajes de programación lógicos. Más tarde los enfoques de programación lógica y teoría de tipos se combinarían en un nuevo lenguaje para *ELF* (Edinburgh logical framework).

A mediados de los años noventa, surgieron implementaciones de marcos lógicos basadas en definiciones inductivas, definiciones inductivas parciales, lógica de reescritura y sistemas deductivos etiquetados. Desde entonces un número considerable de marcos lógicos se han propuesto e implementado para diferentes propósitos; ejemplo de ello son[2]: Maude, Twelf, Pi, Isabelle, Coq, Lego, Elan, etc.

Los marcos lógicos son adecuados para la codificación de formalismos de deducción natural. En particular, la lógica modal se considera difícil de implementar de una forma directa. Encontramos en la literatura implementaciones de la lógica modal en marcos como *ELF* e *Isabelle*. Sin embargo, o bien siguen el estilo Hilbert [18] o son muy especializadas [4] y su corrección es sutil.

1.2. Lógica modal y el Teorema de la Deducción

A continuación presentamos una breve discusión (basada en lo expuesto en [16]) de la razón por la cual es complicado abordar la lógica modal mediante un sistema de deducción natural: la validez del teorema de la deducción. Se empleará la notación de secuentes, los cuales son expresiones de la forma $\Gamma \vdash A$ y se interpretan como: la fórmula A se deriva a partir del conjunto de fórmulas Γ.

El teorema de la deducción es un metateorema en lógica matemática que establece que si una fórmula B puede ser derivada a partir del conjunto de fórmulas $\Gamma \cup \{A\}$, entonces $A \to B$ se puede derivar a partir de Γ. Lo anterior se representa en términos de secuentes como:

$$\text{Si } \Gamma, A \vdash B \text{ entonces } \Gamma \vdash A \to B$$

Los sistemas de Hilbert estándar para la lógica modal, por lo general se componen, además de los esquemas axiomáticos, de dos reglas de inferencia: modus ponens y la regla de necesitación. Se afirma en distintas fuentes [25], [12], [8], que el teorema de la deducción, en la forma en la que se formula usualmente, no es válido en lógicas modales que contienen la regla de necesitación: a partir de A se infiere $\Box A$. Al analizar los argumentos que afirman la invalidez del teorema de la deducción en lógicas modales, todo indica que el problema surge al interpretar de forma inadecuada la derivabilidad a partir de un conjunto de fórmulas, en un sistema axiomático. Los siguientes párrafos exponen dos de estas interpretaciones inadecuadas.

[2] En [28], el lector puede encontrar una lista extensa de marcos lógicos, desde los precursores como automath hasta los contemporáneos como Coq e Isabelle.

La primera interpretación errónea afirma que la regla de necesitación

$$\frac{A}{\Box A} \quad \text{admite la derivación} \quad A \vdash \Box A\ .$$

Sin embargo, $A \to \Box A$ no es un teorema, invalidando así el teorema de la deducción. Este razonamiento resulta sospechoso pues las presentaciones de la regla de necesitación que encontramos en los libros de texto [9], enfatizan que el significado de esta regla es: $\vdash \Box A$ siempre que $\vdash A$, o lo que es lo mismo, si A es un teorema entonces $\Box A$ también lo es.

Otra interpretación errónea consiste en entender la derivabilidad a partir de un conjunto de fórmulas en un sistema axiomático, como la adición de éstas al conjunto de axiomas. Supóngase que se quiere demostrar que $A \to \Box A$ en algún sistema axiomático modal, en el que se cumple el teorema de la deducción, a partir del secuente $A \vdash \Box A$. Si se agrega A a la lista de axiomas, entonces se puede aplicar la regla de necesitación y concluir $\Box A$ a partir de A, es decir $A \to \Box A$. Sin embargo, esto es incorrecto.
En los sistemas de Hilbert tradicionales, no hay noción de *supuestos* (suposiciones) que no sean un axioma, y en las reglas de inferencia siempre se tienen axiomas o teoremas como premisas, lo cual garantiza la corrección de la regla de necesitación. De aquí que, si se consideran las hipótesis como axiomas y se emplea la regla de necesitación sin restricciones, se puede inferir $\Box A$ a partir de A, y la formulación del teorema de la deducción, falla.

Melvin Fitting en [13] propone hacer una modificación a la noción de derivabilidad de tal forma que se realice una distinción entre dos tipos de premisas, de tal forma que la regla de necesitación se aplique sólo a uno de estos tipos. Se distingue entre *premisas globales* y *premisas locales*: las primeras son los esquemas axiomáticos o bien verdades tautológicas obtenidas a partir de los axiomas; las segundas se refieren a verdades contingentes. La regla de necesitación debe ser aplicada únicamente a premisas globales, no locales.

La idea anterior se refleja en la siguiente definición de derivabilidad expuesta en [13] pero expresada con la notación de secuentes que se utiliza a la largo de esta tesis:

Definición 1. *Sea $\mathcal{L}$ un conjunto de esquemas de axiomas para una lógica modal,* S *y* U *conjuntos de fórmulas (donde los elementos de* U *no son axiomas), y* F *una fórmula que no es un axioma. La fórmula* F *es derivable (en $\mathcal{L}$) a partir del conjunto* S *de premisas globales y del conjunto* U *de premisas locales, si hay una secuencia de secuentes -resultado de aplicaciones de reglas de derivación- que termina en* F*.*
En la parte global cada fórmula es una instancia de un miembro de $\mathcal{L}$, un miembro de S *o se obtiene a partir de fórmulas de estos conjuntos por modus ponens o necesitación. En la parte local cada fórmula es una instancia de un miembro de $\mathcal{L}$, un miembro de* U*, o se obtiene a partir de fórmulas de estos conjuntos por modus ponens (la regla de necesitación no está permitida en esta parte).*

Si F *es derivable a partir de los conjuntos* S *y* U *se representa como* $\mathrm{S}; \mathrm{U} \vdash \mathrm{F}$*.*

Con esta definición, de acuerdo al planteamiento de Fitting, se rescata el teorema de la deducción.

Dado que en la lógica modal intuicionista con la que se trabajará es necesario que se valga la regla de necesitación, es importante tener cuidado en su formalización. Una de las motivaciones para utilizar un sistema de deducción con contextos (hipótesis localizadas) y no un sistema etiquetado, es que los contextos permitirán un manejo sencillo de los conjuntos de premisas globales y locales.

Además en la sección 1.1 se ha mencionado que los marcos lógicos son adecuados para la codificación de formalismos de deducción natural, aquellos cuya metateoría no se aleja de la metateoría del marco lógico, una razón más para emplear un sistema con contextos pues como ya se ha indicado, el marco lógico a utilizar es Coq. En el siguiente capítulo se muestra un sistema de deducción natural con contextos para la lógica modal intuicionista, que satisface el teorema de la deducción, entre otras propiedades.

Capítulo 2

Deducción Natural para la Lógica Modal

En este capítulo se presenta y discute el sistema de Deducción Natural para la lógica modal intuicionista, propuesto por Frank Pfenning y Rowan Davies en [24]. La idea central es seguir la metodología de Martin-Löf, la cual propone hacer una distinción entre juicios y proposiciones. A partir de ésta se formula el significado constructivo para las modalidades de necesidad y posibilidad, el cual da pauta a un sistema de deducción natural.

2.1. Juicios y proposiciones

Martin-Löf proporciona fundamentos para la lógica basados en una distinción entre juicios y proposiciones. Establece que juzgar es conocer y un juicio evidente es un objeto de conocimiento, de tal forma que una prueba o demostración es lo que hace a un juicio evidente.

Se trabaja principalmente con juicios de la forma *A es una proposición* o *A es verdadera*, para éste último suponiendo que A es una proposición. Bajo esta idea, se interpreta que el hecho de saber que *A es una proposición*, significa que se conoce aquello que verifica a A. Por otro lado, saber que *A es verdadera* significa que se sabe cómo verificar A.

En la caracterización de la lógica por medio de juicios, se utilizan dos reglas por cada conectivo u operador; estas reglas son conocidas como de introducción y de eliminación en los sistemas de Deducción Natural introducidos por Getzen en 1935. El significado de una proposición está dado por aquello que cuenta como una verificación de la misma, es decir, se obtiene de los objetos que son utilizados para demostrarla, lo cual se refleja en las reglas de introducción que permiten concluir cuándo una proposición es verdadera; estas reglas de introducción se complementan con las reglas de eliminación que proporcionan una forma de obtener información al descomponer una proposición.

Hasta ahora hemos mencionado los juicios de la forma *A es una proposición* y *A es verdadera*, sin embargo, éstos no son suficientes para demostrar el esquema correspondiente a la implicación, puesto que nos gustaría afirmar que $A \to B$ es verdadera si siempre que A es verdadera B también lo es. Para esto se introducen a continuación, los juicios y pruebas hipotéticas. El esquema general de un juicio hipotético es

$$J_1, \ldots, J_n \vdash J$$

el cual expresa que se deriva J asumiendo verdaderas cada una de las hipótesis del conjunto$\{J_1, \ldots, J_n\}$, es decir tenemos J a partir de las hipótesis J_1 hasta J_n. En una prueba hipotética del juicio anterior,

podemos utilizar cada una de las hipótesis J_i dándolas por hecho; en consecuencia podemos sustituir una derivación arbitraria de J_i para obtener un juicio que no depende de J_i. La forma particular del juicio hipotético que necesitamos es

$$A_1 \; verdadera, \ldots, A_n \; verdadera \vdash A \; verdadera$$

El principio de sustitución para esta forma de juicios con hipótesis se formula como sigue:

Principio de sustitución sobre hipótesis verdaderas

$$\text{Si} \quad \Gamma, A \; verdadera \vdash J \quad \text{y} \quad \Gamma \vdash A \; verdadera \quad \text{entonces} \quad \Gamma \vdash J$$

En el desarrollo del presente trabajo nos interesaremos particularmente en los casos donde el juicio J es de la forma C *verdadera*, por lo que en un secuente que deriva el juicio J escribiremos C en vez de C *verdadera*. Asimismo, cuando se hace explícita una fórmula de la lista de hipótesis, escribiremos A en vez de A *verdadera*.

Existen juicios que no dependen de hipótesis que afirman la verdad de proposiciones, es decir, juicios que son ciertos independientemente de cualquier conjunto de hipótesis verdaderas, a éstos les llamaremos juicios categóricos. A continuación se introduce el juicio de que A es válida, que se denota por A *válida*, presuponiendo que A es una proposición. La evidencia para la validez de A es la evidencia incondicional de A, lo cual se refleja en la siguiente definición:

Definición de validez:

1) Si $\cdot \vdash A$ entonces A *válida*
2) Si A *válida* entonces $\Gamma \vdash A$

es decir, A es válida si es derivable a partir de un conjunto vacío de hipótesis, además si A es válida entonces es verdadera bajo cualquier contexto. El siguiente paso es incorporar hipótesis de la forma A *válida* en los juicios hipotéticos, dado que el orden es irrelevante, se separan hipótesis verdaderas de válidas, obteniendo el siguiente esquema

$$B_1 \; válida, \ldots, B_m \; válida \mid A_1 \; verdadera, \ldots, A_n \; verdadera \vdash A \; verdadera$$

En lo que resta de este trabajo emplearemos Δ para representar un conjunto de suposiciones válidas y Γ un conjunto de suposiciones verdaderas, por lo que cuando se trata de hipótesis, escribiremos únicamente la proposición que se supone es verdadera o válida. En cuanto al consecuente del juicio, se hace la restricción de probar elementos de la forma A *verdadera*; esto es posible pues A *válida* está definido directamente en términos de verdad.

El significado de los juicios categóricos recae en el *principio de sustitución general*:

$$\text{Si} \quad \Delta \, , \, B \vdash J \quad \text{y} \quad \Delta \vdash B \quad \text{entonces} \quad \Delta \vdash J$$

Como ya se ha dicho, se concluirán únicamente juicios de verdad, por lo que reescribiendo la primera parte y agregando hipótesis verdaderas, obtenemos la siguiente versión del **principio de sustitución sobre hipótesis válidas**, utilizada en el resto de esta tesis:

$$\text{Si} \quad \Delta, B \mid \Gamma \vdash J \quad \text{y} \quad \Delta \mid \cdot \vdash B \quad \text{entonces} \quad \Delta \mid \Gamma \vdash J$$

Obsérvese que el principio de sustitución no es visto como una regla de inferencia, sino que define una propiedad sobre los juicios hipotéticos que se utiliza en el diseño del sistema formal.

2.2. Sistema de Deducción Natural

Debido a que diversas aplicaciones de la lógica modal únicamente requieren del concepto de necesidad, el sistema que se presenta a continuación, incluye reglas de introducción y eliminación para el conectivo de implicación y el operador de necesidad.

La primera regla que se presenta, es la regla general para el uso de hipótesis:

$$\frac{}{\Delta \mid \Gamma, A, \Gamma' \vdash A}\ (tHyp)$$

La regla $tHyp$ afirma que de suponer A verdadera, se deriva que A es verdadera. En lo que a las hipótesis válidas se refiere, también se tiene una regla del uso de éstas, expresada en términos de verdad y cuya forma se justifica a partir de la definición de validez. La idea detrás de la regla $vHyp$ es la siguiente: si B pertenece a un conjunto de hipótesis válidas, entonces a partir de este conjunto se deriva la validez de B, por definición, sabemos que si B es válida entonces se deriva B *verdadera* a partir de cualquier conjunto de hipótesis verdaderas pero del mismo conjunto de hipótesis válidas, a partir de las cuales se derivó B *valid*. Como las reglas de inferencia tienen la restricción de derivar únicamente juicios de verdad, la regla del uso de hipótesis válidas queda como sigue:

$$\frac{}{\Delta, B\,, \Delta' \mid \Gamma \vdash B}\ (vHyp)$$

Ahora se explica el significado de la implicación, el cual está dado por aquello que cuenta como una verificación de ésta. Decimos que $A \to B$ es verdadera, si suponiendo que A es verdadera B también lo es:

$$\frac{\Delta \mid \Gamma, A \vdash B}{\Delta \mid \Gamma \vdash A \to B}\ (\to I)$$

Por otro lado, si se sabe que $A \to B$ es verdadera, es decir que B es verdadera bajo la hipótesis A (por el significado de la regla de introducción de la implicación), y también se tiene una evidencia de la verdad de A, se puede utilizar ésta evidencia para descargar la hipótesis y obtener así evidencia para la verdad de B:

$$\frac{\Delta \mid \Gamma \vdash A \to B \quad \Delta \mid \Gamma \vdash A}{\Delta \mid \Gamma \vdash B}\ (\to E)$$

El siguiente paso es incorporar los juicios categóricos así como la definición de validez, de tal forma que se logre expresar la regla de necesitación de forma correcta. Se puede considerar que las hipótesis A_1 *verdadera* , ... , A_n *verdadera* , describen el conocimiento de un mundo dado. También se puede interpretar que el juicio A *válida* , expresa que A es verdadera en un mundo del que no sabemos nada. En otras palabras, A es *necesariamente verdadera*. Obsérvese que al verificar la verdad de A sin presuponer conocimiento alguno, se puede hablar de *verdad necesaria*.

Escribiremos $\Box A$ para representar la proposición que expresa que A es válida. La regla de introducción permite concluir la verdad de $\Box A$ a partir de la validez de A:

$$\frac{\Delta \mid \cdot \vdash A}{\Delta \mid \Gamma \vdash \Box A}\ (\Box I)$$

Por otro lado, la regla de eliminación correspondiente no es tan sencilla de construir; la regla propuesta sigue el esquema de las reglas usuales para la eliminación de la disyunción o la cuantificación existencial: el conocimiento de que $\Box A$ es verdadera nos permite asumir que A es válida, de aquí que la regla se formula como sigue:

$$\frac{\Delta \mid \Gamma \vdash \Box A \qquad \Delta, A \mid \Gamma \vdash C}{\Delta \mid \Gamma \vdash C}\ (\Box E)$$

El sistema de deducción natural (al que nombraremos $DLMI_N$) queda definido como sigue:

Proposición	$A ::= varP \mid A \to A \mid \Box A$
Variable proposicional	$varP ::= P_0 \mid P_1 \mid \ldots \mid P_n$
Hipótesis verdaderas	$\Gamma ::= \cdot \mid \Gamma, A$
Hipótesis válidas	$\Delta ::= \cdot \mid \Delta, A$

Se adoptará la convención usual de que la implicación se asocia a la derecha, de tal forma que

$$A \to B \to C \text{ se entiende como } A \to (B \to C).$$

Los conjuntos de hipótesis verdaderas y válidas se combinan en el juicio hipotético de la forma

$$\Delta \mid \Gamma \vdash A$$

sujeto a las siguientes reglas de inferencia:

$$\frac{}{\Delta \mid \Gamma, A, \Gamma' \vdash A}\ (tHyp)$$

$$\frac{\Delta \mid \Gamma, A \vdash B}{\Delta \mid \Gamma \vdash A \to B}\ (\to I) \qquad \frac{\Delta \mid \Gamma \vdash A \to B \quad \Delta \mid \Gamma \vdash A}{\Delta \mid \Gamma \vdash B}\ (\to E)$$

$$\frac{}{\Delta, B, \Delta' \mid \Gamma \vdash B}\ (vHyp)$$

$$\frac{\Delta \mid \cdot \vdash A}{\Delta \mid \Gamma \vdash \Box A}\ (\Box I) \qquad \frac{\Delta \mid \Gamma \vdash \Box A \qquad \Delta, A \mid \Gamma \vdash C}{\Delta \mid \Gamma \vdash C}\ (\Box E)$$

Observe el lector, que el operador de negación en este sistema de deducción, no existe, por lo que en realidad trabajamos con una lógica minimal, aún cuando se podría definir la negación intuicionista de la manera usual, es decir como $\neg A = A \to \bot$. Sin embargo, dado que las modalidades de *necesidad* y (más adelante) *posibilidad* se definen independientemente una de la otra, es decir no se consideran duales y el razonamiento requerido por el asistente de pruebas Coq es estrictamente constructivo, la negación resulta irrelevante.

Se introduce a continuación la noción de *derivación* de un juicio.

Definición 2. *Una* prueba o derivación Π *de un juicio* $J =_{def} \Delta \mid \Gamma \vdash A$ *es una secuencia finita de juicios* $\Pi = \langle J_1, \ldots, J_k \rangle$ *tal que* $J_k = J$ *y para cada* $1 \leq i \leq k$ *se cumple alguna de las siguientes condiciones:*

- J_i *es una instancia de la regla* (tHyp).
- J_i *es una instancia de la regla* (vHyp).
- J_i *es la conclusión de una instancia de alguna de las reglas de inferencia cuyas premisas son* $J_{l_1}, \ldots, J_{l_n}$ *con* $l_1, \ldots, l_n \leq i$

Decimos que un juicio J es derivable (demostrable) si existe una derivación de J.

Ejemplo 1. *El juicio* $\cdot \mid \cdot \vdash \Box A \to A$ *es derivable. (Axioma T)*

1.	$\cdot \mid \Box A \vdash \Box A$	$(tHyp)$
2.	$A \mid \Box A \vdash A$	$(vHyp)$
3.	$\cdot \mid \Box A \vdash A$	$(\Box\ E \quad 1,2)$
4.	$\cdot \mid \cdot \vdash \Box A \to A$	$(\to\ I \quad 3)$

Donde la secuencia de juicios enumerados del 1 al 4, son una prueba o derivación del juicio $\cdot \mid \cdot \vdash \Box A \to A$ *, por lo que se afirma que es derivable.*

Ejemplo 2. *El juicio* $\cdot \mid \cdot \vdash \Box A \to \Box\Box A$ *es derivable. (Axioma 4)*

1.	$A \mid \cdot \vdash A$	$(vHyp)$
2.	$A \mid \cdot \vdash \Box A$	$(\Box\ I \quad 1)$
3.	$A \mid \Box A \vdash \Box\Box A$	$(\Box\ I \quad 2)$
4.	$\cdot \mid \Box A \vdash \Box A$	$(tHyp)$
5.	$\cdot \mid \Box A \vdash \Box\Box A$	$(\Box\ E \quad 4,3)$
6.	$\cdot \mid \cdot \vdash \Box A \to \Box\Box A$	$(\to\ I \quad 5)$

Donde la secuencia de juicios enumerados del 1 al 6, son una prueba o derivación del juicio $\cdot \mid \cdot \vdash \Box A \to \Box\Box A$*, de aquí que el juicio es derivable.*

Ejemplo 3. *El juicio* $\cdot \mid \cdot \vdash \Box(A \to B) \to (\Box A \to \Box B)$ *es derivable. (Axioma K)*

1.	$A, A \to B \mid \cdot \vdash A$	$(vHyp)$
2.	$A, A \to B \mid \cdot \vdash A \to B$	$(vHyp)$
3.	$A, A \to B \mid \cdot \vdash B$	$(\to E \;\; 2,1)$
4.	$A, A \to B \mid \Box(A \to B), \Box A \vdash \Box B$	$(\Box I \;\; 3)$
5.	$A \mid \Box(A \to B), \Box A \vdash \Box(A \to B)$	$(tHyp)$
6.	$A \mid \Box(A \to B), \Box A \vdash \Box B$	$(\Box E \;\; 5,4)$
7.	$\cdot \mid \Box(A \to B), \Box A \vdash \Box A$	$(tHyp)$
8.	$\cdot \mid \Box(A \to B), \Box A \vdash \Box B$	$(\Box E \;\; 7,6)$
9.	$\cdot \mid \Box(A \to B) \vdash \Box A \to \Box B$	$(\to I \;\; 8)$
10.	$\cdot \mid \cdot \vdash \Box(A \to B) \to (\Box A \to \Box B)$	$(\to I \;\; 9)$

Donde la secuencia de juicios enumerados del 1 al 10, son una prueba o derivación del juicio $\cdot \mid \cdot \vdash \Box(A \to B) \to (\Box A \to \Box B)$*, por lo que es derivable.*

De los ejemplos 1, 2 y 3, se observa que los axiomas de la lógica modal conocidos como axioma T, 4 y K, que corresponden al sistema modal $S4$, se pueden demostrar dentro de este sistema de deducción natural.

Proposición 1. *El sistema de inferencia* $DLMI_N$ *satisface la ley de intercambio:*

$$Si \;\; \Delta \mid \Gamma, A, B, \Gamma' \vdash C \;\; entonces \;\; \Delta \mid \Gamma, B, A, \Gamma' \vdash C$$

Demostración. Por inducción sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$.

- **(*tHyp*)** Dado que el juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla *tHyp*, se tienen dos casos:

  1. $A = C$ o $B = C$
     De aquí que, por la regla de *tHyp* se tiene que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.
  2. Si no ocurre el punto anterior entonces $\exists \Gamma_1, \Gamma_2$ tal que $\Gamma = \Gamma_1, C, \Gamma_2$ o bien $\Gamma' = \Gamma_1, C, \Gamma_2$, al aplicar la regla *tHyp* se concluye $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **(*vHyp*)** Como el juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla *vHyp*, se tiene que $\exists \Delta_1, \Delta_2$ tal que $\Delta = \Delta_1, C, \Delta_2$.

  Empleando la regla *vHyp* concluimos que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **($\rightarrow I$)** Puesto que $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ es la conclusión de la instancia de la regla de introducción de la implicación, establecemos $C = C_1 \rightarrow C_2$. Se sabe entonces que $\Delta \mid \Gamma, A, B, \Gamma', C_1 \vdash C_2$, por hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, A, \Gamma', C_1 \vdash C_2$, que al aplicar la regla de $(\rightarrow I)$ se tiene $\Delta \mid \Gamma, B, A, \Gamma' \vdash C_1 \rightarrow C_2$, esto es $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **($\rightarrow E$)** Dado que el juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla $(\rightarrow E)$, se sabe que existe D tal que $\Delta \mid \Gamma, A, B, \Gamma' \vdash D \rightarrow C$ y $\Delta \mid \Gamma, A, B, \Gamma' \vdash D$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, A, \Gamma' \vdash D \rightarrow C$ y $\Delta \mid \Gamma, B, A, \Gamma' \vdash D$. Aplicando la regla de $(\rightarrow E)$, se concluye que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$

- **($\Box I$)** Al ser $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ la conclusión de la instancia de la regla $(\Box I)$, establecemos $C = \Box C_1$. Se sabe entonces que $\Delta \mid \cdot \vdash C_1$, aplicando la regla $(\Box I)$ se deduce $\Delta \mid \Gamma, B, A, \Gamma' \vdash \Box C_1$, es decir $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **($\Box E$)** Dado que el juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla $(\Box E)$, se sabe que existe D tal que $\Delta \mid \Gamma, A, B, \Gamma' \vdash \Box D$ y $\Delta, D \mid \Gamma, A, B, \Gamma' \vdash C$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, A, \Gamma' \vdash \Box D$ y $\Delta, D \mid \Gamma, B, A, \Gamma' \vdash C$. Aplicando la regla de $(\Box E)$, se concluye que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

□

Proposición 2. *El sistema de inferencia $DLMI_N$ satisface la ley de contracción:*

$$\textit{Si } \Delta \mid \Gamma, A, A, \Gamma' \vdash C \textit{ entonces } \Delta \mid \Gamma, A, \Gamma' \vdash C$$

Demostración. La demostración se construye por inducción (similar a la demostración de la Proposición 1) sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, A, A, \Gamma' \vdash C$. □

Proposición 3. *El sistema de inferencia $DLMI_N$ satisface la ley de debilitamiento :*

$$\textit{Si } \Delta \mid \Gamma, \Gamma' \vdash C \textit{ entonces } \Delta \mid \Gamma, B, \Gamma' \vdash C$$

Demostración. Por inducción sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, \Gamma' \vdash C$.

- **(*tHyp*)** Puesto que el juicio $\Delta \mid \Gamma, \Gamma' \vdash C$ se obtuvo como instancia de la regla *(tHyp)*, sucede que $\exists \Gamma_1, \Gamma_2$ tal que $\Gamma = \Gamma_1, C, \Gamma_2$ o bien $\Gamma' = \Gamma_1, C, \Gamma_2$.
  Cumpliéndose cualquiera de ambos casos, considerando el conjunto de hipótesis verdaderas Γ, B, Γ' y al aplicar la regla *(tHyp)* se concluye $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

- **(*vHyp*)** Como el juicio $\Delta \mid \Gamma, \Gamma' \vdash C$ se obtuvo como instancia de la regla *(vHyp)*, se tiene que $\exists \Delta_1, \Delta_2$ tal que $\Delta = \Delta_1, C, \Delta_2$, empleando la regla de (*vTyp*) concluimos que $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

- **($\rightarrow I$)** Puesto que $\Delta \mid \Gamma, \Gamma' \vdash C$ es la conclusión de la instancia de la regla de introducción de la implicación, establecemos $C = C_1 \rightarrow C_2$. Se sabe entonces que $\Delta \mid \Gamma, \Gamma', C_1 \vdash C_2$, por hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, \Gamma', C_1 \vdash C_2$, que al aplicar la regla de $(\rightarrow I)$ se tiene $\Delta \mid \Gamma, B, \Gamma' \vdash C_1 \rightarrow C_2$, esto es $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

- **($\rightarrow E$)** Dado que el juicio $\Delta \mid \Gamma, \Gamma' \vdash C$ se obtuvo como instancia de la regla $(\rightarrow E)$, se sabe que existe D tal que $\Delta \mid \Gamma, \Gamma' \vdash D \rightarrow C$ y $\Delta \mid \Gamma, \Gamma' \vdash D$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, \Gamma' \vdash D \rightarrow C$ y $\Delta \mid \Gamma, B, \Gamma' \vdash D$. Aplicando la regla de $(\rightarrow E)$, se concluye que $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

- **($\Box I$)** Al ser $\Delta \mid \Gamma, \Gamma' \vdash C$ la conclusión de la instancia de la regla ($\Box I$), establecemos $C = \Box C_1$. Se sabe entonces que $\Delta \mid \cdot \vdash C_1$, aplicando la regla ($\Box I$) se deduce $\Delta \mid \Gamma, B, \Gamma' \vdash \Box C_1$, es decir $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

- **($\Box E$)** Dado que el juicio $\Delta \mid \Gamma, \Gamma' \vdash C$ se obtuvo como instancia de la regla ($\Box E$), se sabe que existe D tal que $\Delta \mid \Gamma, \Gamma' \vdash \Box D$ y $\Delta, D \mid \Gamma, \Gamma' \vdash C$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, \Gamma' \vdash \Box D$ y $\Delta, D; \mid \Gamma, B, \Gamma' \vdash C$. Aplicando la regla de ($\Box E$), se concluye que $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

□

Proposición 4. *El sistema $DLMI_N$ también satisface las leyes estructurales análogas, sobre los contextos de fórmulas válidas:*

- Intercambio
  Si $\Delta, A, B, \Delta' \mid \Gamma \vdash C$ *entonces* $\Delta, B, A, \Delta' \mid \Gamma \vdash C$
- Contracción
  Si $\Delta, A, A, \Delta' \mid \Gamma \vdash C$ *entonces* $\Delta, A, \Delta' \mid \Gamma \vdash C$
- Debilitamiento:
  Si $\Delta, \Delta' \mid \Gamma \vdash C$ *entonces* $\Delta, B, \Delta' \mid \Gamma \vdash C$

Demostración. Estas propiedades se demuestran por inducción estructural, de manera análoga a las demostraciones de las proposiciones 1, 2 y 3.

□

Teorema 1. Sustitución: *El sistema de inferencia para la lógica modal $DLMI_N$ satisface el siguiente principio de sustitución:*

$$\textit{Si } \Delta \mid \Gamma \vdash A \textit{ y } \Delta \mid \Gamma, A, \Gamma' \vdash C \textit{ entonces } \Delta \mid \Gamma, \Gamma' \vdash C$$

Demostración. Por inducción sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, A, \Gamma' \vdash C$.

- **(*tHyp*)** Dado que el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash C$ se obtuvo como instancia de la regla *tHyp*, tenemos dos casos:
  1. $A = C$
     Por hipótesis sabemos que $\Delta \mid \Gamma \vdash A$, por lo que $\Delta \mid \Gamma; \Gamma' \vdash A$ dado que $A = C$ sucede $\Delta \mid \Gamma; \Gamma' \vdash C$.
  2. Si $A \neq C$ entonces sucede que $\exists \Gamma_1, \Gamma_2$ tal que $\Gamma; \Gamma' = \Gamma_1, C, \Gamma_2$.
     Por la regla de *(tHyp)* se sigue que $\Delta \mid \Gamma, \Gamma' \vdash C$.
- **(*vHyp*)** Como el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash C$ se obtuvo como instancia de la regla *(vHyp)*, se tiene que $\exists \Delta_1, \Delta_2$ tal que $\Delta = \Delta_1, C, \Delta_2$, empleando la regla de (*vHyp*) concluimos que $\Delta_1, C, \Delta_2 \mid \Gamma ; \Gamma' \vdash C$.
- **($\rightarrow I$)** Puesto que $\Delta \mid \Gamma, A, \Gamma' \vdash C$ es la conclusión de la instancia de la regla de introducción de la implicación, establecemos $C = C_1 \rightarrow C_2$. Se sabe entonces que
  $\Delta \mid \Gamma, A, \Gamma', C_1 \vdash C_2$, por hipótesis de inducción se tiene que $\Delta \mid \Gamma ; \Gamma', C_1 \vdash C_2$, secuente que al aplicarle la regla ($\rightarrow I$) se concluye $\Delta \mid \Gamma ; \Gamma' \vdash C_1 \rightarrow C_2$, esto es $\Delta \mid \Gamma ; \Gamma' \vdash C$.

- **($\rightarrow E$)** Dado que el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash C$ se obtuvo como instancia de la regla ($\rightarrow E$), sabemos que existe C_1 tal que $\Delta \mid \Gamma, A, \Gamma' \vdash C_1 \rightarrow C$ y $\Delta \mid \Gamma, A, \Gamma' \vdash C_1$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, \Gamma' \vdash C_1 \rightarrow C$ *verdadera* y $\Delta \mid \Gamma, \Gamma' \vdash C_1$. Aplicando la regla de ($\rightarrow E$), se concluye que $\Delta \mid \Gamma, \Gamma' \vdash C$.

- **($\Box I$)** Al ser $\Delta \mid \Gamma, A, \Gamma' \vdash C$ la conclusión de la instancia de la regla ($\Box I$), establecemos que $C = \Box C_1$. Se sabe entonces que $\Delta \mid \cdot \vdash C_1$, aplicando la regla ($\Box I$) se deduce $\Delta \mid \Gamma; \Gamma' \vdash \Box C_1$, es decir $\Delta \mid \Gamma; \Gamma' \vdash C$.

- **($\Box E$)** Como el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash C$ se obtuvo como instancia de la regla ($\Box E$), se sabe que existe una proposición C_1 tal que $\Delta \mid \Gamma, A, \Gamma' \vdash \Box C_1$ y $\Delta, C_1 \mid \Gamma, A, \Gamma' \vdash C$, por la hipótesis de inducción afirmamos que $\Delta \mid \Gamma; \Gamma' \vdash \Box C_1$ y $\Delta, C_1 \mid \Gamma; \Gamma' \vdash C$. Aplicando la regla de ($\Box E$), se concluye que $\Delta \mid \Gamma; \Gamma' \vdash C$.

□

La formulación del principio de sustitución para hipótesis válidas también se satisface en el sistema $DLMI_N$ y es el que se enuncia a continuación:

Teorema 2. Sustitución: *El sistema de inferencia para la lógica modal $DLMI_N$ satisface el siguiente principio de sustitución para hipótesis válidas:*

$$\textit{Si } \Delta, B, \Delta' \mid \Gamma \vdash C \textit{ y } \Delta \mid \cdot \vdash B \textit{ entonces } \Delta, \Delta' \mid \Gamma \vdash C$$

Es importante notar que el principio de sustitución no debe ser considerado como una regla de inferencia, sino como una propiedad acerca de los juicios que se emplean en el sistema de deducción.

2.3. Posibilidad

El concepto dual de *verdad necesaria* es el de *verdad posible*. Se dice que A es *posiblemente verdadera* si existe un mundo [1] en el cual A es verdadera. A diferencia de la lógica clásica, en nuestro caso no hay razón para esperar que la verdad posible se pueda definir en términos de la verdad necesaria; el análisis de este concepto desde el enfoque de juicios parece difícil si no se hace referencia a la existencia de mundos particulares, sin embargo, se logra empleando una combinación de juicios hipotéticos y categóricos.

La pregunta central es: ¿cómo emplear el conocimiento de que A es posiblemente verdadera?. Saber que A es posiblemente verdadera, significa que hay un mundo en el cual A es verdadera, pero sobre el cual no sabemos nada más. Por lo tanto, si al suponer que A es verdadera (y nada más) se concluye que C es posible, entonces debe suceder que C es posible. Utilizando la notación A *posible* que representa el juicio de que A es posible, se tiene:

$$\text{Si } A \textit{ posible} \text{ y } A \textit{ verdadera} \vdash C \textit{ posible} \text{ entonces } C \textit{ posible}$$

Es importante notar que sólo se pueden obtener conclusiones con respecto a la posibilidad de C, pero no su verdad; de hecho, la única manera que podemos establecer que A es posible es mostrar que A es verdadera:

$$\text{Si } A \textit{ verdadera} \text{ entonces } A \textit{ posible}.$$

[1] En términos de marcos de Kripke: A es *posiblemente verdadera* si existe un mundo accesible, en el cual A es verdadera.

Por supuesto, este razonamiento puede incluir y hacer uso de hipótesis, de tal forma que en la definición formal éstas se harán explícitas.

Definición de posibilidad:

1) Si $\Gamma \vdash A$ *verdadera* entonces $\Gamma \vdash A$ *posible*
2) Si $\Gamma \vdash A$ *posible* y A *verdadera* $\vdash C$ *posible* entonces $\Gamma \vdash C$ *posible*

es decir, si la verdad de A es derivable a partir de un conjunto de hipótesis Γ, entonces se puede derivar la posibilidad de A a partir de ese mismo conjunto Γ; además, si se tiene una derivación de la posibilidad de A a partir de un conjunto de hipótesis G y de suponer la verdad de A (pues del inciso anterior, A *posible* si A *verdadera*) se sigue C *posible* para alguna proposición C, entonces se concluye que C *posible* se deriva del conjunto G.

En el sistema de deducción que se construye, se desea considerar necesidad y posibilidad simultáneamente, pues ambos interactúan al preguntar por la verdad relativa a los mundos. Estableciendo la convención de que ambas, necesidad y posibilidad, deben referirse a los mismos mundos, entonces la definición de *posiblemente verdadero* se extiende permitiendo suposiciones sobre la validez, como sigue:

Definición de posibilidad con necesidad:

1) Si $\Delta \mid \Gamma \vdash A$ *verdadera* entonces $\Delta \mid \Gamma \vdash A$ *posible*
2) Si $\Delta \mid \Gamma \vdash A$ *posible* y $\Delta \mid A \vdash C$ *posible* entonces $\Delta \mid \Gamma \vdash C$ *posible*

Obsérvese que en 2), las hipótsis válidas Δ están disponibles para derivar el juicio C *posible* a partir de A *verdadera*. Esto porque la hipótesis en Δ son verdaderas en todos los mundos posibles y por lo tanto, en particular, en aquellos en los que se asumen que A es verdadera. Nótese que el segundo inciso tiene la forma del principio de sustitución y será utilizado como tal.

Al considerar validez en el sistema se introdujo una nueva forma de hipótesis, los juicios A *válida* pero no un nuevo juicio para derivarlos. Contrario a esto, para considerar posibilidad, no se requiere introducir una nueva forma de antecedente (hipótesis), en vez de ello se necesita una regla para derivar A *posible*.

El inciso 1) de la definición, nos permite concluir A *posible* a partir de A *verdadera*, lo cual se refleja en la regla:

$$\frac{\Delta \mid \Gamma \vdash A\,verdadera}{\Delta \mid \Gamma \vdash A\,posible}\ (nd\_tp)$$

El siguiente paso es representar el concepto de posibilidad con el operador $\Diamond$, generando las reglas de introducción y eliminación para este operador y siguiendo las ideas descritas anteriormente en la definición de posibilidad, se obtiene lo siguiente:

$$\frac{\Delta \mid \Gamma \vdash A\,posible}{\Delta \mid \Gamma \vdash \Diamond A\,verdadera}\ (\Diamond I) \qquad \frac{\Delta \mid \Gamma \vdash \Diamond A\,verdadera \qquad \Delta \mid A \vdash C\,posible}{\Delta \mid \Gamma \vdash C\,posible}\ (\Diamond E)$$

El principio de sustitución para validez, empleando el nuevo juicio $C\ posible$ como consecuente, justifica una nueva variante de la regla de eliminación para el operador $\Box$ (necesidad):

$$\frac{\Delta \mid \Gamma \vdash \Box A\, verdadera \qquad \Delta, A \mid \Gamma \vdash C\, posible}{\Delta \mid \Gamma \vdash C\, posible}\ (\Box E_p)$$

El sistema de Deducción Natural con necesidad y posibilidad (al que nombraremos $DLMI_{NP}$) queda definido como sigue:

$$\begin{array}{cc} \textit{Proposición} & A ::= varP \mid A \to A \mid \Box A \mid \Diamond A \\ \textit{Variable proposicional} & varP ::= P_0 \mid P_1 \mid ... \mid P_n \\ \textit{Hipótesis verdaderas} & \Gamma ::= \cdot \mid \Gamma, A \\ \textit{Hipótesis válidas} & \Delta ::= \cdot \mid \Delta, A \end{array}$$

Los juicios básicos $A\, verdadera$, $A\ válida$ y $A\ posible$ se combinan en dos esquemas de juicios hipotéticos:

$$\Delta \mid \Gamma \vdash A\, verdadera$$
$$\Delta \mid \Gamma \vdash A\, posible$$

sujetos a las siguientes reglas de inferencia:

$$\frac{}{\Delta \mid \Gamma, A, \Gamma' \vdash A\ verdadera}\ (tHyp) \qquad \frac{}{\Delta, B, \Delta' \mid \Gamma \vdash B\ verdadera}\ (vHyp)$$

$$\frac{\Delta \mid \Gamma, A \vdash B\ verdadera}{\Delta \mid \Gamma \vdash A \to B\ verdadera}\ (\to I) \qquad \frac{\Delta \mid \Gamma \vdash A \to B\ verdadera \quad \Delta \mid \Gamma \vdash A\ verdadera}{\Delta \mid \Gamma \vdash B\ verdadera}\ (\to E)$$

$$\frac{\Delta \mid \cdot \vdash A\ verdadera}{\Delta \mid \Gamma \vdash \Box A\ verdadera}\ (\Box I) \qquad \frac{\Delta \mid \Gamma \vdash \Box A\ verdadera \qquad \Delta, A \mid \Gamma \vdash C\ verdadera}{\Delta\ ;\ \Gamma \vdash C\ verdadera}\ (\Box E)$$

$$\frac{\Delta \mid \Gamma \vdash A\, posible}{\Delta \mid \Gamma \vdash \Diamond A\, verdadera}\ (\Diamond\ I) \qquad \frac{\Delta\ ;\ \Gamma \vdash \Box A\, verdadera \qquad \Delta, A\ ;\ \Gamma \vdash C\, posible}{\Delta \mid \Gamma \vdash C\, posible}\ (\Box E_p)$$

$$\frac{\Delta \mid \Gamma \vdash A\ verdadera}{\Delta \mid \Gamma \vdash A\ posible}\ (nd\_tp) \qquad \frac{\Delta \mid \Gamma \vdash \Diamond A\ verdadera \qquad \Delta \mid A \vdash C\ posible}{\Delta \mid \Gamma \vdash C\ posible}\ (\Diamond E)$$

Es importante mencionar que el sistema de deducción que se presenta en el artículo [24], en el cual se basa este trabajo, no incorpora la regla (*nd_tp*) como tal, en vez de ello, es algo que se da por hecho debido a que se trata justamente de la definición de posibilidad.

La inclusión de la regla (*nd_tp*) es motivada por nuestro objetivo principal: desarrollar una implementación del sistema de deducción; así, la forma en la que en la implementación se tiene noción de que si $\Delta \mid \Gamma \vdash A$ *verdadera* entonces $\Delta \mid \Gamma \vdash A$ *posible* , es a través de la inserción de esta regla.

A continuación se muestran algunos ejemplos de derivación de fórmulas que involucran el operador de posibilidad.

Ejemplo 4. *El juicio* $\cdot \mid \cdot \vdash A \to \Diamond A$ *verdadera es derivable. (Axioma* $\Diamond T$*)*

1. $\cdot \mid A \vdash A$ *verdadera* $(tHyp)$
2. $\cdot \mid A \vdash A$ *posible* (nd_tp)
3. $\cdot \mid A \vdash \Diamond A$ *verdadera* $(\Diamond\ I\ \ 2)$
4. $\cdot \mid \cdot \vdash A \to \Diamond A$ *verdadera* $(\to\ I\ \ 3)$

La secuencia de juicios enumerados del 1 al 4, son una prueba o derivación del juicio $\cdot\ ;\ \cdot \vdash A \to \Diamond A$ *verdadera .*

Ejemplo 5. *El juicio* $\cdot \mid \cdot \vdash \Diamond\Diamond A \to \Diamond A$ *verdadera es derivable. (Axioma* $\Diamond 4$*)*

1. $\cdot \mid A \vdash A$ *verdadera* $(tHyp)$
2. $\cdot \mid A \vdash A$ *posible* (nd_tp)
3. $\cdot \mid \Diamond A \vdash \Diamond A$ *verdadera* $(tHyp)$
4. $\cdot \mid \Diamond A \vdash A$ *posible* $(\Diamond\ E\ \ 3, 2)$
5. $\cdot \mid \Diamond\Diamond A \vdash \Diamond\Diamond A$ *verdadera* $(tHyp)$
6. $\cdot \mid \Diamond\Diamond A \vdash A$ *posible* $(\Diamond\ E\ \ 5, 4)$
7. $\cdot \mid \Diamond\Diamond A \vdash \Diamond A$ *verdadera* $(\Diamond\ \ I\ \ 7)$
8. $\cdot \mid \cdot \vdash \Diamond\Diamond A \to \Diamond A$ *verdadera* $(\to\ I\ \ 8)$

La secuencia de juicios enumerados del 1 al 8, son una prueba o derivación del juicio $\cdot \mid \cdot \vdash \Diamond\Diamond A \to \Diamond A$ *verdadera .*

Ejemplo 6. *El juicio* $\cdot \mid \cdot \vdash \Box(A \to B) \to (\Diamond A \to \Diamond B)$ *verdadera es derivable. (Axioma* $\Diamond K$*)*

1. $A \to B \mid A \vdash A \to B$ *verdadera* $(vHyp)$
2. $A \to B \mid A \vdash A$ *verdadera* $(tHyp)$

3. $A \to B \mid A \vdash B$ *verdadera* $(\to\ E\ 1,2)$

4. $A \to B \mid A \vdash B$ *posible* $(nd\_tp\ 3)$

5. $A \to B \mid \Box(A \to B), \Diamond A \vdash \Diamond A$ *verdadera* $(tHyp)$

6. $A \to B \mid \Box(A \to B), \Diamond A \vdash B$ *posible* $(\Diamond\ E\ 5)$

7. $A \to B \mid \Box(A \to B), \Diamond A \vdash \Diamond B$ *verdadera* $(\Diamond\ I\ 6)$

8. $\cdot \mid \Box(A \to B), \Diamond A \vdash \Box(A \to B)$ *verdadera* $(tHyp)$

9. $\cdot \mid \Box(A \to B), \Diamond A \vdash \Diamond B$ *verdadera* $(\Box\ E\ 7,8)$

10. $\cdot \mid \Box(A \to B) \vdash \Diamond A \to \Diamond B$ *verdadera* $(\to\ I\ 9)$

11. $\cdot \mid \cdot \vdash \Box(A \to B) \to (\Diamond A \to \Diamond B)$ *verdadera* $(\to\ I\ 10)$

La secuencia de juicios enumerados del 1 al 11, son una prueba o derivación del juicio $\cdot \mid \cdot \vdash \Box(A \to B) \to (\Diamond A \to \Diamond B)$ *verdadera por lo que es derivable.*

Las leyes estructurales que satisface este nuevo sistema $DLMI_{NP}$, se enlistan a continuación.

Proposición 5. *El sistema de inferencia* $DLMI_{NP}$ *satisface la ley de intercambio:*

$$Si\ \Delta \mid \Gamma, A, B, \Gamma' \vdash C\ \ entonces\ \Delta \mid \Gamma, B, A, \Gamma' \vdash C$$

Demostración. Por inducción sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$.

En la sección 2.2, **proposición 1**, se desarrolló la demostración correspondiente al caso en que el juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como derivación a través de las reglas : ***tHyp , vHyp , → I , → E , □ I y □ E*** ; por lo anterior, en esta demostración, únicamente se desarrollan los casos para las reglas de derivación del sistema $DLMI_{NP}$ que no están en $DLMI_{NP}$.

- **($\Diamond$ *I*)** Si sucede que el juicio $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla de introducción del $\Diamond$, es decir $\Delta \mid \Gamma, A, B, \Gamma' \vdash \Diamond F$ para alguna F tal que $C = \Diamond F$, sabemos que $\Delta \mid \Gamma, A, B, \Gamma' \vdash F$ *posible* , por hipótesis de inducción tenemos que $\Delta \mid \Gamma, B, A, \Gamma' \vdash F$ *posible* . Aplicando la regla de $(\Diamond\ I)$ concluimos que $\Delta \mid \Gamma, B, A, \Gamma' \vdash \Diamond F$, es decir $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **($\Diamond$ *E*)** Supóngase que $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla de eliminación del $\Diamond$, por lo que este juicio tiene la estructura $\Delta \mid \Gamma, A, B, \Gamma' \vdash F$ *posible* donde $C = F$ *posible* entonces se sabe que $\Delta \mid \Gamma, A, B, \Gamma' \vdash \Diamond E$ y $\Delta \mid E \vdash F$ *posible* . Por hipótesis de inducción se cumple que $\Delta \mid \Gamma, B, A, \Gamma' \vdash \Diamond E$ y $\Delta \mid E \vdash F$ *posible* , aplicando la regla $(\Diamond\ E)$ a estas hipótesis, se concluye que $\Delta \mid \Gamma, B, A, \Gamma' \vdash F$ *posible* de aquí que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **(*nd_tp*)** Dado que $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo a través de la regla (nd_tp), se tiene que $\Delta \mid \Gamma, A, B, \Gamma' \vdash F$ *posible* para alguna F tal que $C = F$ *posible* y además $\Delta \mid \Gamma, A, B, \Gamma' \vdash F$ *verdadera* . El juicio $\Delta \mid \Gamma, B, A, \Gamma' \vdash F$ *verdadera* es válido por hipótesis de inducción. Aplicando la regla (nd_tp) se concluye que $\Delta \mid \Gamma, B, A, \Gamma' \vdash F$ *posible* por lo que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

- **($\Box$ E_p)**Supóngase que $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ se obtuvo como instancia de la regla $\Box E_p$, por lo que este juicio tiene la estructura $\Delta \mid \Gamma, A, B, \Gamma' \vdash F$ *posible* donde $C = F$ *posible* entonces se sabe que $\Delta \mid \Gamma, A, B, \Gamma' \vdash \Box E$ y $\Delta, E \mid \Gamma, A, B, \Gamma' \vdash F$ *posible*. Por hipótesis de inducción se cumple que $\Delta \mid \Gamma, B, A, \Gamma' \vdash \Box E$ y $\Delta, E \mid \Gamma, B, A, \Gamma' \vdash F$ *posible*, aplicando la regla ($\Box E_p$) a estas hipótesis, se concluye que $\Delta \mid \Gamma, B, A, \Gamma' \vdash F$ *posible* de aquí que $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$.

□

Proposición 6. *El sistema de inferencia $DLMI_{NP}$ satisface la ley de contracción:*

$$Si\ \Delta \mid \Gamma, A, A, \Gamma' \vdash C\ \ entonces\ \Delta \mid \Gamma, A, \Gamma' \vdash C.$$

Demostración. La demostración se construye por inducción sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, A, A, \Gamma' \vdash C$, de forma similar a la demostración de la proposición anterior.

□

Proposición 7. *El sistema de inferencia $DLMI_{NP}$ satisface la ley de debilitamiento :*

$$Si\ \Delta \mid \Gamma, \Gamma' \vdash C\ \ entonces\ \Delta \mid \Gamma, B, \Gamma' \vdash C.$$

Demostración. Por inducción sobre la estructura de la derivación del juicio $\Delta \mid \Gamma, \Gamma' \vdash C$. Como se ha planteado en la demostración de la ley de intercambio para el sistema $DLMI_{NP}$, se desarrollarán los caso de las reglas de derivación del sistema $DLMI_{NP}$ que no son parte del sistema $DLMI_N$.

- **($\Diamond$ *I*)** Puesto que $\Delta \mid \Gamma, \Gamma' \vdash C$ es la conclusión de la instancia de la regla de introducción del $\Diamond$, establecemos $C = \Diamond C_1$. Se sabe entonces que $\Delta \mid \Gamma, \Gamma' \vdash C_1$ *posible*, por hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, \Gamma' \vdash C_1$ *posible*, que al aplicar la regla ($\Diamond$ I) se obtiene $\Delta \mid \Gamma, B, \Gamma' \vdash \Diamond C_1$, esto es $\Delta \mid \Gamma, B, \Gamma' \vdash C$.
- **($\Diamond$ *E*)** Dado que el juicio $\Delta \mid \Gamma, \Gamma' \vdash C$ se obtuvo como instancia de la regla ($\Diamond$ E), se sabe que existe D tal que $\Delta \mid \Gamma, \Gamma' \vdash \Diamond D$ y $\Delta \mid D \vdash C$ *posible*, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, \Gamma' \vdash \Diamond D$ y $\Delta \mid D \vdash C$ *posible*. Aplicando la regla de ($\Diamond E$), se concluye que $\Delta \mid \Gamma, B, \Gamma' \vdash C$.
- **(*nd_tp*)** Al ser $\Delta \mid \Gamma, \Gamma' \vdash C$ la conclusión de la instancia de la regla (*nd_tp*), establecemos $C = C_1$ *posible*. Se sabe entonces que $\Delta \mid \Gamma, \Gamma' \vdash C_1$ *verdadera*, por la hipótesis de inducción tenemos $\Delta \mid \Gamma, B, \Gamma' \vdash C_1$ *verdadera* aplicando la regla (*nd_tp*) se deduce $\Delta \mid \Gamma, B, \Gamma' \vdash C_1$ *posible*, es decir $\Delta \mid \Gamma, B, \Gamma' \vdash C$.
- **($\Box$*E*$_p$)** Dado que el juicio $\Delta \mid \Gamma, \Gamma' \vdash C$ se obtuvo como instancia de la regla ($\Box E_p$), se sabe que existe D tal que $\Delta \mid \Gamma, \Gamma' \vdash \Box D$ y $\Delta, D \mid \Gamma, \Gamma' \vdash C$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, B, \Gamma' \vdash \Box D$ y $\Delta, D \mid \Gamma, B, \Gamma' \vdash C$. Aplicando la regla de ($\Box E_p$), se concluye que $\Delta \mid \Gamma, B, \Gamma' \vdash C$.

□

Proposición 8. *El sistema $DLMI_{NP}$ también satisface las leyes estructurales análogas, sobre los contextos de fórmulas válidas:*

- Intercambio
  Si $\Delta,\ A\ ,\ B\ ,\ \Delta'\ \mid \Gamma \vdash C$ *entonces* $\Delta,\ B\ ,\ A\ ,\ \Delta' \mid \Gamma \vdash C$
- Contracción
  Si $\Delta,\ A\ ,\ A\ ,\ \Delta' \mid \Gamma \vdash C$ *entonces* $\Delta,\ A\ ,\ \Delta' \mid \Gamma \vdash C$
- Debilitamiento:
  Si $\Delta,\ \Delta' \mid \Gamma \vdash C$ *entonces* $\Delta,\ B\ ,\ \Delta' \mid \Gamma \vdash C$

Demostración. De forma análoga a las demostraciones de estas leyes estructurales para el sistema $DLMI_N$, la demostración se desarrolla por inducción estructural.

□

El teorema de sustitución se enuncia como sigue, de acuerdo con [24]:

Teorema 3. Sustitución: *El sistema de inferencia para la lógica modal con implicación, necesidad y posibilidad satisface:*

a) Si $\Delta \mid \Gamma,\ A\ ,\ \Gamma' \vdash C$ *verdadera y* $\Delta \mid \Gamma \vdash A$ *verdadera entonces* $\Delta \mid \Gamma\ ,\Gamma' \vdash C$ *verdadera.*

b) Si $\Delta \mid \Gamma,\ A\ ,\ \Gamma' \vdash C$ *posible y* $\Delta \mid \Gamma \vdash A$ *verdadera entonces* $\Delta \mid \Gamma\ ,\Gamma' \vdash C$ *posible.*

c) Si $\Delta, B, \Delta' \mid \Gamma \vdash C$ *verdadera y* $\Delta \mid \cdot \vdash B$ *verdadera entonces* $\Delta, \Delta' \mid \Gamma \vdash C$ *verdadera.*

d) Si $\Delta, B, \Delta' \mid \Gamma \vdash C$ *posible y* $\Delta \mid \cdot \vdash B$ *verdadera entonces* $\Delta, \Delta' \mid \Gamma \vdash C$ *posible.*

e) Si $\Delta \mid A \vdash C$ *posible y* $\Delta \mid \Gamma \vdash A$ *posible entonces* $\Delta \mid \Gamma \vdash C$ *posible.*

Debido a la incorporación de la regla (*nd_tp*) en el sistema $DLMI_{NP}$, en el proceso de demostración del teorema anterior, se observó que para concluir la demostración del inciso b) se necesitaba dar por hecho el inciso a) y viceversa. De la misma forma, se presenta interdependencia entre los incisos c) y d). Este conflicto también se presenta, desde luego, al realizar la implementación en Coq. Por lo anterior, en este trabajo utilizaremos la siguiente versión del teorema de sustitución:

Teorema 4. Sustitución en $DLMI_{NP}$: *El sistema de inferencia para la lógica modal $DLMI_{NP}$ satisface:*

ab) Si $\Delta \mid \Gamma \vdash A$ *verdadera y* $\Delta \mid \Gamma,\ A\ ,\ \Gamma' \vdash J$ *entonces* $\Delta \mid \Gamma\ ,\Gamma' \vdash J$

cd) Si $\Delta \mid \cdot \vdash B$ *verdadera y* $\Delta, B, \Delta' \mid \Gamma \vdash J$ *entonces* $\Delta, \Delta' \mid \Gamma \vdash J$

e) Si $\Delta \mid A \vdash C$ *posible y* $\Delta \mid \Gamma \vdash A$ *posible entonces* $\Delta \mid \Gamma \vdash C$ *posible*

Donde J es un juicio, es decir, $J = C$ verdadera o bien $J = C$ posible.

Para la demostración de este teorema haremos uso de las siguientes propiedades conocidas como *de debilitamiento*:

1. Si $\Delta \mid \Gamma \vdash J$ entonces $\Delta \mid \Gamma; \Gamma' \vdash J$ para cualquier contexto de fórmulas verdaderas Γ'.
2. Si $\Delta \mid \Gamma \vdash J$ entonces $\Delta; \Delta' \mid \Gamma \vdash J$ para cualquier contexto de fórmulas válidas Δ'.

Demostración del inciso ab). Por inducción sobre la estructura de la derivación del segundo secuente.

P.D. Si $\Delta \mid \Gamma \vdash A \; verdadera$ y $\Delta \mid \Gamma, A, \Gamma' \vdash J$ entonces $\Delta \mid \Gamma, \Gamma' \vdash J$

- **(*tHyp*)** Dado que el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash J$ se obtuvo como instancia de la regla *(tHyp)*, establecemos $J = C \; verdadera$ para alguna C y tenemos dos casos:
  1. $A = C$
     Por hipótesis sabemos que $\Delta \mid \Gamma \vdash A \; verdadera$, por la propiedad 1 se afirma que $\Delta \mid \Gamma; \Gamma' \vdash A \; verdadera$, dado que $A = C$ sucede $\Delta \mid \Gamma; \Gamma' \vdash C \; verdadera$, es decir $\Delta \mid \Gamma; \Gamma' \vdash J$.
  2. Si $A \neq C$ entonces sucede que $\exists \Gamma_1, \Gamma_2$ tal que $\Gamma; \Gamma' = \Gamma_1, C, \Gamma_2$.
     Por la regla de *(tHyp)* se sigue que $\Delta \mid \Gamma, \Gamma' \vdash C \; verdadera$, es decir $\Delta \mid \Gamma, \Gamma' \vdash J$.
- **(*vHyp*)** Como el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash J$ se obtuvo como instancia de la regla *(vHyp)*, se tiene que $J = C \; verdadera$ para alguna C proposición y $\exists \Delta_1, \Delta_2$ tal que $\Delta = \Delta_1, C, \Delta_2$, empleando la regla de (*vHyp*) concluimos que $\Delta_1, C, \Delta_2 \mid \Gamma ; \Gamma' \vdash C \; verdadera$, es decir $\Delta \mid \Gamma ; \Gamma' \vdash J$.
- **($\rightarrow$ *I*)** Puesto que $\Delta \mid \Gamma, A, \Gamma' \vdash J$ es la conclusión de la instancia de la regla de introducción de la implicación, establecemos $J = C_1 \rightarrow C_2 \; verdadera$. Se sabe entonces que $\Delta \mid \Gamma, A, \Gamma', C_1 \vdash C_2 \; verdadera$, por hipótesis de inducción se tiene que $\Delta \mid \Gamma ; \Gamma', C_1 \vdash C_2 \; verdadera$, que al aplicar la regla de ($\rightarrow I$) se tiene $\Delta \mid \Gamma ; \Gamma' \vdash C_1 \rightarrow C_2 \; verdadera$, esto es $\Delta \mid \Gamma ; \Gamma' \vdash J$.
- **($\rightarrow$ *E*)** Dado que el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash J$ se obtuvo como instancia de la regla ($\rightarrow E$), se tiene que $J = C \; verdadera$ para alguna proposición C y además sabemos que existe C_1 tal que $\Delta \mid \Gamma, A, \Gamma' \vdash C_1 \rightarrow C$ y $\Delta; \Gamma, A, \Gamma' \vdash C_1$, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma, \Gamma' \vdash C_1 \rightarrow C$ y $\Delta; \Gamma, \Gamma' \vdash C_1$. Aplicando la regla de ($\rightarrow E$), se concluye que $\Delta; \Gamma, \Gamma' \vdash C \; verdadera$, es decir $\Delta \mid \Gamma, \Gamma' \vdash J$.
- **($\Box$*I*)** Al ser $\Delta \mid \Gamma, A, \Gamma' \vdash J$ la conclusión de la instancia de la regla ($\Box I$), establecemos que $J = \Box C \; verdadera$. Se sabe entonces que $\Delta \mid \cdot \vdash C \; verdadera$, aplicando la regla ($\Box I$) se deduce $\Delta \mid \Gamma; \Gamma' \vdash \Box C \; verdadera$, es decir $\Delta \mid \Gamma; \Gamma' \vdash J$.
- **($\Box$*E*)** Como el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash J$ se obtuvo como instancia de la regla ($\Box E$), se sabe que existe una proposición C tal que $J = C \; verdadera$ y además existe C_1 tal que $\Delta \mid \Gamma, A, \Gamma' \vdash \Box C_1 \; verdadera$ y $\Delta, C_1 \mid \Gamma, A, \Gamma' \vdash C \; verdadera$, por la hipótesis de inducción afirmamos que $\Delta \mid \Gamma; \Gamma' \vdash \Box C_1 \; verdadera$ y $\Delta, C_1 \mid \Gamma; \Gamma' \vdash C \; verdadera$. Aplicando la regla de ($\Box E$), se concluye que $\Delta \mid \Gamma; \Gamma' \vdash C \; verdadera$ o lo que es lo mismo $\Delta \mid \Gamma; \Gamma' \vdash J$.
- **($\Diamond$ *I*)** Puesto que $\Delta \mid \Gamma, A, \Gamma' \vdash J$ es la conclusión de la instancia de la regla de introducción del $\Diamond$, establecemos $J = \Diamond C \; verdadera$. Se sabe entonces que $\Delta \mid \Gamma, A, \Gamma' \vdash C \; posible$, por hipótesis de inducción se tiene que $\Delta \mid \Gamma; \Gamma' \vdash C \; posible$, que al aplicar la regla de ($\Diamond I$) se tiene $\Delta \mid \Gamma; \Gamma' \vdash \Diamond C \; verdadera$, esto es $\Delta \mid \Gamma; \Gamma' \vdash J$.

- **($\Diamond$ E)** Dado que el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash J$ se obtuvo como instancia de la regla ($\Diamond$ E), se sabe que existe una proposición C tal que $J = C$ *posible* y demás existe C_1 tal que $\Delta \mid \Gamma, A, \Gamma' \vdash \Diamond C_1$ *verdadera* y $\Delta \mid C_1 \vdash C$ *posible*, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma; \Gamma' \vdash \Diamond C_1$ *verdadera* y $\Delta \mid C_1 \vdash C$ *posible*. Aplicando la regla de ($\Diamond$ E), se concluye que $\Delta \mid \Gamma; \Gamma' \vdash C$ *posible*, es decir $\Delta \mid \Gamma; \Gamma' \vdash J$.

- **($\Box E_p$)** Si el secuente $\Delta \mid \Gamma, A, \Gamma' \vdash J$ se obtuvo como instancia de la regla ($\Box$ E_p), se sabe que existe una proposición C tal que $J = C$ *posible* y demás existe C_1 tal que $\Delta \mid \Gamma, A, \Gamma' \vdash \Box C_1$ *verdadera* y $\Delta, C_1 \mid \Gamma, A, \Gamma' \vdash C$ *posible*, por la hipótesis de inducción se tiene que $\Delta \mid \Gamma; \Gamma' \vdash \Box C_1$ *verdadera* y $\Delta, C_1 \mid \Gamma; \Gamma' \vdash C$ *posible*. Aplicando la regla de ($\Box$ E_p), se concluye que $\Delta \mid \Gamma; \Gamma' \vdash C$ *posible*, es decir $\Delta \mid \Gamma; \Gamma' \vdash J$.

- **(*nd_tp*)** Al ser $\Delta \mid \Gamma, A, \Gamma' \vdash J$ la conclusión de la instancia de la regla (*nd_tp*), establecemos $J = C$ *posible* para alguna proposición C. Se sabe entonces que $\Delta \mid \Gamma, A, \Gamma' \vdash C$ *verdadera*, por la hipótesis de inducción tenemos $\Delta \mid \Gamma; \Gamma' \vdash C$ *verdadera* aplicando la regla (*nd_tp*) se deduce $\Delta \mid \Gamma; \Gamma' \vdash C$ *posible*, es decir $\Delta \mid \Gamma; \Gamma' \vdash J$.

Demostración del inciso cd). Análoga a la prueba anterior.

Demostración del inciso e) Por inducción sobre la estructura de la derivación del segundo secuente.

P.D. Si $\Delta \mid A \vdash C$ *posible* y $\Delta \mid \Gamma \vdash A$ *posible* entonces $\Delta \mid \Gamma \vdash C$ *posible*

Obsérvese que el secuente $\Delta \mid \Gamma \vdash A$ *posible* sólo se puede obtener a través de las reglas de derivación ($\Diamond$ E), ($\Box$ E_p) y (*nd_tp*). Por lo que la prueba se desarrolla como sigue:

- **($\Diamond$ E)** En el caso de que el secuente $\Delta \mid \Gamma \vdash A$ *posible* se obtuvo como instancia de la regla ($\Diamond$ E), se sabe que existe una proposición B tal que $\Delta \mid \Gamma \vdash \Diamond B$ *verdadera* y $\Delta \mid B \vdash A$ *posible*, por la hipótesis de inducción se tiene que $\Delta \mid B \vdash C$ *posible*. Aplicando la regla de ($\Diamond$ E) con $\Delta \mid \Gamma \vdash \Diamond B$ *verdadera* y $\Delta \mid B \vdash C$ *posible*, se concluye que $\Delta \mid \Gamma \vdash C$ *posible*.

- **($\Box E_p$)** Si el secuente $\Delta \mid \Gamma \vdash A$ *posible* se obtuvo como instancia de la regla ($\Box$ E_p), se sabe que existe una proposición B tal que $\Delta \mid \Gamma \vdash \Box B$ *verdadera* y $\Delta, B \mid \Gamma \vdash A$ *posible*. La hipótesis de inducción para este último secuente se formula como:
  H.I Si $\Delta, B \mid A \vdash C$ *posible* y $\Delta, B \mid \Gamma \vdash A$ *posible* entonces $\Delta, B \mid \Gamma \vdash C$ *posible*.
  Para hacer uso de la hipótesis de inducción, se requiere mostrar que $\Delta, B \mid A \vdash C$ *posible*, secuente que se obtiene al aplicar la propiedad 2 a $\Delta \mid A \vdash C$ *posible*. Aplicando la H.I a $\Delta, B \mid A \vdash C$ *posible* y $\Delta, B \mid \Gamma \vdash A$ *posible* se tiene que $\Delta, B \mid \Gamma \vdash C$ *posible* . Aplicando la regla de ($\Box$ E_p) a los secuentes $\Delta \mid \Gamma \vdash \Box B$ *verdadera* y $\Delta, B \mid \Gamma \vdash C$ *posible* , se concluye que $\Delta \mid \Gamma \vdash C$ *posible*.

- **(*nd_tp*)** Al ser $\Delta \mid \Gamma \vdash A$ *posible* la conclusión de la instancia de la regla (*nd_tp*), se sabe que $\Delta \mid \Gamma \vdash A$ *verdadera*. De $\Delta \mid A \vdash C$ *posible* y por la propiedad 1 tenemos $\Delta \mid \Gamma, A \vdash C$ *posible* aplicando el teorema de sustitución *ab*) se deduce $\Delta \mid \Gamma \vdash C$ *posible*.

□

En este capítulo se ha expuesto y analizado el sistema de Deducción Natural $DLMI_{NP}$ para la lógica modal intuicionista. Se han demostrado propiedades estructurales sobre este sistema, así como un teorema de sustitución tanto para los contextos de fórmulas válidas como verdaderas. En el siguiente capítulo se desarrolla la formalización de este sistema en el asistente de pruebas Coq. Para ello, se iniciará con una breve introducción a este asistente de pruebas para después, mostrar las características de la codificación.

Capítulo 3

Formalización

En este capítulo se presenta el desarrollo de la formalización del sistema de deducción $DLMI_{NP}$, en el asistente de pruebas Coq. En la primera sección se presentan los aspectos generales de los asistentes de prueba para después, en la segunda sección hacer una breve introducción al asistente de pruebas, Coq y sus características. Se continúa con el análisis de la formalización del sistema de deducción, presentando fragmentos de código de la implementación (tanto de la representación de la lógica como de las demostraciones de sus propiedades) con los cuales se va explicando simultáneamente la forma en que trabaja el asistente de pruebas. El código en coq de la formalización del sistema de deducción así como la demostración de sus propiedades, se encuentran en el script *Archivo1.v* adjunto a la publicación de esta tesis que se puede consultar en `http://dgb.unam.mx/`.

3.1. Asistentes de Prueba

Un *asistente de prueba* es un sistema computacional que permite al usuario generar pruebas de manera interactiva, de tal forma que el usuario es quien tiene el control del proceso de derivación, contrario a los *demostradores automáticos de teoremas* (de los que se hablará más adelante), en los que no existe interacción con el usuario. Con respecto a las matemáticas, los asistentes de prueba juegan un papel primordial en el aspecto de definición y demostración, más que en el cómputo en sí, de tal forma que el usuario puede establecer una teoría matemática, definir propiedades y realizar razonamientos lógicos con ellos. Aunque algunos asistentes también permiten definir funciones, el objetivo principal es construir demostraciones de forma interactiva.

Si se está completamente convencido de que se cumple cierta propiedad o un teorema, probablemente su formalización y verificación con la ayuda de un asistente de prueba no resulte de mayor interés. Sin embargo hay situaciones en las que se pone en tela de juicio cierta prueba, ya sea porque es grande y compleja, como la prueba del teorema de los cuatro colores de Gonthier (donde lo adecuado es emplear recursos computacionales para verificarla), o en el ámbito de las ciencias de la computación, porque la prueba involucra razonamiento sobre software complejo, por ejemplo la prueba de una propiedad de un compilador, la cual involucra la sintaxis de un lenguaje de programación con un gran número de instrucciones; éstas, entre otras situaciones, dan pie a que una prueba en papel no sea aceptada del todo.

Es importante no confundir un asistente de pruebas con un *demostrador automático de teoremas*. Éste último es un sistema que consiste de en conjunto de procedimientos de decisión, que permiten demostrar automáticamente fórmulas en un formato restringido; un ejemplo de demostradores automáticos es Vampire [31], ganador de la *copa mundial de demostradores de teoremas* en 11 ocasiones. Si bien es cierto que los demostradores automáticos son poderosos, cuentan con una expresividad

limitada, lo cual no permite establecer correctamente una teoría matemática sobre ellos.

El lenguaje que se emplea en las demostraciones en un asistente de pruebas se basa en una lógica de orden superior. El asistente hace uso de *tácticas* seleccionadas por el usuario, para generar una prueba formal que corresponda a una prueba matemática estándar.

Los asistentes de prueba auxilian al usuario principalmente en los siguientes aspectos:

- Formalización de teorías o especificaciones: definiciones, axiomas, etc.
- Verificación y generación de pruebas en un lenguaje lógico como es de la deducción natural, mediante la automatización de heurísticas (tácticas) de prueba.
- Administración de los resultados, proporcionando herramientas como editores, bibliotecas, documentación, entre otros; y extracción de programas.

Algunos ejemplos de asistentes de pruebas son: ACL2 (A Computational Logic for Applicative Common Lisp) desarrollado por Matt Kaufmann y Strother Moore, Coq por INRIA, la familia de asistentes de prueba HOL (Higher Order Logic), Lego diseñado y desarrollado por Randy Pollack, que implementa sistemas como el cálculo de construcciones, el cálculo de construcciones generalizado y la teoría unificada de tipos dependientes; NuPRL por Joseph Bates y Robert Constable, Isabelle que es un sucesor de HOL y fue desarrolado por Larry Paulson (Cambridge), Tobias Nipkow (München) y Makarius Wenzel (Paris-Sud). Por último mencionamos Twelf el cual es una implementación del Edinburgh Logical Framework, desarrollado por Frank Pfenning y Carsten Schürmann. [28]

3.2. Coq

Coq es un asistente de pruebas basado en la correspondencia de Curry-Howard. Relaciona términos de un cálculo lambda tipado con árboles de prueba de un sistema lógico en forma de deducción natural. Coq implementa una lógica de orden superior: el cálculo de las construcciones inductivas[27], una extensión del cálculo de las construcciones [10].
Desde un enfoque práctico, Coq puede ser visto como un lenguaje de programación funcional —parecido a Haskell u OCaml—pero que cuenta con un sistema de tipos que permite expresar propiedades lógicas; de hecho, la forma usual de trabajar con este asistente de prueba es la siguiente: se desarrolla un programa en Coq (o bien la implementación de una teoría matemática) y las propiedades sobre el programa (o la teoría) se demuestran en Coq.

Coq se usa principalmente para:

- Desarrollar en general pruebas formalmente correctas de manera interactiva.
- Desarrollar pruebas matemáticas: el caso de la formalización del teorema de los cuatro colores de Gonthier [15], es una prueba de este teorema que consiste en la reducción del problema a 633 casos posibles. Gonthier formalizó todo esto en Coq, la reducción a 633 casos, la definición de los algoritmos que los verifican y las pruebas de corrección correspondientes.
- Desarrollar software certificado, es decir, extraer un programa certificado[1] a partir de la prueba constructiva de su especificación formal. Ejemplo de esto es el compilador Compcert [30]; un compilador para el lenguaje C, implementado y certificado utilizando Coq.

[1]El concepto de certificación empleado en lenguajes de programación y métodos formales, se refiere a la emisión de un certificado: una prueba matemática formal que garantiza que un programa cumple su especificación.

Coq implementa un lenguaje matemático de alto nivel para la especificación de programas llamado Gallina, el cual está basado en el Cálculo de Construcciones Inductivas. De esta forma Gallina combina una lógica de orden superior con un lenguaje de programación funcional ricamente tipado.

A través de un lenguaje de comandos llamado *Vernacular*, Coq permite:

- Definir funciones o predicados para ser evaluados de manera eficiente.
- Establecer teoremas matemáticos y especificaciones de software.
- Desarrollar de forma interactiva pruebas formales de los teoremas.
- Extraer programas certificados en lenguajes como Objetive Caml, Haskell o Scheme.

Al ser Coq un sistema de desarrollo de pruebas, ofrece métodos de prueba interactivos, algoritmos de decisión, semi-decisión y un lenguaje de *tácticas* que permiten al usuario generar sus propias pruebas.

A continuación se explica el concepto de *táctica* en Coq. Una regla de derivación es un vínculo entre alguna (y sólo una) fórmula, que llamamos la conclusión y algunas fórmulas que llamamos premisas. Una regla de derivación se puede interpretar de dos maneras. La primera de ellas es: "Si se asumen *Premisa1, Premisa2, ... , PremisaN*, entonces se deduce *Conclusión*". Por ejemplo, si se tiene una prueba de A y una prueba de B, entonces por la regla de derivación $(\wedge\, I)$ se tiene una prueba de $A \wedge B$. Esto se conoce como razonamiento hacia adelante[2] partiendo de las premisas y llegando a la conclusión. La segunda forma de interpretación es : "para probar *Conclusión*, antes hay que probar *Premisa1, Premisa2, ... , PremisaN* ". Por ejemplo, para probar $A \wedge B$, hay que demostrar A y demostrar B, según la regla de $(\wedge\, I)$. Este es el razonamiento hacia atrás: para mostrar la conclusión hay que demostrar las premisas. En este sentido diremos que la conclusión es el objetivo a probar y las premisas son los sub-objetivos.

A diferencia de las reglas de derivación de un sistema deductivo, las tácticas en Coq implementan el razonamiento hacia atrás. Para los efectos de este documento, establecemos que una meta consta de dos partes, un *contexto* –que se refiere a las hipótesis–y un *objetivo*. Cuando las tácticas son aplicadas a una meta, reemplazan ésta con las sub-metas que genera. Decimos entonces que una táctica reduce una meta a sus sub-metas. Esto se ve reflejado en el concepto de *prueba dirigida por metas*, que es la metodología que emplea Coq y diversos asistentes de pruebas para construir las demostraciones. El mecanismo de operación en este metodología es el siguiente:

1. El usuario introduce un enunciado (que representa la fórmula) que desea probar.
2. El sistema despliega el enunciado como una fórmula que tiene que se busca demostrar. También se da un contexto de hechos locales (hipótesis) que pueden ser utilizados para la construcción de la prueba.
3. El usuario indica un comando , correspondiente a una táctica, para descomponer la meta en otras más simples.
4. El sistema reemplaza la fórmula original por las fórmulas que corresponden a los sub-objetivos que ahora deben ser probados.
5. Cuando no hay más metas por demostrar, la prueba ha sido finalizada.

[2]Forward reasoning.

En Coq, cada objetivo (conclusión de la meta) tiene asociado un número. A la conclusión de la meta actual le corresponde el número 1. De forma predeterminada, cuando el usuario introduce una táctica, ésta se aplica a la meta actual, pero se puede abordar un objetivo concreto de una la lista de metas, escribiendo `n:tac` que significa, aplicar la táctica `tac` a la meta número `n`.

Es importante señalar que en el razonamiento hacia atrás no se puede aplicar cualquier regla de inferencia a una conclusión dada. Se tiene que verificar que la conclusión dada tenga la misma estructura que la conclusión de la regla de inferencia. Es por ello que no todas las tácticas se pueden utilizar para reducir cualquier meta. En otras palabras, antes de aplicar una táctica a un objetivo dado, el sistema verificará que algunas condiciones previas se cumplan. Si no es el caso, el sistema envía un mensaje de error, de tal forma que no se puede continuar con la construcción de la prueba hasta que se utilice la táctica adecuada.

3.3. Formalización del sistema $DLMI_{NP}$

La implementación en Coq de una especificación que da solución a un problema consiste en la descripción y desarrollo de ciertos componentes esenciales: los datos, las operaciones y las propiedades así como las pruebas de estas propiedades.
Con el fin de discutir la formalización del sistema de deducción natural que se expuso en el capítulo 2 y realizar una breve y rápida introducción a las principales características de Coq, se presentarán ordenadamente los cuatro componentes que se acaban de mencionar, a través de ejemplos extraídos de la formalización [3] que se ha construido. Es importante aclarar que este capítulo no pretende ser un manual de referencia o un texto de documentación sobre Coq. El objetivo de las siguientes secciones es proporcionar una sencilla y rápida introducción al sistema, a través de la descripción de los elementos empleados en la formalización del sistema $DLMI_{NP}$. Si el lector desea ahondar en ciertos temas relacionados con el uso y funcionamiento de Coq, se recomienda consultar [6].

3.3.1. Datos

Antes de entrar de lleno en la formalización del sistema, tomaremos un momento para hablar de la notación que emplea Coq.

La notación $A : B$ se utiliza para dos propósitos: el primero es indicar que el tipo de la expresión A es la expresión B y el segundo propósito es expresar que A es una prueba de B. Ambas interpretaciones son equivalentes de acuerdo con la correspondencia de Curry-Howard. En el cálculo de las construcciones inductivas, todos los elementos deben tener un tipo asociado. En particular, el tipo de un tipo es llamado *clase* [4]. Todos los elementos y expresiones en Coq están clasificados en tres categorías: las clases `Prop`, `Type` y `Set`. La primera es la clase para proposiciones, es decir, las proposiciones bien formadas son del tipo `Prop`; la segunda clase contempla los tipos de datos y estructuras matemáticas, es decir, los tipos y estructuras matemáticas bien formados son del tipo `Type`. Por último un miembro de la clase `Set` es una especificación.

Como en cualquier lenguaje de programación, se requiere definir nuevos tipos o estructuras de datos para expresar por ejemplo, que un elemento (dato) encaja en cierta colección de casos, donde cada caso contiene un número de campos. Los tipos o estructuras de datos se definen inductivamente. En el sistema de deducción $DLMI_{NP}$, tenemos las siguientes definiciones:

[3] El código en coq de esta formalización se encuentra en el archivo adjunto DLMInp.v
[4] En inglés *sort*.

$$\begin{array}{ll} \textit{Proposición} & A ::= varP \mid A \to A \mid \Box A \mid \Diamond A \\ \textit{Variable proposicional} & varP ::= P_0 \mid P_1 \mid ... \mid P_n \end{array}$$

Esta información corresponde a las siguientes líneas de código de la fomalización en Coq.

```
Inductive Proposition : Type :=
  | Varp : N -> Proposition
  | Impl : Proposition -> Proposition -> Proposition
  | Box  : Proposition -> Proposition
  | Dia  : Proposition -> Proposition.
```

Esto define un nuevo tipo `Proposition` y expresa que los elementos de este tipo se obtienen de cuatro formas diferentes. En el primer caso hay un campo que espera recibir un Natural (`N` es un elemento primitivo en Coq), de tal forma que si n $\in$ N[5], el nuevo elemento se denota como `Varp n` y es de tipo `Proposition`. Para el segundo caso, hay dos campos que esperan recibir sendos datos de tipo `Proposition`, digamos `t1, t2`, entonces el nuevo elemento se representa como `Impl t1 t2`. Los dos elementos restantes construyen elementos de la forma `Box t` y `Dia t`, donde `t` es un dato de tipo `Proposition`. En busca de una representación de fácil manejo y comprensible, se empleará la siguiente notación:

```
Notation "x ==> y" := (Impl x y) (at level 55, right associativity).
Notation "# x" := (Box x)       (at level 54, right associativity).
Notation "<> x" := (Dia x)     (at level 53, right associativity).
```

En las líneas anteriores, los comandos de la derecha que se encuentran dentro de paréntesis, indican la precedencia y asociatividad de los operadores que se definen. Para manejar la ambigüedad en las expresiones, Coq hace uso de niveles de precedencia en un rango de 0 a 100 y reglas de asociatividad: a la derecha, a la izquierda o sin asociatividad. En la sección *Coq.Init.Notations* de [29] se en listan notaciones cuyo nivel y asociatividad están fijos, esta lista sirve como guía para establecer los valores de estos atributos en nuestras definiciones.

El tipo de dato que representa un juicio, se construye de la siguiente forma:

```
Inductive Judgm :=
 | JTrue : Proposition -> Judgm
 | JPoss : Proposition -> Judgm.
```

Esta definición indica que hay dos tipos de juicios. En el primer caso se generan los que afirman la verdad de una proposición y el segundo caso genera los juicios que afirman la posibilidad de una proposición.

El comando `Check` proporciona un mecanismo para verificar cuándo una expresión está bien formada, indicando el tipo de la expresión que se quiere verificar. Por ejemplo:

```
Check Varp 20.
 Varp 20 : Proposition.
```

[5] N representa los números naturales.

```
Check Impl (Box (Varp 3)) (Dia (Dia (Varp 6))).
 #Varp 3 ==> <> <> Varp 6  : Proposition

Check JTrue (Varp 5).
 JTrue (Varp 5)  : Judgm

Check JPoss (Dia  5).
 Error: The term "5" has type "Z" while it is expected to have type  "Proposition".
```

En el bloque de definición de datos, se incluye también una estructura que representa contextos (una lista de proposiciones). Hay dos formas de generar un contexto: con el constructor `empty`, que representa un contexto vacío; o bien con el constructor `snoc` que toma un contexto y una proposición, produciendo un nuevo elemento de tipo `ctx`, al agregar la proposición al final del contexto que recibió. Como ya se habrá notado y se reflejará en las siguientes secciones, los constructores y las funciones en Coq están generalmente en forma currificada. [6]

```
Inductive ctx : Type :=
 | empty : ctx
 | snoc : ctx -> Proposition -> ctx.
```

Se empleará la siguiente notación sobre contextos.

```
Notation "G,p" := (snoc G p) (at level 20, t at next level).
```

Como se puede observar en la definición formal, los contextos de hipótesis verdaderas y de hipótesis válidas se construyen exactamente de la misma forma, por lo que en la implementación, ambos resultan un alias del tipo `ctx`.

```
Definition TrueHyps : Set := ctx.
Definition  ValHyps : Set := ctx.
```

Por último, se tiene una definición plasmada en la formalización del sistema que refleja la construcción de un secuente $\Delta \mid \Gamma \vdash J$ de manera inductiva, a través de las reglas de derivación. Se analiza a continuación la formalización de una de las reglas en particular, de tal forma que sea más comprensible el código posterior.

Considérese la regla de la eliminación de la implicación:

$$\frac{\Delta \mid \Gamma \vdash A \to B \; verdadera \quad \Delta \mid \Gamma \vdash A \; verdadera}{\Delta \mid \Gamma \vdash B \; verdadera} \; (\to E)$$

Si bien no se expresa explícitamente en la regla, se sobreentiende que ésta se aplica para cualesquiera Δ y Γ, listas de hipótesis válidas y verdaderas, respectivamente, así como para cualesquiera proposiciones A y B. La regla de eliminación de la implicación se puede representar como sigue:

[6] La *currificación* es una correspondencia entre las funciones que reciben una tupla con múltiples argumentos y las funciones que reciben estos mismos argumentos pero de forma yuxtapuesta. La versión currificada de una función que recibe una tupla de argumentos, es una función que recibe un único argumento y devuelve una función intermedia que completa las operaciones.

Sean Δ, Γ contextos de hipótesis válidas y verdaderas, respectivamente, y sean A, B proposiciones,

$$\forall \Delta, \Gamma, A, B \left(\text{Si } \Delta \mid \Gamma \vdash A \to B \; verdadera \text{ y } \Delta \mid \Gamma \vdash A \; verdadera \text{ entonces } \Delta \mid \Gamma \vdash B \; verdadera \right)$$

El secuente $\Delta \mid \Gamma \vdash J$ se formaliza mediante el término `ND_Proof D G J`. Como ya se ha mencionado, los constructores y las funciones en Coq, están currificadas, por lo que la formalización del enunciado anterior, que es un caso dentro de la definición inductiva de derivación, se codifica como sigue:

```
|  nd_apply : forall (D : ValHyps) (G : TrueHyps) (A B : Proposition),
             ND_Proof D G (JTrue (A ==> B)) -> ND_Proof D G (JTrue A)
             ->ND_Proof D G (JTrue B)
```

Para continuar, recuérdese que las reglas de derivación del sistema $DLMI_{NP}$ son las siguientes:

$$\frac{}{\Delta \mid \Gamma, A, \Gamma' \vdash A \; verdadera} \; (tHyp) \qquad \frac{}{\Delta, B, \Delta' \mid \Gamma \vdash B \; verdadera} \; (vHyp)$$

$$\frac{\Delta \mid \Gamma, A \vdash B \; verdadera}{\Delta \mid \Gamma \vdash A \to B \; verdadera} \; (\to I) \qquad \frac{\Delta \mid \Gamma \vdash A \to B \; verdadera \quad \Delta \mid \Gamma \vdash A \; verdadera}{\Delta \mid \Gamma \vdash B \; verdadera} \; (\to E)$$

$$\frac{\Delta \mid \cdot \vdash A \; verdadera}{\Delta \mid \Gamma \vdash \Box A \; verdadera} \; (\Box I) \qquad \frac{\Delta \mid \Gamma \vdash \Box A \; verdadera \qquad \Delta, A \mid \Gamma \vdash C \; verdadera}{\Delta \mid \Gamma \vdash C \; verdadera} \; (\Box E)$$

$$\frac{\Delta \mid \Gamma \vdash A \, posible}{\Delta \mid \Gamma \vdash \Diamond A \, verdadera} \; (\Diamond I) \qquad \frac{\Delta \mid \Gamma \vdash \Box A \, verdadera \qquad \Delta, A \mid \Gamma \vdash C \, posible}{\Delta \mid \Gamma \vdash C \, posible} \; (\Box E_p)$$

$$\frac{\Delta \mid \Gamma \vdash A \; verdadera}{\Delta \mid \Gamma \vdash A \; posible} \; (nd\_tp) \qquad \frac{\Delta \mid \Gamma \vdash \Diamond A \; verdadera \qquad \Delta \mid A \vdash C \; posible}{\Delta \mid \Gamma \vdash C \; posible} \; (\Diamond E)$$

La construcción de una derivación de un secuente, sujeta a estas reglas descritas, queda representado en Coq de la siguiente forma:

```
Inductive ND_Proof : ValHyps -> TrueHyps -> Judgm -> Prop :=

| nd_thyp : forall (D : ValHyps) (G G' : TrueHyps) (A : Proposition),
             ND_Proof D ((G,A) ; G') (JTrue A)

| nd_vhyp : forall (D D': ValHyps) (G : TrueHyps) (B : Proposition),
             ND_Proof (D,B ; D') G (JTrue B)

| nd_intro : forall (D : ValHyps) (G : TrueHyps) (A B : Proposition),
             ND_Proof D (G,A) (JTrue B) -> ND_Proof D G (JTrue (A ==> B))

| nd_apply : forall (D : ValHyps) (G : TrueHyps) (A B : Proposition),
             ND_Proof D G (JTrue (A ==> B)) -> ND_Proof D G (JTrue A)
             ->ND_Proof D G (JTrue B)

| nd_boxI : forall (D : ValHyps) (G : TrueHyps) (A: Proposition),
             ND_Proof D empty (JTrue A)-> ND_Proof D G (JTrue (Box A))

| nd_boxE : forall (D : ValHyps) (G : TrueHyps) (A C : Proposition),
             ND_Proof D G (JTrue (Box A)) -> ND_Proof (D,A) G (JTrue C)
             -> ND_Proof D G (JTrue C)

| nd_boxEp : forall (D : ValHyps) (G : TrueHyps) (A C : Proposition),
             ND_Proof D G (JTrue (Box A)) -> ND_Proof (D,A) G (JPoss C)
             -> ND_Proof D G (JPoss C)

| nd_diaI : forall (D : ValHyps) (G : TrueHyps) (A: Proposition),
             ND_Proof D G (JPoss A) -> ND_Proof D G (JTrue (Dia A))

| nd_diaE : forall (D : ValHyps) (G : TrueHyps) (A C: Proposition),
             ND_Proof D G (JTrue (Dia A)) -> ND_Proof D (empty,A) (JPoss C)
             -> ND_Proof D G (JPoss C)

| nd_tp: forall (D : ValHyps) (G : TrueHyps) (A: Proposition),
             ND_Proof D G (JTrue A) ->  ND_Proof D G (JPoss A).
```

A lo largo de este trabajo, en algunas ocasiones se ha empleado el secuente $D \mid G \vdash J$, en donde J puede ser un juicio de verdad o de posibilidad, indistintamente. En otras circunstancias, se requiere hacer explícito cuando el secuente deriva un juicio de verdad o posibilidad y entonces escribimos $D \mid G \vdash A\ \textit{verdadera}$ o bien $D \mid G \vdash A\ \textit{posible}$. Estas ideas se expresan en la siguiente notación:

```
Notation "D | G |--j J" := (ND_Proof D G J) (at level 30).

Notation "D | G |-- A" := (ND_Proof D G (JTrue A)) (at level 30).

Notation "D | G |--p A" := (ND_Proof D G (JPoss A)) (at level 30).
```

3.3.2. Operaciones

En esta sección se presentan las operaciones definidas sobre los tipos de datos presentados en la sección anterior. Las operaciones se especifican a través de funciones, posiblemente recursivas, sobre cualquiera de los tipos de datos que se han definido. Para indicar que se trata de una función recursiva, se emplea la palabra reservada `Fixpoint`; la especificación de la función debe respetar la definición recursiva del tipo de dato sobre el que ésta trabaja. A esto se le conoce como definición dirigida por la sintaxis. Las llamadas recursivas de la función sólo se pueden hacer sobre variables y estas variables se obtienen a partir del parámetro recibido inicialmente a través del *casamiento de patrones* [7]. Estos conceptos se ven reflejados y se explican en los siguientes ejemplos:

```
Fixpoint conc (G G' : ctx) : ctx :=
   match G' with
     | empty => G
     | snoc G1 p => snoc (conc G G1) p
   end.

Notation "G ; G'" := (conc G G')  (at level 20).
```

El código anterior representa la operación *concatenación* de dos contextos. El casamiento de patrones se aplica al segundo contexto recibido `G'` y la llamada recursiva se realiza estrictamente sobre un subtérmino de `G'`. En el caso de que la llamada recursiva no se haga sobre un subtérmino de alguno de los argumentos, Coq simplemente rechazará la función recursiva (pues probablemente no terminará de computarse). De esta manera, el mecanismo recursivo aceptado por Coq es la recursión estructural. Sin embargo, si en realidad la función termina, se pueden emplear mecanismos más complejos para redefinir la función [8]. Otras de las operaciones definidas sobre contextos son `elem` y `nb_occ`; la primera indica si una proposición es elemento de un contexto y la segunda calcula el número de ocurrencias de una proposición en un contexto dado.

```
Fixpoint elem (a:Proposition) (G:ctx)  : Prop :=
  match G with
  | empty => False
  | G',b => b = a  elem a G'
  end.
```

Para la operación que calcula el número de ocurrencias de una proposición `A` en un contexto `G`, escrito como `nb_occ G A`, el casamiento de patrones se realiza sobre la estructura del contexto `G`. De acuerdo con la definición estructural, para el cálculo del número de ocurrencias, hay que analizar dos posibilidades:

- $G = empty$
  En este caso, el número de ocurrencias de la proposición `A` en el contexto *empty*, es 0.
- $G = G', p$, para algún contexto G' y p una proposición.
  Aquí se derivan nuevamente dos casos:
  - $A = p$, entonces se calcula el número de ocurrencias de `A` en `G'` y se le suma 1.
  - $A \neq p$, entonces resta calcular únicamente el número de ocurrencias de `A` en `G'`.

[7] En inglés *pattern matching.*

[8] El desarrollo de este trabajo requiere únicamente de definiciones recursivas estructurales, sin embargo, existen otros mecanismos que permiten definir funciones recursivas no estructurales, en Coq, como Function

Intuitivamente la codificación de la idea anterior resulta como sigue:

```
Fixpoint nb_occ (G:ctx) (A:Proposition) : nat :=
   match G with
     | empty => 0%nat
     | snoc G' p  =>  if p = A then  S (nb_occ G' A) else  nb_occ G' A
   end.
```

Sin embargo, bajo esta formalización, Coq responde lo siguiente:

```
Error: The term "p = A" has type "Prop" which is not a (co-)inductive type.
```

Como se puede observar, el término `p = A` no es un elemento de tipo booleano que nos indique la igualdad entre dos proposiciones, sino que pertenece al tipo `Prop`, por lo que no puede emplearse en la definición como una guardia (en contraste con la expresión $a = b$ en la definición de `elem`). Es importante aclarar que `Prop` es el conjunto de todas las expresiones demostrables (es decir, de las cuales se puede construir una prueba), mientras que `Bool` se refiere a las expresiones que se pueden evaluar a verdadero o falso. El hecho de que expresiones del tipo `p = A` sean del tipo Prop, indican que debe proporcionarse una prueba de ellas, por lo que la igualdad, en este caso igualdad entre proposiciones, debe ser decidible.

Para resolver este problema se utiliza el tipo `sumbool`, que como se describe en la biblioteca *Coq.Init.Epecif*, es un tipo booleano equipado con la justificación de su valor, veamos a continuación cómo es que se emplea en la solución de nuestro problema.

El tipo `sumbool` se define inductivamente como sigue:

```
Inductive sumbool (A B : Prop) : Set :=
   left : A -> sumbool A B  | right : B -> sumbool A B.
```

Coq proporciona una convención sintáctica para este tipo, donde `sumbool A B` se escribe como `{A} + {B}`. Este tipo inductivo se adapta bien para describir funciones de análisis que deben devolver valores booleanos en lenguajes de programación usuales. En la mayoría de estos lenguajes, es posible verificar si dos valores de un mismo tipo son iguales (bajo ciertas restricciones cuando se trata de valores que tienen un tipo funcional). Coq garantiza esta propiedad de decibilidad a través de la siguiente especificación:

$$\texttt{Definition eq\_dec (A:Type) := } \forall \texttt{ x y :A , } \{x = y\} \ + \ \{x \neq \ y\}$$

Esta definición expresa que para cualesquiera dos elementos de tipo A, sucede que o son iguales o son diferentes pero nunca ambos. La convención en las especificaciones en Coq es nombrar con el sufijo "_ dec", a aquellas funciones que se utilizan para decidir la igualdad entre dos elementos.

En nuestro caso, se han desarrollado las siguientes propiedades de decidibilidad entre dos proposiciones y entre dos juicios:

```
Proposition eq_p_dec (A B:Proposition): {A = B} + {A <> B}.

Proposition eq_j_dec (A B:Judgm): {A = B} + {A <> B}.
```

Entonces la definición del número de ocurrencias es la siguiente:

```
Fixpoint nb_occ (G:ctx) (A:Proposition) : nat :=
   match G with
    | empty => 0%nat
     | snoc D p  =>  match eq_p_dec p A with
                    | left _ =>  S (nb_occ D A)
                    | right _ =>  nb_occ D A
                    end
   end.
```

Lo anterior se interpreta como sigue: si el contexto `G` es vacío, el número de presencias de la proposición `A` en `G` es cero. Si el contexto no es vacío se puede escribir como `G = snoc D p`; puesto que `eq_p_dec` está descrita como una disyunción exclusiva, si ocurre el lado izquierdo, es decir `left _`, sucede que p=A , es decir, el número de presencias de `A` en `G` es el número de presencias de `A` en `D`, más una unidad, pues $p = A$. Por el contrario, si se tiene el lado derecho de la suma booleana, es decir `right _`, significa que $p \neq A$ por lo que de acuerdo a la definición que dimos en un principio, el número de presencias de `A` en `G` es igual al número de presencias de `A` en `D`.

En la siguiente sección se presentan las propiedades lógicas que cumple el sistema $DLMI_{NP}$ con sus respectivas pruebas.

3.3.3. Propiedades del sistema $DLMI_{NP}$ y sus pruebas

Con base en las definiciones de la sección 3.3.1 y las operaciones definidas en 3.3.2 ahora se busca construir pruebas de propiedades que cumple el sistema. Como preámbulo, se expone la noción del principio de inducción que genera Coq para cada estructura de datos definida inductivamente. También se enlistan y explican algunas de las tácticas con las que se trabaja en la construcción de pruebas.

Principio de inducción

Al definirse estructuras de datos de forma inductiva, tales como `Proposition, ctx` y `ND_Proof`, el asistente de pruebas define automáticamente un principio de inducción asociado. Este principio se enuncia en un teorema que se aplica automáticamente cuando se utiliza la táctica *induction* al escribir una prueba que se construye por inducción estructural. Para los tipos definidos `Proposition, ctx` y `ND_proof`, el principio de inducción asociado se enuncia en `Proposition_ind, ctx_ind` y `ND_proof_ind` respectivamente. En las siguientes líneas, se presentan y explican brevemente los principios de inducción para los primeros dos tipos. Por su cuenta, el lector puede revisar y analizar `ND_proof_ind` con el comando `Check ND_Proof_ind.`

```
ctx_ind
     : forall  P : ctx -> Prop,
       P empty ->
       (forall c : ctx, P c -> forall p : Proposition,  P (snoc c p))  ->
       forall c : ctx, P c
```

Las líneas de código del principio de inducción para contextos expresan: Sea `P` una propiedad[9] sobre contextos, si se muestra que el contexto vacío cumple `P` y además, si a partir de la suposición

[9]En Coq una propiedad es una función con contradominio en Prop

de que cualquier contexto `c` cumple `P`, entonces se muestra que la construcción `snoc c p` también cumple `P` para cualquier proposición `p`. Este principio de inducción nos permite concluir que cualquier contexto cumple la propiedad `P`.

La idea en el principio de inducción para `Proposition` es similar. Sea `P` una propiedad sobre elementos de tipo `Proposition`, para concluir que cualquier proposición cumple la propiedad `P`, basta mostrar que todo elemento de la forma `Varp n` (generado por la regla base de la definición recursiva) cumple la propiedad; después es necesario mostrar que los elementos construidos a través de las reglas recursivas es decir, `p1 ==> p2`, `# p1` y `<>p1`, también cumplen la propiedad, teniendo como hipótesis de inducción que `p1` y `p2` cumplen `P`.

El principio de inducción es el siguiente:

```
Proposition_ind
     : forall P : Proposition -> Prop,
       (forall n : N, P (Varp n)) ->
       (forall p : Proposition,
        P p -> forall p0 : Proposition, P p0 -> P (p ==> p0))  ->
       (forall p : Proposition, P p -> P (#p)) ->
       (forall p : Proposition, P p -> P (<> p)) ->
       forall p : Proposition, P p
```

Tácticas y el razonamiento hacia atrás

Hay una gran colección de tácticas en el sistema de Coq, cada una de las cuales es adecuada para metas con determinada estructura. Cada meta consta de dos partes, un *contexto* y un *objetivo*. Los elementos del contexto usualmente tienen la forma `a:type` o `H:formula` y les llamamos hipótesis; todos estos elementos representan hechos que se asumen ciertos temporalmente y que se utilizan para demostrar el objetivo. En la siguiente tabla (tomada de [7]) se enlistan algunas tácticas básicas. Es importante recordar que cada conectivo puede aparecer en el contexto de hipótesis (y entonces se asume que esta hipótesis se ha nombrado `H`) o bien en el objetivo de la meta; de aquí que la táctica que se utilizará no sea la misma en ambos de los casos. Se deben identificar la táctica a utilizar dependiendo de si existe presencia del conectivo en las hipótesis o en el objetivo de la meta.

	$\rightarrow$	$\forall$	$\wedge$
Hipótesis	apply H	apply H	elim H case H destruct H as [H1 H2]
Objetivo	intro/intro H	intro/intro H	split

	$\neg$	$\exists$	$\vee$
Hipótesis	elim H case H	elim H case H destruct H as [x H1]	elim H case H destruct H as [H1 H2]
Objetivo	intro/intro H	exists v	left o right

	$=$	False	
Hipótesis	rewrite H rewrite $<-$ H	elim H case H	
Objetivo	reflexivity		

En esta tabla en el primer bloque, al colocarse en el renglón de las hipótesis y en la última columna (que tiene el símbolo de conjunción), se puede leer por ejemplo, que si para probar un objetivo se desea utilizar una hipótesis `H` cuyo conectivo principal es una conjunción, se pueden aplicar las tácticas `elim H` , `case H` o `destruct H as [H1 H2]`. Por otro lado si el objetivo de la meta que se quiere probar tiene como conectivo principal una conjunción (posicionándose en el renglón de objetivo, columna correspondiente a la conjunción), entonces se debe aplicar la táctica `split`.

Los siguientes párrafos explican brevemente cómo funcionan algunas de estas tácticas. Sean HS una lista de hipótesis y A, B proposiciones. Emplearemos la siguiente notación: $[HS \parallel A]$ representa la meta con la lista de hipótesis HS y objetivo A.

- `intro` : Para probar la meta $[HS \parallel A \to B]$, basta probar $[HS, A \parallel B]$.
- `apply H`: Para probar $[HS, H : A \to B \parallel B]$, basta probar la meta $[HS \parallel A]$.
- `split` : Para probar $[HS \parallel A \wedge B]$, basta probar $[HS \parallel A]$ y $[HS \parallel B]$.
- `elim H` : Para probar $[HS, H : A \wedge B \parallel C]$, basta probar que $[HS \parallel A \to B \to C]$.
- `left` : Para probar $[HS \parallel A \vee B]$, basta probar $[HS \parallel A]$.
- `elim H` : Para probar $[HS, H : A \vee B \parallel C]$, basta probar $[HS \parallel A \to C]$ y $[HS \parallel B \to C]$.
- `destruct H as [H1 H2]`: Para probar $[HS, H : A \wedge B \parallel C]$, basta probar $[HS, H1 : A, H2 : B \parallel C]$.

Hay además, otras tácticas de importancia que si bien no corresponden a la introducción o eliminación de conectivos lógicos, también son parte de la construcción de las pruebas. Ejemplos de estas tácticas son:

- `exact H`: La prueba de $[HS, H : A \parallel A]$, está finalizada.
- `reflexivity`: El razonamiento sobre proposiciones que contienen símbolo de igualdad recae en esta táctica que se emplea cuando se busca probar que dos expresiones son iguales.
- `trivial`: La prueba de la meta $[HS, A \parallel A]$, ha finalizado, no hay nada más que hacer.
- `rewrite H`: Para probar $[HS, H : A = B \parallel C]$, basta probar $[HS, H : A = B \parallel C[A := B]]$, donde $C[A := B]$ significa, reemplazar en C todas las presencias de A por B.

El ejemplo a continuación, muestra la prueba de la siguiente propiedad:
Proposición : $(p \wedge q \to r) \to p \to q \to r$.

Es importante notar que las reglas de derivación que se utilizan a continuación son las reglas de Deducción natural usuales para la lógica proposicional; se busca en este ejemplo reflejar el uso de las tácticas, obsérvese que p, q y r son variables proposicionales cualesquiera, que en Coq son declaradas de tipo `Prop` y que son completamente ajenas a los elementos que se construyen dentro del sistema $DLMI_{NP}$. La siguiente es una muestra de una prueba con deducción natural usual, que corresponde al razonamiento hacia adelante.

Proposición: $\vdash (p \wedge q \to r) \to p \to q \to r$.

1 $(p \wedge q \to r)\ ,\ p\ ,\ q \vdash p$ hyp

2 $(p \wedge q \to r)\ ,\ p\ ,\ q \vdash q$ hyp

3 $(p \wedge q \to r)\ ,\ p\ ,\ q \vdash p \wedge q$ $(\wedge I\ \ 1, 2)$

4 $(p \wedge q \to r)\ ,\ p\ ,\ q \vdash p \wedge q \to r$ hyp

5 $(p \wedge q \to r)\ ,\ p\ ,\ q \vdash r$ $(\to E\ \ 4, 3)$

6 $(p \wedge q \to r)\ ,\ p \vdash q \to r$ $(\to I\ \ 5)$

7 $(p \wedge q \to r) \vdash p \to q \to r$ $(\to I\ \ 6)$

8 $\vdash (p \wedge q \to r) \to p \to q \to r$ $(\to I\ \ 7)$

Con el enfoque del razonamiento hacia atrás, se presenta una tabla que muestra la relación entre hipótesis, conclusión de la meta y táctica aplicada en cada paso de la construcción de la prueba para la misma proposición.

Hipótesis	Objetivos	Táctica
	$(p \wedge q \to r) \to p \to q \to r$	
H: $p \wedge q \to r$	$p \to q \to r$	intro
H: $p \wedge q \to r$ H0: p	$q \to r$	intro
H: $p \wedge q \to r$ H0: p H1 : q	r	intro
H: $p \wedge q \to r$ H0: p H1 : q	$p \wedge q$	apply H.
H: $p \wedge q \to r$ H0 : p H1 : q	p q	split
H: $p \wedge q \to r$ H0 : p H1 : q	q	trivial
	$\square$	trivial

La tabla se interpreta de la siguiente manera:
En el primer renglón la meta consiste en construir una prueba de $(p \wedge q \to r) \to p \to q \to r$, con un conjunto vacío de hipótesis. En el segundo renglón, al aplicar la táctica `intro` al objetivo del primer renglón, la nueva meta consiste en dar una prueba para $p \to q \to r$, apoyándose de la hipótesis H que asume que $p \wedge q \to r$ es verdadera. Para el tercer paso, se aplica nuevamente la táctica `intro` al objetivo del renglón anterior, de esta manera el nuevo objetivo es probar $q \to r$, utilizando las hipótesis $H : p \wedge q \to r$ y $H0 : p$.

Saltando al quinto renglón de la tabla, se puede leer que para probar el objetivo (del renglón 4) r, se utilizará la táctica `apply` sobre la hipótesis $H : p \wedge q \to r$. Esta táctica establece que para probar el consecuente de una implicación, basta probar el antecedente de la misma, por ello el nuevo objetivo se convierte en $p \wedge q$, que es precisamente el antecedente de la hipótesis H.
El siguiente paso emplea la táctica `split`, la cual indica que para probar una conjunción, basta construirlas pruebas del rango izquierdo y el rango derecho de la conjunción, bajo las mismas hipótesis. En el último renglón se observa que para probar el objetivo q, se hace uso de la táctica `trivial`, puesto que el objetivo se encuentra en el conjunto de hipótesis. De esta forma, no quedan más objetivos por procesar y la prueba ha sido finalizada.

En matemáticas una prueba es absoluta. Básicamente, la corrección de una prueba puede ser determinada por cualquiera que conozca del tema. Una prueba matemática puede ser expresada como una serie de pequeños y muy simples pasos, cada uno de los cuales se puede verificar de forma sencilla que son irrefutables. Estos pasos son tan pequeños y simples que en las pruebas que hacemos en papel o que se encuentran en libros y artículos, ni siquiera es necesario mencionarlos, aunque podrían ser explicados a detalle sin mayor problema.

Se expone esta idea para respaldar la afirmación de que, entre los que desarrollan una prueba y los que leen la misma, hay un acuerdo implícito sobre la validez de estos pasos básicos y los métodos de combinación de éstos para crear pruebas más complejas. Todo esto se refleja y genera un problema, al intentar mecanizar la verificación de una prueba matemática, pues la validez de todos estos pasos básicos que se dan por hecho en las pruebas en papel, tiene que demostrarse en la formalización dentro del asistente de pruebas, ya que éste no puede hacer uso de los pasos básicos si no se ha demostrado su validez. Lo anterior representa una brecha considerable, a nivel de detalle, entre las pruebas que se escriben en los libros y artículos, y las pruebas que se construyen en un asistente de pruebas.

Un concepto importante en el desarrollo de la formalización y demostración de propiedades del sistema de deducción, es la equivalencia de contextos. En la definición formal que se ha expuesto, un contexto se considera un conjunto de proposiciones, sin embargo, para su formalización se representa con listas, lo cual obliga a demostrar las propiedades que cumplen los contextos siendo conjuntos, en su representación con listas. Para ello se utiliza la siguiente definición de equivalencia de contextos, basada en la definición del número de ocurrencias de una proposición en un contexto y que establece que dos contextos G y G' son equivalentes si el número de ocurrencias de cualquier proposición es el mismo en ambos contextos.

```
Definition equiv (G G' : ctx) :=
           forall (z : Proposition), (nb_occ G z) = (nb_occ G' z) .
```

Antes de exponer las pruebas de las propiedades más importantes del sistema de deducción, se enlistan algunas propiedades básicas, que en las demostraciones formales se dan por sentadas pero que en la formalización se han tenido que verificar.

Propiedades sobre contextos

El objetivo de esta sección es enlistar propiedades sobre contextos, aquellas que han sido imprescindibles en las pruebas de propiedades acerca del sistema $DLMI_{NP}$. En busca de continuar con la comprensión de los elementos de Coq que se han descrito hasta el momento, se ha seleccionado al azar una de las propiedades sobre contextos, expondremos su demostración matemática y haremos un análisis de su formalización y la prueba su Coq.

Proposición 9. *Para cualquier contexto G, (empty;G) = G, es decir, la concatenación del contexto vacío con cualquier otro contexto, resulta ese mismo contexto.*

Demostración. La demostración se realiza por inducción sobre el contexto G.

- $G = empty$.
  P.D. $empty; empty = empty$. Aplicando la definición de *conc* en el lado izquierdo de la igualdad obtenemos

  $empty = empty$, lo cual es cierto por reflexividad.

- Hipótesis de inducción: (empty;G') = G', para cualquier contexto G',

- $G = G', p$
  P.D. $empty; (G', p) = G', p$, Aplicando la definición de *conc* en el lado izquierdo tenemos

  $(empty; G'), p = G', p$, utilizando la hipótesis de inducción se concluye

  $G', p = G', p$, lo cual es cierto por reflexividad.

□

La formalización y prueba de esta propiedad en Coq es la siguiente:

```
Proposition E_conc : forall (G : ctx) , (empty;G) = G.
Proof.
intros.
induction G.
simpl.
reflexivity.
rewrite <- IHG.
simpl.
rewrite IHG.
rewrite IHG.
reflexivity.
Qed.
```

La táctica `simpl` realiza una manipulación algebraica del término buscando obtener una expresión más simple. La relación entre metas y tácticas que se genera y aplica en el desarrollo de la prueba se puede observar en la siguiente tabla:

Hipótesis	Objetivos	Táctica
	$forall\ G : ctx, (empty\,;G) = G$	
$G : ctx$	$empty\,;G = G$	intro
$G : ctx$	$empty\,;\ empty = empty$ $empty\,;(G,p) = G,p$	induction G
$G : ctx$	$empty = empty$ $empty\,;(G,p) = G,p$	simpl
$G : ctx$ $IHG : empty;G = G$	$empty\,;(G,p) = G,p$	reflexivity
$G : ctx$ $IHG : empty;G = G$	$(empty\,;G),p = G,p$	simpl
$G : ctx$ $IHG : empty;G = G$	$G,p = G,p$	rewrite IHG
	□	reflexivity

Se recomienda al lector ejecutar y analizar algunas de las pruebas de las propiedades que se enlistan a continuación.

- `eq_snoc: forall (G G':ctx) (A:Proposition),G = G' -> G,A = G',A.`
- `conc_E : forall (G : ctx) (A : Proposition),(G,A) = ((G,A) ; empty).`
- ```
  perm0: forall (G G' : TrueHyps) (A B: Proposition),
                  (((G,A);G'),B) = ((G,A);(G',B)) .
  ```

Las siguientes son propiedades que involucran a la función `elem`, que revisa si una proposición es elemento de un contexto.

- `elem_empty : forall (a:Proposition), ~ elem a empty.`

  En este enunciado el símbolo `~`, corresponde a la negación por lo que el enunciado significa que ninguna proposición es elemento del conjunto vacío.
- `elem_head : forall (a:Proposition) (G:ctx), elem a (G,a).`
- `elem_cons : forall (a b:Proposition) (G:ctx), elem b G -> elem b (G,a).`
- `elem_1: forall (A : Proposition) (G G' : ctx),  elem A ((G,A);G').`
- `elem_inv : forall (A B:Proposition) (G:ctx), elem B (G,A) -> (A = B)  elem B G.`
  ```

- `elem_split : forall (A : Proposition) (G : ctx) , elem A G -> exists G1, exists G2, G=(G1,A);G2 .`
- `nocc_elR: forall (G:ctx) (A:Proposition), elem A G -> nb_occ G A <> 0%nat .`
- `nocc_el : forall (G:ctx) (A:Proposition), nb_occ G A <> 0%nat -> elem A G.`
- `elem_union_split : forall (A:Proposition) (G G':ctx), elem A (G;G') -> elem A G  elem A G'.`
- `elem_union_L: forall (A:Proposition) (G:ctx), elem A G -> forall (G':ctx), elem A (G;G').`
- `elem_union_R: forall (A:Proposition) (G G':ctx), elem A G' -> elem A (G;G').`
- `elemSplit: forall (G: ctx) (A B: Proposition),  B <> A -> elem B G -> forall (G0 G0': ctx), G ~= (G0,A);G0' -> elem B (G0;G0').`

Por último se enlistan las propiedades que involucran el concepto de contextos equivalentes. Obsérvese que, en particular, las primeras tres propiedades establecen que la equivalencia de contextos es una relación de equivalencia.

- `equiv_refl: forall (G : ctx), equiv G G.`
- `equiv_sym: forall (G G': ctx),  equiv G G' -> equiv G' G.`
- `equiv_trans: forall (G G1 G2: ctx), equiv G G1 -> equiv G1 G2 -> equiv G G2.`
- `equiv_snoc : forall (A : Proposition) (G G': ctx), equiv G G' -> equiv (G , A) (G', A).`
- `equiv_conc: forall (G G' G1 : ctx), equiv G G' -> equiv (G;G1) (G';G1).`
- `equiv_perm: forall (A B : Proposition) (G G': ctx), equiv G G' -> equiv ((G,A),B) ((G',B),A).`
- `equiv_perm1: forall (A : Proposition) (G G' : ctx), equiv ((G ; G') , A) ((G , A) ; G').`
- `equiv_commutative: forall (G G' : ctx),  equiv (G;G') (G';G).`
- `equiv_perm3: forall (A : Proposition) (G G' : ctx), equiv ((G , A) ; G') ((G' , A) ; G) .`
- `equiv_4: forall (G G1 G2: ctx), equiv G1 G2 -> equiv (G1; G) (G2; G) .`
- `eq_nempty: forall (G:ctx) (p:Proposition), ~ equiv (G,p) empty.`
- `eq_empty: forall (G:ctx), equiv G empty -> G = empty.`
- `eq_elem : forall (A : Proposition) (G G':ctx), equiv G G' -> elem A G -> elem A G'.`
- `eq_elsplit: forall (A : Proposition) (G G': ctx), equiv G G' -> elem A G -> exists G1, exists G2, G' = (G1,A);G2.`

Propiedades del sistema $DLMI_{NP}$

A lo largo del capítulo 2, se presentaron las propiedades estructurales que satisface el sistema $DLMI_{NP}$, como el intercambio, debilitamiento y contracción, tanto para contextos de hipótesis verdaderas como de hipótesis válidas, además se han demostrado formalmente. También se ha enunciado el principio de sustitución para este sistema de inferencia con implicación, necesidad y posibilidad. Este principio se ha formulado en tres partes, una que corresponde a hipótesis verdaderas, la segunda corresponde a hipótesis válidas, la tercera y última considera únicamente juicios que derivan posibilidad (expresiones de la forma *A posible*). Las propiedades estructurales, así como el principio de inducción que se han formalizado se enlistan a continuación y sus respectivas demostraciones en Coq se pueden consultar en el material complementario [10] :

Las propiedades de

- *Intercambio*
  Si $\Delta \mid \Gamma, A, B, \Gamma' \vdash C$ entonces $\Delta \mid \Gamma, B, A, \Gamma' \vdash C$
- *Debilitamiento*
  Si $\Delta \mid \Gamma, \Gamma' \vdash C$ entonces $\Delta \mid \Gamma, B, \Gamma' \vdash C$
- *Contracción*
  Si $\Delta \mid \Gamma, A, A, \Gamma' \vdash C$ entonces $\Delta \mid \Gamma, A, \Gamma' \vdash C$

corresponden a las siguientes formalizaciones:

```
Proposition nd_exchange:
             forall (D: ValHyps) (G G': TrueHyps) (A B: Proposition) (C:Judgm),
                  ND_Proof D ( (G,A),B ; G' ) C -> ND_Proof D ((G,B),A ; G') C.

 Proposition nd_weakening :
       forall (D: ValHyps) (G G' : TrueHyps) (A : Judgm),
       ND_Proof D (G ; G') A -> forall (B : Proposition), ND_Proof D (G, B ; G') A.


Proposition nd_contraction:
               forall (D: ValHyps) (G G': TrueHyps) (A: Proposition) (C:Judgm),
                    ND_Proof D ( (G,A),A ; G' ) C -> ND_Proof D (G,A ; G') C.
```

Como se explicó en la sección 2.3, la incorporación de la regla *nd_tp* al sistema no permitía la construcción de la prueba para el teorema de sustitución tal y como se enuncia en [24]. Se genera una interdependencia en la cual la demostración del inciso a) requería la demostración del inciso b) y viceversa; lo mismo con los incisos c) y d). Para resolverlo, se planteó la siguiente formulación del teorema de sustitución, presentada en el Teorema 4 de dicha sección y que, para facilitar su consulta, reproducimos a continuación:

Sustitución en $DLMI_{NP}$: El sistema de inferencia para la lógica modal $DLMI_{NP}$ satisface:

ab) Si $\Delta \mid \Gamma \vdash A$ *verdadera* y $\Delta \mid \Gamma, A, \Gamma' \vdash J$ entonces $\Delta \mid \Gamma, \Gamma' \vdash J$

cd) Si $\Delta \mid \cdot \vdash B$ *verdadera* y $\Delta, B, \Delta' \mid \Gamma \vdash J$ entonces $\Delta, \Delta' \mid \Gamma \vdash J$

[10] Este material complementario se refiere al script Archivo1.v que se adjunta a la publicación de esta tesis.

e) Si $\Delta \mid A \vdash C\ posible$ y $\Delta \mid \Gamma \vdash A\ posible$ entonces $\Delta \mid \Gamma \vdash C\ posible$

Donde J es un juicio, es decir, $J = C\ verdadera$ o bien $J = C\ posible$.

Su formalización, usando la notación introducida al final de la sección 3.3.1, es la siguiente:

```
Theorem substitutionT: forall (D: ValHyps) (G: TrueHyps) (A : Proposition),
                        (D | G |-- A) ->
                        forall (G':TrueHyps) (J:Judgm),
                               (D | ((G,A);G') |--j  J)
                                  -> (D | (G;G') |--j J).


Theorem substitutionJV: forall (D : ValHyps) (B : Proposition),
                        (D | empty |-- B) ->
                        forall (D':ValHyps) (G:TrueHyps) (J:Judgm),
                               (((D,B);D') | G |--j  J)
                                  -> ((D;D') | G |--j J).

Theorem substitutionT5: forall (D: ValHyps) (A C: Proposition),
                        (D | (empty,A) |--p C) ->
                        forall (G:TrueHyps),
                               (D | G |--p  A)
                                  -> (D | G |--p C).
```

Para la construcción de las pruebas de estas proposiciones y teoremas se han empleado propiedades básicas, que se obvian en las demostraciones matemáticas pero que ha sido imprescindible construir su prueba en la formalización del sistema. Se enlistan a continuación:

```
Proposition equiv_Proof: forall (A : Judgm) (D : ValHyps) ,
                     forall (G:TrueHyps), ND_Proof D G A ->
                     forall (G':TrueHyps), equiv G G' -> ND_Proof D G' A.

Proposition equiv_ProofV: forall (A : Judgm) (G : TrueHyps) (D:ValHyps),
                     ND_Proof D G A ->
                     forall (D':ValHyps), equiv D D' -> ND_Proof D' G A.

Proposition nd_perm1:
             forall (D: ValHyps) (G G' : TrueHyps) (A :Judgm) (B: Proposition),
             ND_Proof D ((G;G'),B) A -> ND_Proof D ((G,B);G') A.

Proposition nd_perm1V:
             forall (D D': ValHyps) (G: TrueHyps) (A :Judgm) (B: Proposition),
             ND_Proof ((D;D'),B) G A -> ND_Proof ((D,B);D') G A.

Proposition nd_perm0: forall (D: ValHyps) (G G' : TrueHyps) (A B: Proposition),
             ND_Proof D ((G,A);(G',B)) (JTrue A) ->
```

```
             ND_Proof D (((G,A);G'),B) (JTrue A).

Proposition nd_perm0V: forall (D D': ValHyps) (G : TrueHyps) (A B: Proposition),
             ND_Proof ((D,A);(D',B)) G (JTrue A) ->
             ND_Proof (((D,A);D'),B) G (JTrue A).

Proposition nd_weak_last : forall (D: ValHyps) (G: TrueHyps) (A : Judgm),
             ND_Proof D G A -> forall (B : Proposition), ND_Proof D (G, B) A.

Proposition nd_weak_lastV : forall (D: ValHyps) (G: TrueHyps) (A : Judgm),
             ND_Proof D G A -> forall (B : Proposition), ND_Proof (D, B) G A.


Proposition weakCtx: forall (D: ValHyps) (G : TrueHyps) (A: Judgm),
             ND_Proof D G A -> forall (G': TrueHyps), ND_Proof D (G;G') A.

Proposition weakCtxV: forall (D: ValHyps) (G : TrueHyps) (A: Judgm),
             ND_Proof D G A -> forall (D': ValHyps), ND_Proof (D;D') G A.

Proposition ndphyp: forall (D: ValHyps) (G : TrueHyps) (A: Proposition),
            elem A G -> ND_Proof D G (JTrue A).

Proposition nd_vhypE: forall (D: ValHyps) (G : TrueHyps) (A: Proposition),
            elem A D -> ND_Proof D G (JTrue A).
```

Como habrá notado el lector, esta formalización verifica la relación de derivabilidad de un juicio a partir de un contexto de fórmulas y aborda propiedades de esta relación; no debe considerarse como un verificador al que se le puede proporcionar una fórmula para decidir si ésta es verdadera o válida.

Con lo anterior concluye el capítulo de formalización. Se presentó un panorama de los asistentes de prueba y una breve introducción a Coq. Se expuso la formalización del sistema de Deducción Natural para la lógica modal intuicionista, y a través de ejemplos de código se fueron explicando las características del asistente de pruebas. El siguiente y último capítulo de esta tesis, exhibe las conclusiones de lo desarrollado hasta el momento, así como algunos puntos considerados trabajo a futuro que se han detectado en el desarrollo de esta formalización.

Conclusiones y Trabajo futuro

3.4. Conclusiones

La meta principal y producto final de este trabajo es ofrecer una formalización de un sistema de deducción natural para la lógica modal intuicionista, sobre un asistente de pruebas, es este caso Coq. En la primera parte se dio una breve introducción a los marcos lógicos y se presentaron los argumentos que sostienen la invalidez del teorema de la deducción en la lógica modal, sugiriendo que el problema radica en una interpretación errónea de la regla de necesitación.

Es importante mencionar que el sistema $DLMI_{NP}$ satisface trivialmente el Teorema de la Deducción, debido a la inclusión de la regla de introducción de la implicación.

En el segundo capítulo, se presentó y analizó el sistema de deducción natural para la lógica modal intuicionista, propuesto por Pfenning y Davies en [24]. A lo largo del capítulo se replantean los fundamentos de la lógica modal, siguiendo la metodología de Martin-Löf, la cual permite definir juicios hipotéticos y categóricos con los cuales se representa la relación de derivabilidad de una fórmula de la lógica modal intuicionista a partir de un conjunto de fórmulas verdaderas y un conjunto de fórmulas válidas. Del análisis sobre el significado de los operadores modales de necesidad y posibilidad se obtuvieron las reglas respectivas de introducción y eliminación de tales operadores lógicos en el sistema $DLMI_{NP}$.

Debido a que en el sistema discutido no se incluye la negación, no se puede afirmar que los operadores de necesidad y posibilidad sean duales. Por esta razón surge la pregunta acerca de qué lógica es la que se ha discutido. A continuación se hace un breve análisis para responderla.

Basado en el trabajo de Viganó (1997), Kobayashi (1997) y Alechina (1998), Pfenning afirma que el concepto de necesidad puede ser caracterizado axiomáticamente a través de la regla de necesitación

$$\frac{\vdash A \; verdadera}{\vdash \Box A \; verdadera} \; (nec)$$

junto con los siguientes tres axiomas:

$$\begin{array}{l} \vdash \Box(A \to B) \to (\Box A \to \Box B) \;\; verdadera \\ \vdash \Box A \to A \;\; verdadera \\ \vdash \Box A \to \Box\Box A \;\; verdadera \end{array}$$

que son conocidos como los axiomas K, T y 4 respectivamente. Las derivaciones de estos axiomas en el sistema $DLMI_N$ se pueden consultar en la sección 2.2 en los ejemplos 1,2 y 3. También se puede encontrar la construcción en Coq de las pruebas para cada uno de los axiomas en el material complementario *Archivo1.v*, enunciados como:

```
Theorem Box_K: forall (A B:Proposition),
                  empty | empty |-- ((#(A ==> B)) ==> ((#A) ==> (#B))).

Theorem Box_T: forall (A:Proposition),
                  empty |empty |-- ((#A) ==> A).

Theorem Box_4: forall (A:Proposition), empty | empty |-- #A ==> ##A.
```

Por otro lado, en lo que respecta a la modalidad de posibilidad, se afirma que puede ser caracterizada axiomáticamente por los siguientes elementos :

$$\vdash \Box(A \to B) \to (\Diamond A \to \Diamond B) \ \ verdadera$$
$$\vdash A \to \Diamond A \ \ verdadera$$
$$\vdash \Diamond\Diamond A \to \Diamond A \ \ verdadera$$

cuyas derivaciones se han desarrollado en los ejemplos 4,5 y 6 de la sección 2.3 ; estos axiomas son conocidos como $\Diamond K, \Diamond T$ y $\Diamond 4$, para cada uno de ellos se ha construido una prueba que en la formalización en Coq se pueden encontrar enunciados como sigue:

```
Theorem Dia_K: forall (A B:Proposition),
                     empty | empty |-- #(A ==> B) ==> (<>A ==> <>B).

Theorem Dia_T: forall (A:Proposition), empty | empty |-- (A ==> <> A).

Theorem Dia_4: forall (A:Proposition), empty | empty |-- (<> <> A ==> <> A).
```

Pues bien, son precisamente los axiomas $K, T, 4, \Diamond K, \Diamond T$ y $\Diamond 4$, junto con la regla de necesitación, los que caracterizan el sistema modal S4 constructivo (CS4), el cual es una versión del sistema S4 intuicionista, introducido por Prawitz en 1965; CS4 se obtiene extendiendo la lógica proposicional intuicionista con los operadores modales $\Box$ y $\Diamond$ y los seis axiomas que hemos mencionado. De aquí que la lógica que subyace en el sistema $DLMI_{NP}$ es la lógica CS4.

Hay diversas variantes de la lógica constructiva S4 en la literatura. Sin embargo la lógica CS4 de la que hablamos, es minimal en el sentido de que demuestra un menor número de teoremas que la mayoría de las otras variantes. En particular, no asume que posiblemente falso ($\Diamond\bot$) y falso son equiprobables, es decir, $\Diamond\bot \to \bot$ no es un teorema en este sistema. Como se indica en [2], CS4 es adecuada para probar sobre ella una versión del Isomorfismo Curry-Howard, lo cual se considera una aplicación computacional importante.

El tercer capítulo de este trabajo, se enfocó en la presentación y análisis de la formalización del sistema $DLMI_{NP}$ en el asistente de pruebas Coq. Se presentó un breve panorama de los asistentes de prueba y una corta introducción a Coq. La exposición de las principales características de este asistente de pruebas se realizó a través de ejemplos extraídos de la formalización que se construyó, explicando el significado y funcionamiento de diferentes líneas de código.

3.5. Trabajo a futuro

En esta última sección, se abordará uno de los temas que resultaron de gran interés durante el desarrollo de esta tesis, sobre el que se inició una investigación y que en un futuro se pretende profundizar: la semántica.

La mayoría del trabajo sobre semántica emplea marcos de Kripke para tratar con la lógica modal constructiva. Debido a que estamos trabajando con un sistema de deducción natural y no con sistema de Hilbert, resulta necesario extender o modificar alguna de las semánticas que se han propuesto, de tal forma que se pueda expresar el concepto de consecuencia lógica a partir de un conjunto de fórmulas verdaderas y uno de fórmulas válidas.

Aún cuando la gran mayoría de los autores concluyen que un modelo de Kripke de la Lógica modal constructiva debe consistir de un conjunto de mundos W y dos relaciones de accesibilidad, una intuicionista $\leq$ y otra modal R, no existe un concenso acerca de cómo deberían interactuar estas relaciones y cómo deben ser utilizadas para interpretar necesidad y posibilidad.

Para el breve estudio que se realizó sobre la semántica, se tomaron como base los modelos de Kripke propuestos por Alechina en [2].

Definición 3. *Un modelo de Kripke para la lógica constructiva S4 es una estructura M de la forma $(W, \leq, R, \models)$, donde W es un conjunto no vacío y $\leq$ y R son relaciones binarias reflexivas y transitivas sobre W, tal que:*

- *$\leq$ es hereditaria con respecto a las variables proposicionales, es decir para cada variable p y mundos w, w', si $w \leq w'$ y $M, w \models p$, entonces $M, w' \models p$.*
- *R y $\leq$ se relacionan como sigue: si $w \leq w'$ y $w'Rv$ entonces existe v' tal que wRv' y $v' \leq v$.*
- *La relación $\models$ cumple las siguientes propiedades:*
  - $w \models \top$
  - $w \models A \to B$ *si y sólo si* $\forall w'.w \leq w' \Rightarrow (w' \models A \Rightarrow w' \models B)$
  - $w \models \Box A$ *si y sólo si* $\forall w'.w \leq w' \Rightarrow \forall u.w'Ru \Rightarrow u \models A$
  - $w \models \Diamond A$ *si y sólo sis* $\forall w'.w \leq w' \Rightarrow \exists u.w'Ru \wedge u \models A$

  Nótese que no se considera el caso $w \not\models \bot$, lo que permitiría mundos inconsistentes, en vez de eso se tiene:

  - *Si $w \models \bot$ y $w \leq w'$, entonces $w' \models \bot$*
  - *Si $w \models \bot$, entonces para cada variable proposicional p, se tiene que $w \models p$, (asegurándose de que $\bot \to A$ continúa siendo válida).*

Como es usual, una fórmula A es **verdadera** en un modelo M si para cada $w \in W$, $M, w \models A$. Una fórmula A es **válida** ($\models A$) si es verdadera en todos los modelos; una fórmula es **satisfacible** si existe un modelo y un mundo donde se satisface. La parte crucial, consiste en definir cuándo una fórmula A es **consecuencia lógica** de un conjunto de fórmulas verdaderas Γ y un conjunto de fórmulas válidas Δ.

Se trabajó con las siguientes propuestas, donde $\Delta \mid \Gamma \models A$ significa que la fórmula A es consecuencia lógica de los conjuntos de fórmulas verdaderas Γ y de fórmulas válidas Δ:

Definición 1: $\Delta \mid \Gamma \models A$ si y sólo si para todo M, w, si $M, w \models \Gamma$ y $\forall B \in \Delta, \ \models B$, entonces $M, w \models A$.

Definición 2: $\Delta \mid \Gamma \models A$ si y sólo si para todo M, w, si $M, w \models \Gamma$ y $\forall B \in \Delta$, $w \models \Box B$, entonces $M, w \models A$.

Sin embargo, ninguna de las definiciones anteriores permitió demostrar el teorema de corrección que se formula como sigue:

Teorema 5 (***Corrección del sistema*** $DLMI_{NP}$)**.** *Sea* Γ, Δ *un conjunto de fórmulas verdaderas y válidas, respectivamente, y* A *una fórmula:*

$$\Delta \mid \Gamma \vdash A \ \Rightarrow \ \Delta \mid \Gamma \models A$$

Las anomalías se presentaron en los casos en los cuales la derivación del secuente se obtiene por las reglas de introducción y eliminación del operador de necesidad $\Box I$ y $\Box E_p$ y cuando para la demostración se requería de la regla *nd_tp*. El análisis de la falla es estos intentos de demostración del teorema de corrección, así como la propuesta de una definición adecuada para la relación de consecuencia lógica o bien el uso de una semántica distinta, como la semántica categórica, se considera trabajo a futuro.

Bibliografía

[1] Alchourrón Carlos. *Introducción: Concepciones de la lógica.* En Enciclopedia Iberoamericana de Filosofía. Vol. 7. (pp. 11-47) Editorial Trotta.

[2] Alechina, N; Mendler, M; de Paiva, V; Ritter, E. *Categorical and Kripke Semantics for constructive S4 modallLogic.* Computer Science Logic, 2001; 2001 September 10-13; Paris France.

[3] Avron, A.; Honsel, F.; Mason, I. *Using Typed Lambda Calculus to implement formal systems on a machine.* In Proceedings of the workshop on programming logic. Report 37 ed. P. Dybjer, B. Nordström, K. Petersson, and J. Smith. University of Göteborg. 1987.

[4] Basin, D., Seán Matthews, L. Viganó *A modular Presentation of modal logics in a logical framework*. In The Tbilisi Symposium on Language, Logica and Computation: Selected Papers, CSLI Publications. 1998

[5] Basin, D., Seán Matthews, L. Viganó *Natural deduction for non-classical logics.* Studia Logica 60: pp 119-160. (1998).

[6] Bertot, Y.; Casteran, P. *Interactive Theorem Proving and Program Development.* Texts in Theoretical Computer Science. Springer Verlag 2004.

[7] Bertot Yves *Coq in a Hurry.* Types Summer School, also used at the University of Nice Goteborg, Nice. 2006.

[8] Chagrov A.; Zakharyaschev, M. *Modal logic.* Clarendon Press. 1997.

[9] Chellas, B. *Modal Logic: An introduction.* Cambridge University Press. 1980.

[10] Coquand, T.; Huet, G. *The calculus of constructions.* Information and Computation, 76. 1988.

[11] Fairtlough, M.; Mendler, M. *An intuitionistic modal logic with applications to the formal verification of hardware.* En Proceedings of Conference on Computer Science Lógic, volume 933 of Lecture Notes in Computer Science, 1995.

[12] Fagin R.; Halpern, J.; Moses, Y.; Vardi, M. *Reasoning about knowledge.* MIT Press; !st MIT Press Paperback Ed edition.

[13] Fitting, M.C. *Modal Proof Theory* In Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 85—138. Elsevier, Amsterdam (2007).

[14] Geuvers, J.H. *Proof assistants: History, ideas and future.* Sadhana Journal, Academy Proceedings in Engineering Science, Indian Academy of Sciences. 34(1), Special Issue on Interactive Theorem Proving and Proof Checking, pp. 3–25. 2009.

[15] Gonthier George. *A computer-checked proof of the Four Colour Theorem.* Available at `http://www.research.microsoft.com/~gonthier/4colproof.pdf`, 2005.

[16] Hakli R.; Negri, S. *Does the deduction theorem fail for modal logic?.* Synthese vol. 187, issue 3, pp. 849-867. 2011.

[17] Kripke, A. Saul *A completeness Theorem in Modal Logic.* J. Symbolic Logic 24 (1959), no. 1, pp 1-14.

[18] Lescanne Pierre *Mechanizing Common Knowledge Logic using COQ.* In Annals of Mathematics and Artificial Intelligence. Volume 48, Issue 1-2, pp 15-43. (2006)

[19] Martin-Löf *Constructive Mathematics and Computer Programming.* In Logic, Methodology and Philosophy of Science. VI. pp 153-175, North Holland.

[20] Nederpelt, R. P; Geuvers,J. H.; de Vrijer, R. C. *Selected Papers on Automath* Vol. 133 of Studies in Logic and the Foundations of Mathematics. Elsevier, Amsterdam. 1994.

[21] Paulin-Mohring Christine *Définitions Inductives en Théorie des Types d'Ordre Supérieur.* Habilitation a diriger les recherches, Université Claude Bernard Lyon. 1996.

[22] Pelletier, F.J *A brief history of natural deduction.* History and Philosophy of Logic. 1-31.1999.

[23] Pfenning, F.; Davies, R. *A modal analysis of staged computation.* In Proceedings of Symposium on Principles of Programming Languages, pg. 258-270, 1996.

[24] Pfenning, F.; Davies, R. *A judgmental Reconstruction of Modal Logic.* Department of Computer Science, Carnegie Mellon University, Pittsburgh. 2000.

[25] Smorynski, C. *Modal logic and self-reference.* In Handbook of Philosophical Logic, vol. II pp. 441-495, Reidel. 1984.

[26] Stirling, C.P. *Modal logics for communicating systems.* Technical Report CSR-193-85 Dept. Computer Science, Univ. Edinburgh. 1985.

[27] Calculus of Inductive Constructions: `https://coq.inria.fr/doc/Reference-Manual006.html`

[28] Specific Logical Frameworks and Implementations: `http://www.cs.cmu.edu/~fp/lfs-impl.html`

[29] The Coq Proof Assistant: `https://coq.inria.fr`

[30] The Compcert C compiler: `http://compcert.inria.fr/compcert-C.html`

[31] Theorem Proving and Vampire: `http://www.voronkov.com/vampire.cgi`