Escuela Técnica Superior de Ingeniería Informática
Departamento de Matemática Aplicada

# Reducciones totales y parciales para el análisis de validez y construcción de modelos en M3

Tesis Doctoral de
Gabriel Aguilera Venegas
dirigida por
Inmaculada Pérez de Guzmán

Universidad de Málaga
Febrero de 1997

D$^a$ Inmaculada Pérez de Guzmán Molina, Catedrática de Matemática Aplicada de la E.T.S. de Ingeniería Informática de la Universidad de Málaga,

CERTIFICA:

Que D. Gabriel Aguilera Venegas, Licenciado en Ciencias (Matemáticas), ha realizado en el Departamento de Matemática Aplicada de la E.T.S. de Ingeniería Informática de la Universidad de Málaga, bajo mi dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

# Reducciones totales y parciales para el análisis de validez y construcción de modelos en M**3**

Revisado el presente trabajo, estimo que puede ser presentado al Tribunal que ha de juzgarlo.

Y para que conste a efectos de lo establecido en el artículo octavo del Real Decreto 185/1985, autorizo la presentación de este trabajo en la Universidad de Málaga.

Málaga, a 9 de Febrero de 1997

Dra. Inmaculada Pérez de Guzmán Molina
Catedrática de Matemática Aplicada

A Francisco Sanz
*In memoriam*

. . . Han promulgado una ley prohibiendo la lógica.
— ¿La lógica?— repetí—. ¿La lógica?
Aquello no tenía el menor sentido para mí; era como si me hubiera dicho que había sido promulgada una ley contra los hombres.
— ¿Quién puede prohibir la lógica? La lógica existe.

*"Alexias de Atenas". Mary Renault.*

# Agradecimientos

Empezando por el principio, (¿existe otra posibilidad?) quiero agradecer a mis padres, aparte de mi propia existencia claro, un sin fin de cosas que no puedo escribir (el espacio del que dispongo es necesariamente finito). Pero entresacando algunas, su absoluta abnegación en la ayuda constante que me proporcionan, su claro ejemplo de personas de bien, y sus enseñanzas, gracias a las que, entre otras cosas, sé leer (uno de mis placeres favoritos).

Agradezco a mi mujer, Gemma, no sólo el apoyo explícito prestado para la realización de este trabajo, sino además el constante sacrificio que de su tiempo hace en beneficio mío y de nuestra hija Alba. Sin lugar a dudas, sin ella y su cariño, este momento no habría llegado.

A mi hija Alba le agradezco, sus dibujos sobre (encima de) los numerosos borradores de este trabajo, observando algunos de los cuales he descubierto alguna errata (si no he descubierto todas las erratas, obviamente no es por su culpa, ya que ella ha hecho dibujos sobre todas las copias). Además le agradezco la esperanza.

Gracias también a los demás miembros de mi familia, especialmente a mis hermanos Pepe (y Celia) y Mª Trini (Pepe: todavía no sé quién leerá antes la Tesis). A mi tía Carmen y a mi prima (también Carmen) les agradezco el cariño que siempre me han manifestado.

También les doy las gracias a mi familia "política", aunque me cuesta trabajo llamarlos así, ya que con ellos me he sentido siempre tan bien, que siempre los he considerado simplemente mi familia. Especialmente a Iván, que encontró algunos errores en el borrador y a Magdalena que también se interesó por el trabajo.

Gracias a mis amigos, especialmente a "la panda" hoy desperdigada por Cáceres, Cogollos Vega, El Fargue, Granada y Málaga, su interés por el desarrollo del trabajo ha sido una motivación más para su realización.

A Emilio Lozano quiero agradecerle el apoyo y la amistad que recibí cuando estudiaba la carrera, algunas de las bases de este momento se forjaron aquellos días.

También quiero dar las gracias a mis compañeros del Departamento de Matemática Aplicada (que podían perfectamente haberse incluido en el apartado

de amigos) por su apoyo, y las “liberaciones” que con su trabajo me han proporcionado para que yo pudiese terminar este trabajo. En este sentido estoy especialmente agradecido a Charo y a Paco Rodríguez.

Siguiendo con el apartado de amigos, quiero darles las gracias (y se las doy) a los miembros del grupo de investigación GIMAC. Muy especialmente a Manuel Ojeda y a Agustín Valverde que han tenido la amabilidad de corregir innumerables erratas de este trabajo, si a pesar de ello algunas persisten, es bien porque han sido inclusiones mías de última hora o bien porque yo, por error, no he trasladado algunas de sus correcciones.

Según se comenta, el grupo va a ampliarse con un miembro más, cuyo nacimiento está previsto más o menos para la fecha de lectura de esta tesis. Quiero agradecer este hecho a los dos responsables: José Luis y Mavi.

Agradezco a la familia de Inma, especialmente a Elena, David y Tere, su comprensión por la cantidad de horas que Inma me ha dedicado a causa de este trabajo, y sobre todo agradezco su cariño, de alguna forma ellos también son mi familia.

También quiero agradecer a los profesores Daniele Mundici de la Universidad de Milán, Reiner Hähnle de la Universidad de Karlsruhe y Luisa Iturrioz de la Universidad de Lyon, sus amables comentarios, y sus mensajes de ánimo.

Y para terminar le doy las gracias a Inma, ya que su pasión por el tema, su capacidad de trabajo y su inteligencia han hecho que esta tesis no sólo haya resultado apasionante, sino que además haya sido muy gratificante en cuanto a los resultados obtenidos. No menos agradecido estoy por las enseñanzas humanas que de ella he recibido en este tiempo.

# Índice general

# Introducción

Este trabajo se enmarca en un área de especial interés para el científico de la Computación, el de la demostración automática de teoremas, con atención especial a la búsqueda de ATPs que son métodos de construcción de modelos.

Las motivaciones para la demostración automática de teoremas son muy variadas. El propósito de la automatización puede ser su aplicación al razonamiento sobre una cierta gama de problemas prácticos. Por el contrario, podemos plantearnos estudiar sistemas formales y su posible automatización, por su interés en sí mismo, sin tener en mente aplicación alguna. Nuestro trabajo se centra en el primer propósito.

El interés de nuestro grupo de investigación tiene un objetivo final: contribuir en la consecución de la automatización de las tareas "inteligentes" o "deductivas" en el ámbito de las aplicaciones. No se nos escapa el largo camino que queda aún por recorrer; pero nos alienta, tanto lo atractivo de la tarea, como el comprobar que día a día se avanza en su consecución.

En este trabajo nos centramos en las lógicas multivaluadas. En la bibliografía sobre el tema que nos ocupa, se destaca de modo especial la necesidad de dotar a las lógicas multivaluadas de una teoría de la demostración natural, flexible y adecuada para las aplicaciones.

Somos conscientes de que abordar el objetivo de la automatización de las tareas inteligentes centrándonos en un tipo de lógicas, es tan sólo un paso necesario, pero previo a la consideración de un modo global de razonamiento (más cercano al modo de razonar humano, que no entiende de restricciones). Sin embargo, la investigación llevada a cabo por nuestro grupo a lo largo de estos últimos seis años es, en nuestra opinión, un paso hacia adelante.

En el marco de la demostración automática y basándonos en la necesidad, ampliamente aceptada, de proveer a la programación lógica de una alternativa a la *resolución*, con este trabajo nos proponemos contribuir a este fin, con dos prioridades:

- Conseguir un avance significativo, de cara a consolidar la lógica como el lenguaje de representación más idóneo tanto para la especificación de problemas como para la especificación de programas.

- Rescatar, en la alternativa propuesta, el paralelismo intrínseco de la lógica.

Melvin Fitting en su artículo (Fitting, 1994) expresa de este modo nuestra pretensión:

> *La vida sería casi perfecta si tuviésemos un lenguaje de programación lógica que fuese eficiente, tratara naturalmente la negación y tuviera una semántica que reflejara exactamente su mecanismo computacional. Como todos sabemos, la vida no es casi perfecta. Pero, en un honesto compromiso, trabajamos para acercarnos a este ideal.*

¿Por qué una alternativa a la resolución? El método de resolución es el sistema de demostración automática mejor conocido y más ampliamente desarrollado. Su interés radica en sus dos principales características:

1. Su mecanismo es muy sencillo, pues descansa sólo sobre una regla de inferencia, la *regla de Resolución*.

2. Basa su simplicidad en una representación uniforme de los enunciados: se aplica únicamente a fórmulas en forma normal conjuntiva.

No obstante, este método tiene inconvenientes aún no resueltos (a pesar de los numerosos trabajos de investigación realizados con este objetivo):

1. " . . . Desafortunadamente, la uniformidad en que basa su simplicidad, tiene su precio: todo parece igual. A causa de este parecido general, no existe ninguna forma fácil de seleccionar aquellos enunciados que tienen más probabilidad de ser útiles para resolver un problema concreto. Al convertirlo todo a forma clausal, se pierde a menudo una información heurísticamente valiosa que está contenida en la representación original de los hechos . . . . . Es muy difícil para una persona interaccionar con un demostrador por Resolución. Que tal interacción sea posible, es importante desde un punto de vista práctico, debido a que las computadoras no han alcanzado un comportamiento óptimo en la demostración de problemas complicados.". (Rich, 1983)

2. " . . . El principal atractivo del Método de Resolución, es que tiene una sola regla de inferencia (la regla de resolución). No obstante, existe un precio a pagar: El Método de Resolución se aplica a enunciados en forma normal conjuntiva". (Gallier, 1987)

3. "... lo más irritante sobre la Resolución es su falta de naturalidad. Se trabaja con ella del mismo modo que con cualquier otro sistema formal. Sin embargo, nunca nos permite vislumbrar de qué modo resolveríamos el problema por nosotros mismos. Si un demostrador de teoremas por resolución forma parte de un sistema que requiere explicar su modo de razonar a sus usuarios, sería necesario volver a procesar sus demostraciones para obtenerlas en una forma comprensible, incluso para un experto usuario. Sería conveniente encontrar algún otro motor de inferencia que pudiera trabajar de modo más cercano a como nosotros trabajamos, sus demostraciones serían más fáciles de seguir.

   El segundo aspecto a destacar sobre la inconveniencia de la Resolución, es la necesidad de conversión a forma clausal de las fórmulas. Esto conlleva una gran cantidad de trabajo (la mayoría de las veces innecesario) y a la destrucción de cualquier estructura que tuviera el enunciado original del problema. Podríamos incluso asegurar que es precisamente la conversión a forma clausal la causante de la dificultad de seguir las demostraciones por resolución, ya que la relación de las cláusulas con el enunciado original es con frecuencia casi imposible de desentrañar ... " (Ramsay, 1988)

4. Además de las dificultades referidas en las citas anteriores, podemos señalar otra no menos importante: La dificultad de ser extendido a las lógicas no clásicas en general y a las lógicas multivaluadas en particular.

En síntesis, optamos por una alternativa a la resolución debido a que ésta:

1. Coarta la posibilidad de la especificación directa y natural.

2. Dificulta la comprensión del modo en que se obtienen las respuestas.

3. No admite extensiones naturales a las lógicas no clásicas.

Coincidimos con B. Paech (Paech, 1986), en que "es mejor un proceso de derivación que permita detectar fallos, que sea sencillo y permita seguir la derivación interactivamente, como los sistemas tipo Gentzen".

Un análisis de algunos métodos basados en resolución para las lógicas multivaluadas puede verse en (Hähnle, 1993), que incluye un atractivo capítulo sobre la historia de la demostración automática de teoremas para lógicas multivaluadas.

Destaquemos por último, que una de las motivaciones de este trabajo ha sido nuestra participación en el proyecto COST: "Many Valued Logics for Computer Science Applications" y, en particular, la colaboración con los profesores Daniele Mundici de la universidad de Milán, Reiner Hähnle de la universidad de Karlsruhe y Luisa Iturrioz de la universidad de Lyon.

## Aportación

En este trabajo introducimos un nuevo demostrador automático para la lógica trivaluada completa M**3**, al que denominamos TAS-M**3** y que, al igual que los métodos basados en las *tablas semánticas*, es un sistema basado en los trabajos de Gentzen.

TAS-M**3** es una extensión del demostrador TAS-D para la Lógica Clásica Proposicional y usa la metodología TAS (Aguilera *et al.*, 1995a), que ha sido aplicada a la Lógica Clásica Proposicional (Aguilera *et al.*, 1995b), a la Lógica de Primer Orden (Ojeda, 1996) y a la lógica temporal (de Guzmán y Enciso, 1995).

TAS-M**3** es un método de *reducción*.[1] En definitiva, la potencia de TAS-M**3** se basa en los procesos (*reducciones*) que disminuyen el tamaño de la fórmula. Estos procesos se agrupan en la transformación $\mathcal{F}$. Esta transformación *filtra* dinámicamente la información contenida en la estructura de la fórmula, evitando tantas distribuciones de $\wedge$ sobre $\vee$ como permita tal estructura. Esta es, como mostraremos, la forma en que el método consigue su eficiencia.

La idea básica del método es la utilización de la información suministrada por interpretaciones parciales unitarias. Esta idea, como ya se ha visto para la lógica clásica y para lógicas modales, ha demostrado ser sorprendentemente potente.

Los conjuntos de modelos unitarios, llamados conjuntos $\Delta$, se asocian a cada nodo del árbol sintáctico de la negación de la fórmula cuya validez se quiere comprobar. En nuestra opinión, los conjuntos $\Delta$ pueden considerarse como la piedra angular de la metodología TAS, ya que nos permiten concluir si la estructura de la fórmula tiene o no información directa sobre su validez.

Al igual que los conjuntos de signos en el método de *Conjuntos de valores de verdad como signos* de R. Hähnle (Hähnle, 1993) o las etiquetas en los Sistemas Deductivos Etiquetados de Gabbay (Gabbay, 1993), los conjuntos $\Delta$ pueden ser considerados como herramientas que incorporan, como metainformación, valores de verdad. No obstante, podemos destacar una sustancial diferencia en el uso de esta metainformación:

> *El método TAS-M**3** utiliza únicamente interpretaciones parciales unitarias y con un objetivo específico: disminuir el número de distribuciones.*

Por otra parte, al igual que en los métodos citados, TAS-M**3** maneja paralelamente todos los valores de verdad no destacados. Sin embargo, con los conjuntos $\Delta$ se consigue expresar, con una sola etiqueta, no sólo que una fórmula $A$ puede

[1] El nombre hace referencia a que para evitar distribuciones, las fórmulas son *reducidas* usando distintos procesos.

tomar diversos valores de verdad, sino específicamente cuál es la asignación de valores de verdad a los símbolos proposicionales.

Destacamos a continuación las características del método. TAS-M**3** es:

- Un método de refutación,

es decir, en lugar de probar la validez o no de una inferencia, probaremos la insatisfacibilidad o no de la conjunción de las hipótesis y la negación de la conclusión. De esta forma, podemos hacer uso no sólo de las transformaciones que conservan el significado, sino también de la potencia aportada por las transformaciones que conservan únicamente la insatisfacibilidad.

- Un método cercano a la semántica.

Para justificar esta opción, bastaría destacar que esta propiedad es la causa del auge experimentado por el método de las tablas semánticas. Sin embargo, podemos profundizar más:

Como analiza Peter Andrews en su artículo (Andrews, 1981), la demostración de teoremas es difícil y trata con fenómenos complejos ¿Qué hace que una fórmula sea válida? Podemos hacer un planteamiento semántico de esta cuestión. Los teoremas expresan *verdades esenciales* y por esta razón son verdaderos en todos los modelos para el lenguaje en que son expresados. Pero la verdad puede ser percibida desde múltiples perspectivas, por esta razón pueden existir muchos demostradores de teoremas esencialmente diferentes. Este punto de vista es muy atrayente pero no aporta luz alguna sobre la cuestión planteada de qué hace verdaderas a algunas fórmulas y falsas a otras. Por otra parte, los teoremas son fórmulas que tienen demostraciones, y toda demostración en un sistema lógico puede proporcionar alguna luz. Esto parece sugerir que podemos aprender estudiando las formas que pueden tener las demostraciones. No obstante, la mayoría de los hechos más destacados de las demostraciones están fuertemente influidos tanto por el sistema lógico en el que se da la demostración, como por el teorema que está siendo probado.

Intentamos entender qué características tiene la estructura sintáctica de los teoremas que los hace válidos. En definitiva, si todo sistema de demostración está inspirado por la semántica ¿por qué alejarse de ella? Consideramos que lo más útil es disponer de condiciones suficientes, eficientemente implementables, que aseguren la falsedad o veracidad de una fórmula.

- Un método que trabaja con las fórmulas en general,

es decir, sin los inconvenientes señalados para la resolución, y en consonancia con la característica anterior.

- Un método con un alto grado de paralelismo,

todos los procesos que intervienen en TAS-M**3** son intrínsecamente paralelos.

- Un método de construcción de modelos.

Como expresa R. Hähnle (en relación a su método de Conjunto de valores de verdad como símbolos (Hähnle, 1993)): *retrasamos la decisión final de qué modelo suponemos para $\neg A$ pero reteniendo toda la información.*

La construcción de contraejemplos ha merecido poca atención en el campo de la demostración automática, de hecho los trabajos en esta línea son insignificantes en número respecto a los dedicados a los métodos de refutación o deducción. No obstante, la construcción de modelos es considerada como uno de los hechos más destacados en demostración automática (Bledsoe y Loveland, 1984), (Coferra y Zabel, 1991) y así queda reflejado en trabajos relevantes como (Winker, 1982) y (Woss y Winker, 1984).

Las novedades más destacadas en el marco de la demostración automática de teoremas por TAS-M**3**, puede ser esbozada como sigue:

1. Está fuertemente basado en la estructura de la fórmula, concretamente, en la estructura de su árbol sintáctico. Es un método transformacional, de reescritura; procede realizando transformaciones sucesivas del árbol sintáctico de la fórmula, con el fin de adecuar su estructura al objetivo propuesto (todos los demostradores introducidos incluyen TAS en su nombre, como iniciales de *Transformaciones de Árboles Sintácticos*).

2. Las transformaciones introducidas son de dos tipos:

   - Transformaciones que conservan la insatisfacibilidad de la fórmula, es decir, transformaciones $\mathcal{T}$ tales que si $\mathcal{T}(A) = B$ se tiene que

     $$A \text{ es insatisfacible si y sólo si } B \text{ es insatisfacible}$$

     Estas transformaciones se aplican a la fórmula cuya insatisfacibilidad se desea analizar.

   - Transformaciones que conservan el significado de la fórmula, es decir, transformaciones $\mathcal{T}$ tales que si $\mathcal{T}(A) = B$ se tiene que

     $$A \equiv B$$

     Estas transformaciones se aplican a las subfórmulas de la fórmula cuya insatisfacibilidad se desea analizar.

3. Su potencia no sólo está basada en el diseño intrínsecamente paralelo de las transformaciones que intervienen en él, sino también en el hecho de que estas transformaciones no son aplicadas sin más una tras otra. En efecto, mediante un potente uso de los modelos unitarios, incorporamos criterios implementables eficientemente. Estos criterios permiten detectar subfórmulas válidas, insatisfacibles o equivalentes. De esta forma, se puede determinar si la estructura del árbol sintáctico en curso contiene información directa acerca de la insatisfacibilidad o no de la fórmula. En este caso, el método termina. En caso contrario, esta información nos permite reducir el tamaño del problema antes de aplicar la siguiente transformación. Así la eficiencia del método se ve aumentada.

4. De todas las transformaciones involucradas, únicamente sobre una de ellas, la encargada de las distribuciones, recae el peso de la complejidad exponencial del método. Para mejorar su eficiencia, el método utiliza de nuevo los criterios antes citados para determinar las distribuciones "evitables" y permite la ejecución paralela de las distribuciones "no evitables".

Estas características nos permiten afirmar que, en el ámbito de los demostradores basados en los trabajos de Gentzen, TAS-M**3** constituye a una sustancial mejora debido a que:

- Resuelve totalmente el problema de las redundancias. La solución de este problema, presente en el método de Árboles Finitamente Generados de Carnielli, es un objetivo prioritario en el método de Conjuntos de Valores de Verdad de Hähnle.

- Debido a la novedosa técnica incorporada de detección de la posibilidad de *reducción* de la fórmula analizada, disminuye drásticamente el número de distribuciones respecto al resto de los demostradores basados en los trabajos de Gentzen para la lógica $n$-valuadas.

## Contenidos

El trabajo está organizado como sigue:

En el Capítulo 1 hacemos un rápido recorrido por los elementos básicos utilizados en este trabajo. De esta forma intentamos, por una parte, que el trabajo sea autocontenido y por otra, que las notaciones básicas que utilizamos a lo largo del trabajo hayan sido presentadas en este capítulo. En este primer capítulo, abordamos tanto los requisitos algebraicos básicos como los conceptos elementales de lógica desde el punto de vista de las Ciencias de la Computación. Entre

los primeros cabe destacar, por su uso intensivo en este trabajo, los conceptos de árbol, conjunto inductivo y álgebra abstracta. Entre los segundos, se hace especial hincapié en el papel de la lógica en las Ciencias de la Computación y en la importancia de una eficiente automatización de las deducciones.

En el Capítulo 2 presentamos las lógicas multivaluadas, comenzando por una breve introducción histórica, una descripción de su lenguaje y semántica y una presentación general de la deducción en lógicas multivaluadas. Se termina este capítulo presentando algunos ejemplos y aplicaciones de lógicas multivaluadas.

En el Capítulo 3 recordamos brevemente algunos sistemas de demostración automáticas basados en los trabajos de Gentzen. Además de una presentación general de los sistemas, se describen los métodos de Tablas Semánticas para la Lógica Clásica Proposicional y los métodos de Árboles Finitamente Generados de Carnielli y el de Conjuntos de Valores de Verdad como Signos de Hähnle para la lógica multivaluada.

El núcleo de este trabajo lo constituye la presentación inédita del demostrador automático, TAS-M**3**, para la lógica trivaluada completa M**3**. En primer lugar, se introduce la lógica M**3**. A continuación se describe el método, incluyendo los teoremas que establecen su corrección y completitud. Cada uno de los procesos que intervienen en TAS-M**3** se presenta incluyendo su motivación, su modo de ejecución y su interés de cara a la eficiencia. Aunque cada vez que se introduce un proceso importante de TAS-M**3** se muestra un ejemplo ilustrativo, se ha incorporado una sección con ejemplos completos, que ayuda a la visión global del demostrador.

Finalmente se ha añadido un apéndice con un resumen del demostrador TAS-D para la Lógica Clásica Proposicional. La inclusión de este resumen tiene un doble objetivo, por una parte incluir el demostrador inicial de la metodología, para facilitar la comprensión de TAS-M**3**, y por otra presentar algunas mejoras inéditas realizadas sobre el propio TAS-D, que ha motivado la realización de este trabajo en lógica multivaluada.

## Trabajo futuro

Nuestra tarea actual es la implementación de TAS-M**3**, así como incorporar la mejora *Depurar* y la *generación de tareas* tras *Signar* en las implementaciones del resto de los demostradores TAS desarrollados por nuestro grupo GIMAC.

Como objetivo inmediato (que ya hemos comenzado a abordar), nos proponemos la extensión de TAS-M**3** para las lógicas trivaluadas de primer orden. Para ello disponemos como punto de partida, tanto el presente trabajo como las técnicas TAS para la lógica clásica de primer orden introducidas en (Ojeda, 1996).

Otro objetivo a medio plazo, es comenzar a estudiar la posibilidad de extender la metodología TAS a las lógicas infinito-valuadas, de modo que se mantenga su buen comportamiento en el caso finito. Para ello contamos con la inestimable colaboración del profesor Daniele Mundici, que permite complementar el conocimiento de sus trabajos en demostración automática en lógicas infinito-valuadas (Mundici, 1987a; Mundici, 1995a; Mundici, 1987b; Mundici, 1988b), con enriquecedoras reuniones de trabajo.

Por último, pero no menos importante, nos proponemos incorporar el uso de la lógica trivaluada en aplicaciones que requieran su uso de modo eficiente. Concretamente, en aplicaciones que requieren disponer de un mecanismo deductivo para razonar en contextos temporales específicos, con información incompleta y que posibilite la actualización en tiempo real de las bases de conocimiento; aspectos irrenunciables si se pretende aplicarlos a problemas concretos. Este objetivo es prioritario en la reciente propuesta, en la que nuestro grupo es coordinador, del proyecto TMR (Training and Mobility of Researchers) 1994-1998: *The Craft of Formal Modelling: A Cooperative Systems for Technical Applications.* Acrónimo: COSTA.

Este último objetivo requiere la integración de los contextos temporales y multivaluados en la metodología TAS con una meta final:

> *Poner el razonamiento temporal-multivaluado (automatizado) al alcance de usuarios ajenos al formalismo de la lógica.*

# Capítulo 1

# Preliminares

## 1.1. Prerrequisitos Algebraicos

Con la intención de hacer el trabajo autocontenido en la medida de lo posible, se introducen en esta sección los conceptos básicos utilizados a lo largo del mismo, entre ellos, los de cadena, árbol, conjunto inductivo y álgebra abstracta que serán muy utilizados. No se incluyen demostraciones en esta sección, ya que los resultados expuestos son ampliamente conocidos.

### 1.1.1. Relaciones

La noción de relación formaliza el concepto utilizado en la vida real de "relación entre objetos".

Dados dos conjuntos $A$ y $B$, una *relación* de $A$ en $B$ es un subconjunto arbitrario $R$ (posiblemente vacío) de $A \times B$.

Dada una relación $R \subseteq A \times B$, habitualmente $(x, y) \in R$ se representa mediante $xRy$ y se lee "$x$ está relacionado con $y$".

En el caso particular de que $R \subseteq A \times A$ se dice que $R$ es una relación binaria sobre $A$. La relación identidad en $A$, $Id_A = \{(a, a) \mid a \in A\}$ es una relación binaria en $A$.

Damos a continuación definiciones de relaciones obtenidas a partir de otras relaciones:

- Si $R$ y $S$ son relaciones de $A$ en $B$ entonces $R \cup S$ y $R \cap S$ son relaciones de $A$ en $B$ llamadas *unión* e *intersección* de $R$ y $S$, respectivamente.
- Sea $R$ una relación de $A$ en $B$ y $S$ una relación de $B$ en $C$, la relación de $A$ en $C$ denotada $R \circ S$ y definida por

$$(a, c) \in R \circ S \text{ sii existe un } b \in B \text{ tal que } (a, b) \in R \text{ y } (b, c) \in S$$

se denomina *composición* de $R$ y $S$.

- Si $R$ es una relación de $A$ en $B$ entonces $R^{-1}$ es la relación de $B$ en $A$ definida por
$$aR^{-1}b \text{ sii } bRa$$
se llama *inversa* de $R$.

Si $R$ es una relación binaria sobre $A$, $x$ un elemento de $A$ y $V$ un subconjunto de $A$, utilizamos la siguientes notaciones:

- $R(x) = \{y \in A \mid xRy\}$
- $R(V) = \{y \in A \mid \text{ existe } x \in V \text{ con } xRy\}$
- $R^2(V) = R(R(V))$
- En general: $R^n(V) = R(R^{n-1}(V))$ con $n \in \mathbb{N}$ y $n > 1$.

A continuación destacamos una serie de propiedades que puede verificar una relación binaria $R$ sobre un conjunto $A$ y en el teorema 1.1 se dan caracterizaciones de ellas.

i) $R$ es *reflexiva* si y sólo si $xRx$ para todo $x \in A$.

ii) $R$ es *transitiva* si y sólo si supuesto que $xRy$ e $yRz$ se tiene que $xRz$.

iii) $R$ es *simétrica* si y sólo si supuesto que $xRy$ se tiene que $yRx$.

iv) $R$ es *antisimétrica* si y sólo si supuesto que $xRy$ e $yRx$ se tiene que $x = y$.

v) $R$ es un *preorden* si y sólo si es reflexiva y transitiva.

vi) $R$ es un *orden parcial* si y sólo si es un preorden y es antisimétrica.

vii) $R$ es un *orden lineal* si y sólo si es un orden parcial y para todo $x \in A$ y para todo $y \in A$ se verifica que $xRy$ o $yRx$. es conexa.

viii) $R$ es de *equivalencia* si y sólo si es un preorden y es simétrica.

**Teorema 1.1** *Sea $R$ una relación binaria sobre un conjunto $A$ y $V$ un subconjunto de $A$, entonces:*

1. *$R$ es reflexiva si y sólo si $V \subseteq R(V)$ para todo $V \subseteq A$, o equivalentemente, si y sólo si $Id_A \subseteq R$.*
2. *$R$ es transitiva si y sólo si $R^2(V) \subseteq R(V)$ para todo $V \subseteq A$, o equivalentemente, si y sólo si $R^2 \subseteq R$.*

3. *$R$ es simétrica si y sólo si $x \in \bigcap_{y \in R(x)} R(y)$, o equivalentemente, si y sólo si $R^{-1} \subseteq R$.*

4. *$R$ es antisimétrica si y sólo si $x \notin R(R(x) - \{x\})$ para todo $x \in A$.*

5. *$R$ es antisimétrica si y sólo si $R \cap R^{-1} \subseteq Id_A$.*

6. *$R$ es un preorden si y sólo si $R^2(V) = R(V)$ para todo $V \subseteq A$*

7. *$R$ es de equivalencia si y sólo si $\{R(x) \mid x \in A\}$ es una partición de $A$.*

### 1.1.2. Alfabeto y Cadenas

Sea $C$ un conjunto no vacío al que llamamos *alfabeto*. Una *cadena* sobre el alfabeto $C$ es toda secuencia finita $\alpha : \{1, 2, \ldots, n\} \to C$. Al natural $n$ se le llama *longitud* de $\alpha$ y se denota $long(\alpha)$. Las cadenas (también llamadas *palabras*) se denotan por la yuxtaposición de las imágenes de $\alpha$, es decir, si $n = long(\alpha), \alpha(i) = c_i$ e $i \in \{1, 2, \ldots n\}$, entonces $\alpha = c_1 c_2 \ldots c_n$. La *cadena vacía* o *cadena nula*, denotada $\epsilon$, es la función $\epsilon : \varnothing \to C$ y se tiene que $long(\epsilon) = 0$.

Para todo natural $n$, sea $C^n$ el conjunto de las cadenas de longitud $n$ sobre $C$. La unión de la familia $\{C^n\}_{n \in \mathbb{N}}$ se llama el *lenguaje universal* sobre $C$ y se denota por $C^*$, es decir:

$$C^* = \bigcup_{n \in \mathbb{N}} C^n$$

Dadas dos cadenas $\alpha : \{1, 2, \ldots, n\} \to C$ y $\beta : \{1, 2, \ldots, m\} \to C$, se define la *concatenación* de $\alpha$ y $\beta$, denotada $\alpha\beta$, como la cadena $\gamma : \{1, 2, \ldots, n + m\} \to C$ definida por

$$\gamma(i) = \begin{cases} \alpha(i) & \text{si } 1 \leq i \leq n \\ \beta(i - n) & \text{si } n + 1 \leq i \leq n + m \end{cases}$$

La concatenación es una *operación interna* sobre el lenguaje universal, esta operación es asociativa, no es conmutativa en el caso general y la cadena vacía $\epsilon$ es el elemento neutro para dicha operación, es decir, el lenguaje universal $C^*$ con la operación de concatenación tiene estructura de monoide, que será no abeliano salvo en el caso de que $C$ sea unitario.

Dada una cadena $\alpha$, una cadena $\beta$ se dice que es un *prefijo* de $\alpha$, si existe una cadena $\gamma$ tal que $\alpha = \beta\gamma$. Una cadena $\beta$ es un *sufijo* de $\alpha$, si existe una cadena $\gamma$ tal que $\alpha = \gamma\beta$. Una cadena $\beta$ es una *subcadena* de $\alpha$, si existen cadenas $\gamma$ y $\delta$ tales que $\alpha = \gamma\beta\delta$. Un prefijo, sufijo o subcadena $\beta$ de una cadena $\alpha$ se dice *propia* si $\beta \neq \alpha$.

### 1.1.3. Árboles

En esta sección introducimos la noción de árbol debida a Gorn (Gallier, 1987; Huet y et al, 1986) y las nociones básicas sobre árboles. La estructura de árbol es fundamental en la metodología TAS, ya que las fórmulas se representan mediante sus árboles sintácticos, y el algoritmo se describe en términos de transformaciones de árboles sintácticos. [1]

Fijado un conjunto numerable $T$ y una enumeración $\{t_1, t_2, \ldots, t_n, \ldots\}$ del mismo, un *dominio de árbol*, $D$, es un subconjunto de $T^*$ que verifica las siguientes condiciones:

1. Para todo $\alpha \in D$, todo prefijo de $\alpha$ pertenece a $D$.

2. Para todo $\alpha \in D$ tal que $\alpha t_i \in D$, se tiene que $\alpha t_j \in D$ para todo $j$ tal que $1 \leq j < i$.

Obsérvese que por la primera condición, para todo dominio de árbol $D$ no vacío se tiene que $\epsilon \in D$, lo que dará lugar a que todo árbol no vacío tenga raíz. Otras definiciones de árbol, no exigen que éste sea con raíz; por ejemplo, en la teoría de grafos se considera que un árbol es todo grafo no dirigido conexo sin ciclos. Nosotros consideraremos que cualquier árbol no vacío tiene raíz.

Sea $\Sigma$ un conjunto no vacío a cuyos elementos llamamos *etiquetas*. Un $\Sigma$-*árbol* (brevemente un *árbol*) es una función $t : D \to \Sigma$ donde $D$ es un dominio de árbol. A las cadenas $\alpha \in \mathrm{Dom}(t)$ se las acostumbra a llamar *nodos* o *direcciones* del árbol $t$. Si $D = \varnothing$, la función $t$ se denomina *árbol vacío*. Para todo árbol $t$ no vacío, el nodo $\epsilon$ se llama *raíz* de $t$.

El *grado de ramificación* de un nodo $\alpha$ es el cardinal de $\{t_i \in T \mid \alpha t_i \in \mathrm{Dom}(t)\}$ y se denota por $d(\alpha)$. Los nodos $\alpha$ tales que $d(\alpha) = 0$ se llaman *hojas*.

Un árbol $t$ se dice de *ramificación finita* si para todo nodo $\alpha$, $d(\alpha)$ es finito. Un árbol se dice *binario* si para todo nodo $\alpha$ se tiene que $d(\alpha) \leq 2$ y se dice además que es *completo* si para todo nodo no hoja $\alpha$ se tiene que $d(\alpha) = 2$.

Un árbol $t$ se dice *finito* si $\mathrm{Dom}(t)$ es finito y se dice *infinito* en caso contrario. Obviamente todo árbol finito es de ramificación finita. Los árboles de un solo nodo se llaman *árboles hoja*.

Los nodos de la forma $\alpha t_i$, con $t_i \in T$, se llaman *descendientes inmediatos* (o *hijos*) de $\alpha$, y al nodo $\alpha$ *ascendiente inmediato* (o *padre*) de cualquiera de sus descendientes inmediatos $\alpha t_i$. Si el árbol es binario y $d(u) = 2$, los descendientes inmediatos de $\alpha$ se denominan *descendiente izquierdo* y *descendiente derecho*.

Cuando se trabaja con $\Sigma$-árboles para un $\Sigma$ determinado, es costumbre referirse a los nodos por sus etiquetas, y así utilizar expresiones como:

[1] TAS son las iniciales de Transformaciones de Árboles Sintácticos.

"El nodo $a$", cuando debería decirse "el nodo etiquetado con $a$".

"El árbol o árboles cuyos nodos son elementos de $\Sigma$", cuando debería decirse "El árbol o árboles cuyos nodos están etiquetados con elementos de $\Sigma$"

### Caminos

Sea $t$ un árbol y $\alpha, \beta \in \mathrm{Dom}(t)$. Un *camino* desde $\alpha$ hasta $\beta$ es una secuencia finita de nodos $\langle \alpha_1, \alpha_2, \ldots, \alpha_{n+1} \rangle$ tal que

- $\alpha = \alpha_1$ y $\beta = \alpha_{n+1}$.
- Para todo $j$ tal que $2 \leq j \leq n$ se tiene que $\alpha_j = \alpha_{j-1} t_{k_j}$ para algún $t_{k_j} \in T$.

El natural $n$ se llama *longitud* del camino.

En árboles infinitos, cabe considerar caminos infinitos. Un *camino infinito* desde el nodo $\alpha$ es una sucesión de nodos $\{\alpha_1, \alpha_2, \ldots, \alpha_n, \ldots\}$ tal que

- $\alpha = \alpha_1$.
- Para todo $j > 1$ se tiene que $\alpha_j = \alpha_{j-1} t_{k_j}$ para algún $t_{k_j} \in T$

Una *rama* es un camino infinito desde la raíz o un camino desde la raíz a una hoja. Así, las ramas se clasifican en finitas e infinitas. Obviamente, en un árbol finito sólo existen ramas finitas y su número coincide con el número de hojas.

Dado un árbol $t$, para todo nodo $\alpha$ de $t$ existe un único camino desde la raíz al nodo $\alpha$. La longitud de dicho camino se llama la *profundidad o el nivel de* $\alpha$. La noción de profundidad de un nodo estratifica los nodos de un árbol en: nodos de profundidad 0 (la raíz), profundidad 1, profundidad 2, etc.

Se define la *profundidad* o *altura* de un árbol $t$ como

$$Prof(t) = \begin{cases} \text{máx}\{\text{nivel}(\alpha) \mid \alpha \in \mathrm{Dom}(t)\} \text{ si } t \text{ no tiene ramas infinitas} \\ \infty\,, \text{ en otro caso} \end{cases}$$

El siguiente resultado establece una condición suficiente para la existencia de ramas infinitas.

**Lema 1.1 (König)** *En todo árbol infinito de ramificación finita existe al menos una rama infinita.*

### Subárboles

Sea $t$ un árbol y $\alpha$ un nodo de $t$. El *subárbol de raíz* $\alpha$ es el árbol $t_\alpha$ definido como sigue:

- $\mathrm{Dom}(t_\alpha) = \{\beta \mid \alpha\beta \in \mathrm{Dom}(t)\}$.
- $t_\alpha(\beta) = t(\alpha\beta)$.

Para todo árbol $t$ y para todo nodo no hoja $\alpha$ de $t$, los subárboles cuya raíz son los descendientes inmediatos de $\alpha$ se llaman *subárboles de* $\alpha$. En particular, si $d(u) = 2$ se llaman *subárbol izquierdo* y *subárbol derecho.*

Es inmediato que si $t$ es de ramificación finita, todos sus subárboles son también de ramificación finita. Si además $t$ es infinito, por el Lema de König, $t$ tiene subárboles infinitos.

### Recorridos de un árbol

Describimos a continuación dos modos estándar de ordenar, enumerar o recorrer los nodos de un $\Sigma$-árbol finito $t$. Dichas enumeraciones proporcionan una secuencia de sus nodos, contemplados como elementos de $\Sigma$.

*Primero en Profundidad*: En todo árbol finito $t$, la relación $\leq$ en $\mathrm{Dom}(t)$ definida a continuación es de orden total. Diremos que $\alpha \leq \beta$ si se cumple alguna de las condiciones siguientes:

1. $\alpha$ es prefijo de $\beta$
2. Existen cadenas $\gamma, \delta, \lambda \in T^*$ y elementos $t_i, t_j \in T$ con $i < j$ tales que $\alpha = \gamma t_i \delta$ y $\beta = \gamma t_j \lambda$

La enumeración de los nodos de $t$ proporcionada por dicha relación se llama *recorrido primero en profundidad* de $t$. Si $\alpha \leq \beta$, por la primera condición, se dice que $\alpha$ es un *ascendiente* de $\beta$ y que $\beta$ es un *descendiente* de $\alpha$.

*Primero en Anchura:* En todo árbol finito $t$, la relación $\leq$ en $\mathrm{Dom}(t)$ definida a continuación es de orden total. Diremos que $\alpha \leq \beta$ si se cumple alguna de las condiciones siguientes:

1. $\mathrm{long}(\alpha) < \mathrm{long}(\beta)$
2. Existen cadenas $\gamma, \delta, \lambda \in T^*$ con $\mathrm{long}(\delta) = \mathrm{long}(\lambda)$ y elementos $t_i, t_j \in T$ con $i < j$ tales que $\alpha = \gamma t_i \delta$ y $\beta = \gamma t_j \lambda$

La enumeración de los nodos de $t$ proporcionada por dicha relación se llama *recorrido primero en anchura de* $t$.

### 1.1.4. Conjuntos Inductivos

En diferentes áreas de las Ciencias de la Computación es frecuente introducir objetos mediante la definición inductiva de conjuntos. Se entiende por ello la definición de un conjunto caracterizado por ser:

- El menor conjunto que contiene a un conjunto dado $X$ llamado *conjunto base* o *conjunto de átomos*.
- Cerrado para un conjunto $F$ de funciones llamado *conjunto de constructores*.

Sea $A$ un conjunto no vacío, $X$ un subconjunto no vacío de $A$ y $F$ una familia de funciones sobre $A$ (cada elemento $f$ de $F$ tiene su propia aridad $\alpha(f)$), es decir,

$$F = \{f_i : A^{\alpha(f_i)} \to A \mid (i, \alpha(f_i)) \in I \times J\}$$

donde $I$ es un conjunto arbitrario no vacío y $J$ un subconjunto no vacío de $\mathbb{N}^*$.

Un subconjunto $Y$ de $A$ se dice *inductivo sobre $X$ para $F$*, si verifica:

1. $X \subseteq Y$
2. $Y$ es *cerrado* para $F$, es decir, para toda $f : A^{\alpha(f)} \to A$ de $F$ y para todo $(y_1, \ldots, y_{\alpha(f)}) \in Y^{\alpha(f)}$, se tiene que $f(y_1, \ldots, y_{\alpha(f)}) \in Y$.

La intersección de todos los conjuntos inductivos sobre $X$ para $F$, al que denotaremos por $X^+$, es también un conjunto inductivo sobre $X$ para $F$ y se llama *la clausura inductiva* de $X$ para $F$. Como $A$ es, obviamente, inductivo sobre $X$ para $F$, podemos asegurar que $X^+ \neq \varnothing$.

A continuación presentamos una descripción constructiva de $X^+$:

Sea $(X_i)$ la sucesión creciente de conjuntos definida como sigue:

$$\begin{cases} X_1 & = & X \\ X_{i+1} & = & X_i \cup \{f(x_1, \ldots, x_{\alpha(f)}) \mid f \in F \quad \text{y} \quad (x_1, \ldots, x_{\alpha(f)}) \in X_i^{\alpha(f)}\} \end{cases}$$

Entonces,

$$X^+ = \bigcup_{i \geq 1} X_i$$

Advirtamos que cada elemento de $x \in X^+$, puede ser generado por más de una tupla $(f, x_1, \ldots, x_n)$ donde $f \in F$ y $n = \alpha(f)$. Análogamente, cada elemento de $x \in X^+$ puede representarse por un árbol o un conjunto de árboles; específicamente:

- Cada elemento $x$ del conjunto base, por el árbol de un solo nodo etiquetado con $x$.

- Si $x \in X_i$ y $f(x_1, \ldots, x_{\alpha(f)}) \in X_i$ es una representación más de $x$, el árbol:

  (i) su raíz está etiquetada con $f$ y su grado de ramificación es $\alpha(f)$.

  (ii) cada descendiente inmediato de la raíz es la raíz de un árbol que representa a $x_j$.

Ahora tenemos los elementos necesarios para advertir que el principio de inducción no es más que la lectura para $\mathbb{N}$ del siguiente resultado.

**Teorema 1.2 (Principio de Inducción Estructural)** *Sea $X^+$ la clausura inductiva de $X$ para $F$. Sea $\mathcal{P}$ una propiedad relativa a $X^+$ tal que:*

1. *Todo elemento de $X$ verifica $\mathcal{P}$.*
2. *$F$ respeta la propiedad $\mathcal{P}$, es decir, para todo $f \in F$ de aridad $\alpha(f) = n$:*
   *Si $x_1, \ldots, x_n \in X^+$ verifican $\mathcal{P}$, entonces $f(x_1, \ldots, x_n)$ verifica $\mathcal{P}$.*

*En tales condiciones, todos los elementos de $X^+$ verifican $\mathcal{P}$.*

### Clausuras inductivas libremente generadas

Un tipo especial de clausura inductiva de particular interés por su adecuación para ser manipulada, lo forman las clausuras inductivas *libremente generadas*, que son aquéllas en las que los elementos de $X^+$ son generados de forma única a partir del conjunto base $X$ y del conjunto de constructores $F$, es decir, aquellas en las que todo elemento $a$ de $X^+$ es representado por un *único árbol*. Más formalmente:

Sea $A$ un conjunto, $X$ un subconjunto de $A$, $F$ un conjunto de funciones sobre $A$ y $X^+$ la clausura inductiva de $X$ para $F$. Se dice que $X^+$ es *libremente generada* si cumple las siguientes condiciones:

1. Para toda $f : A^{\alpha(f)} \to A$ de $F$, su restricción a $X^+$ es inyectiva, es decir, todo constructor genera elementos distintos desde elementos distintos.

2. Para todo par de elementos $f$ y $g$ de $F$ con $f : A^{\alpha(f)} \to A$ y $g : A^{\alpha(g)} \to A$ y $f \neq g$, las imágenes $f((X^+)^{\alpha(f)})$ y $g((X^+)^{\alpha(g)})$ son disjuntas, es decir, constructores distintos generan elementos distintos.

3. Para toda $f : A^{\alpha(f)} \to A$ de $F$ y para todo $(x_1, \ldots, x_{\alpha(f)})$ de $(X^+)^{\alpha(f)}$, se tiene que $f(x_1, \ldots, x_{\alpha(f)}) \notin X$, es decir, ningún constructor genera elementos del conjunto base.

Como consecuencia de la definición, una clausura inductiva libremente generada tiene cardinal infinito.

En las clausuras inductivas libremente generadas, los conjuntos $X_i$ definidos en la página 31 satisfacen la siguiente propiedad para cualquier función $f \in F$:

$$\textit{Si } (x_1, \ldots, x_{\alpha(f)}) \in X_i^{\alpha(f)} - X_{i-1}^{\alpha(f)}, \textit{ entonces } f(x_1, \ldots, x_{\alpha(f)}) \notin X_i.$$

Por lo tanto, $X^+$ es la unión de la sucesión estrictamente creciente de conjuntos $Y_i$ definidos como sigue:

$$\begin{array}{rcl} Y_1 & = & X \\ Y_2 & = & Y_1 \cup \{f(y_1, \ldots, y_{\alpha(f)}) \mid f \in F \text{ e } (y_1, \ldots, y_{\alpha(f)}) \in Y_1^{\alpha(f)}\} \\ \ldots & \ldots & \\ Y_{i+1} & = & Y_i \cup \{f(y_1, \ldots, y_{\alpha(f)}) \mid f \in F \text{ e } (y_1, \ldots, y_{\alpha(f)}) \in Y_i^{\alpha(f)} - Y_{i-1}^{\alpha(f)}\} \end{array}$$

**Funciones Recursivas**

Bajo ciertas condiciones, es posible extender de forma única una función definida sobre el conjunto base $X$ a su clausura inductiva libremente generada $X^+$.

Sean:

- Un conjunto $A$, $X \subseteq A$, un conjunto $F$ de funciones sobre $A$ y la clausura inductiva de $X$ para $F$, $X^+$.
- Un conjunto $B$ y un conjunto $G$ de funciones sobre $B$.
- Una función $\phi : F \to G$ tal que para todo $f \in F$, $\phi(f)$ tiene la misma aridad que $f$.

En tales condiciones, si $X^+$ es libremente generada, entonces para toda función $h : X \to B$ existe una única función $\widehat{h} : X^+ \to B$ tal que:

1. $\widehat{h}$ es una extensión de $h$, es decir, $\widehat{h}(x) = h(x)$ para todo $x \in X$.
2. $\widehat{h}(f(x_1, \ldots, x_{\alpha(f)})) = \phi(f)(\widehat{h}(x_1), \ldots, \widehat{h}(x_{\alpha(f)}))$ para toda función $f$ de $F$ y para todo $(x_1, \ldots, x_{\alpha(f)}) \in (X^+)^{\alpha(f)}$.

La segunda condición puede expresarse diciendo que el siguiente diagrama

es conmutativo, es decir, $\widehat{h} \circ f = \phi(f) \circ \widehat{h}^{\alpha(f)}$, siendo $\widehat{h}^{\alpha(f)}$ la función definida por: $\widehat{h}^{\alpha(f)}(x_1, \ldots, x_{\alpha(f)}) = (\widehat{h}(x_1), \ldots, \widehat{h}(x_{\alpha(f)}))$

La extensión $\widehat{h}$ definida se dice que es una *función recursiva* o que está *definida inductivamente*. Para ella, conocido el valor de $\widehat{h}$ sobre los elementos del conjunto base, $\widehat{h}$ se define sobre un elemento $a$ de $X^+$ en términos de sí misma sobre elementos de $\widehat{h}$ más simples que $a$ ($\widehat{h}$ "se llama a sí misma").

### 1.1.5. Álgebras Abstractas

Con frecuencia, en la bibliografía sobre Lógica, encontramos el uso de álgebras abstractas para describir la sintaxis y la semántica de diversas lógicas como alternativa al uso de conjuntos inductivos libremente generados. Este hecho se debe a que, por una parte, unifica la notación y, por otra, permite un tratamiento más abstracto del tema. Siguiendo a R. Hähnle (Hähnle, 1993), el uso que nosotros haremos de este concepto estará motivado por la primera causa.

Sea $A$ un conjunto no vacío y $F$ una familia de funciones sobre $A$ (cada elemento $f$ de $F$ con aridad $\alpha(f)$), es decir:

$$F = \{f_i : A^{\alpha(f_i)} \to A \mid (i, \alpha(f_i)) \in I \times J\}$$

donde $I$ es un conjunto arbitrario de índices no vacío y $J$ un subconjunto no vacío de $\mathbb{N}^*$.

Al par $\mathcal{A} = (A, F)$ se le denomina *álgebra abstracta* sobre $A$ o simplemente *álgebra*. Al conjunto $A$ se le llama el *universo* de $\mathcal{A}$. Si $|A| = 1$, diremos que el álgebra es *degenerada*.

Cuando $I$ es finito con $|I| = n$ se suele utilizar $\{1, 2, \ldots, n\}$ para el conjunto $I$ y en vez de escribir $\mathcal{A} = (A, \{f_1, f_2, \ldots, f_n\})$ se abrevia la notación con $\mathcal{A} = (A, f_1, f_2, \ldots, f_n)$.

A la secuencia formada por las distintas aridades de las funciones de $F$ se le denomina *tipo de similaridad* de $\mathcal{A}$, así diremos que el álgebra $\mathcal{A} = (A, f_1, f_2, \ldots, f_n)$ tiene un tipo de similaridad $\langle \alpha(f_1), \alpha(f_2), \ldots, \alpha(f_n) \rangle$. Dos álgebras se dicen *similares* si tienen el mismo tipo de similaridad.

Si $\mathcal{A} = (A, F)$ es un álgebra y $F' \subsetneq F$, diremos que el álgebra $\mathcal{A}' = (A, F')$ es una reducida de $\mathcal{A}$.

Un álgebra abstracta $\mathcal{B} = (B, F)$ es una *subálgebra* del álgebra abstracta $\mathcal{A} = (A, F)$ si $B$ es un subconjunto no vacío de $A$ que es cerrado respecto a las operaciones $f \in F$. Obviamente, todas las subálgebras de un álgebra $\mathcal{A}$ son similares, y la intersección de todas las subálgebras de un álgebra $\mathcal{A}$ es una subálgebra de $\mathcal{A}$.

Si $\mathcal{A} = (A, F)$ es un álgebra y $G$ un conjunto no vacío de $A$, la intersección de todas las subálgebras de $\mathcal{A}$ cuyos universos contienen a $G$ es también una subálgebra de $\mathcal{A}$, de hecho es la menor (en el sentido de la inclusión conjuntista de sus universos) subálgebra de $\mathcal{A}$ cuyo universo contiene a $G$. Sea $\mathcal{A}^G$ esta subálgebra, en este caso se dice que $\mathcal{A}^G$ está *generada* por $G$ y llamamos a $G$ el *conjunto de generadores* de $\mathcal{B}$.

Se dice que $h$ es un *homomorfismo* del álgebra $\mathcal{A} = (A, f_1^{\mathcal{A}}, f_2^{\mathcal{A}}, \ldots, f_n^{\mathcal{A}})$ en el álgebra $\mathcal{B} = (B, f_1^{\mathcal{B}}, f_2^{\mathcal{B}}, \ldots, f_n^{\mathcal{B}})$ si y sólo si se cumplen las siguientes condiciones:

- $\mathcal{A}$ y $\mathcal{B}$ son álgebras similares.

- $h : A \to B$ es una función.
- Para cualquier $i \in \{1, \ldots, n\}$ y para cualquier $(a_1, a_2, \ldots, a_{\alpha(f_i^{\mathcal{A}})}) \in A^{\alpha(f_i^{\mathcal{A}})}$ se verifica que:

$$h(f_i^{\mathcal{A}}(a_1, a_2, \ldots, a_{\alpha(f_i^{\mathcal{A}})})) = f_i^{\mathcal{B}}(h(a_1), h(a_2), \ldots, h(a_{\alpha(f_i^{\mathcal{A}})}))$$

en definitiva, si son conmutativos los diagramas siguientes para todo $i$

es decir, $h \circ f_i^{\mathcal{A}} = f_i^{\mathcal{B}} \circ h^{\alpha(f_i^{\mathcal{A}})}$, siendo $h^{\alpha(f_i^{\mathcal{A}})}$ la función definida por:

$$h^{\alpha(f_i^{\mathcal{A}})}(a_1, \ldots, a_{\alpha(f_i^{\mathcal{A}})}) = (h(a_1), \ldots, h(a_{\alpha(f_i^{\mathcal{A}})}))$$

Sea $\mathcal{A}$ un álgebra y $\mathfrak{A}$ la clase de todas las álgebras similares a $\mathcal{A}$. Se dice que $\mathcal{A}$ es *libre* en $\mathfrak{A}$ si y sólo si existe un conjunto de generadores $G$ de $\mathcal{A}$, tal que para cualquier álgebra $\mathcal{B}$ en $\mathfrak{A}$, toda función $f : G \to B$ puede extenderse a un homomorfismo de $\mathcal{A}$ en $\mathcal{B}$. En este caso se dice que $\mathcal{A}$ es *libremente generada* por $G$ en $\mathfrak{A}$ y que $G$ es un *conjunto de generadores libres* de $\mathcal{A}$ en $\mathfrak{A}$.

**Teorema 1.3** *Si $\mathcal{A}$ es un álgebra libremente generada por $G$ en la clase $\mathfrak{A}$ de álgebras similares, entonces para cualquier álgebra $\mathcal{B}$ en $\mathfrak{A}$, toda función $f : G \to B$ puede extenderse* **de forma única** *a un homomorfismo $\widehat{f}$ de $\mathcal{A}$ en $\mathcal{B}$.*

**Matrices Locales**

**Definición 1.1** Sea $\mathcal{A} = (A, F)$ un álgebra abstracta. Se llama *matriz* a toda terna de la forma $(A, A^*, F)$, con $A^*$ un subconjunto no vacío de $A$.

### 1.1.6. Álgebras Casi-booleanas

**Definición 1.2** Un retículo es un álgebra abstracta $(A, \vee, \wedge)$ con tipo de similaridad $\langle 2, 2\rangle$ y tal que $\vee$ y $\wedge$ satisfacen las propiedades conmutativas, asociativas y de absorción.

Si además se satisfacen las propiedades distributivas:

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c); \qquad a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

diremos que el retículo es distributivo.

Los elementos $a \vee b$ y $a \wedge b$ se denominan, respectivamente, la suma y el producto de $a$ y $b$

Si $(A, \vee, \wedge)$ es un retículo, para todo par de elementos $a, b \in A$ se tiene que:

$$a \vee b = b \quad \text{si y sólo si} \quad a \wedge b = a.$$

Esta propiedad permite definir la relación

$$a \leq b \quad \text{si y sólo si} \quad a \vee b = b.$$

la cual es reflexiva, antisimétrica y transitiva, es decir, $(A, \leq)$ es un conjunto ordenado. Obviamente,

$$a \vee b = \sup\{a, b\}; \qquad a \wedge b = \inf\{a, b\}$$

supremo e ínfimo respecto a $\leq$. Por tanto, tenemos

- $a \vee a = a$; $a \wedge a = a$;
- $a \leq a \vee b$; $a \wedge b \leq a$;
- $b \leq a \vee b$; $a \wedge b \leq b$;
- Si $a \leq c$ y $b \leq c$ entonces $a \vee b \leq c$;
- Si $c \leq a$ y $c \leq b$ entonces $c \leq a \wedge b$;
- Si $a \leq c$ y $b \leq d$ entonces $a \vee b \leq c \vee d$ y $a \wedge b \leq c \wedge d$;

Sea $\mathcal{A} = (A, \vee, \wedge)$ y $\leq$ un orden asociado; si existen en $A$ un elemento mínimo, 0, y un elemento máximo, 1, se tiene que para todo elemento $a \in A$:

- $0 \wedge a = 0$, $0 \vee a = a$
- $1 \wedge a = a$, $1 \vee a = 1$

**Definición 1.3** Un álgebra $\mathcal{A} = (A, \vee, \wedge, ^-, 0, 1)$ con tipo de similaridad $\langle 2, 2, 1, 0, 0\rangle$ se dice *casi-booleana* si el álgebra reducida $(A, \vee, \wedge)$ es un retículo distributivo con unidad 1 y con cero 0 y además la operación monaria $^-$ satisface las propiedades:

- $\overline{\overline{a}} = a$
- $\overline{a \vee b} = \overline{a} \wedge \overline{b}$

De la definición se tiene de modo inmediato que:

- $\overline{0} = 1$, $\overline{1} = 0$
- $\overline{a \wedge b} = \overline{a} \vee \overline{b}$

**Definición 1.4** Un álgebra $\mathcal{A} = (A, \vee, \wedge, ^- , 0, 1)$ con tipo de similaridad $\langle 2, 2, 1, 0, 0\rangle$ es un *álgebra de Boole* si es casi-booleana y además la operación monaria $^-$ satisface las propiedades:

- $a \wedge \overline{a} = 0$
- $a \vee \overline{a} = 1$

## 1.2. Lógica: el Análisis de los Razonamientos

El hecho de que un ordenador pueda efectuar ciertos "razonamientos" causa en algunas personas un desazonado estupor, quizás atribuible a la concepción del ser humano como el único ser capaz de razonar. Sin embargo es conocido de antiguo que ciertos esquemas de lo que consideramos un razonamiento válido, están determinados por leyes que ciertos razonamientos válidos están sujetos a esquemas y gobernados por leyes que pueden ser descritas en términos de la "forma" y no del contenido de los razonamientos, es decir, la validez de un razonamiento puede ser analizada sintácticamente y, por tanto, este análisis puede realizarse mediante un ordenador.

Es precisamente en el campo de las Ciencias de la Computación donde la Lógica ha pasado a ser una herramienta básica en aspectos tan diversos como análisis, síntesis y verificación de programas, programación lógica, inteligencia artificial, control de procesos, robótica, ... Se dice de hecho que: "La Lógica es el Cálculo para las Ciencias de la Computación" (Manna y Waldinger, 1985; Woodcock y Loomes, 1991).

## 1.3. La Lógica en las Ciencias de la Computación

Nuestro grupo de investigación está especialmente interesado en las aplicaciones computacionales de las distintas lógicas, por ello, siguiendo a Martín-Löf (Martin-Löf, 1987), distinguimos en una lógica los cuatro aspectos siguientes:

- El lenguaje.
- La semántica.
- Una teoría de la demostración.
- La automatización de las deducciones.

Detallaremos a continuación cada uno de estos aspectos, expresándolos en un contexto general, para particularizarlos a las lógicas multivaluadas más adelante.

### 1.3.1. El Lenguaje

La descripción del lenguaje que utilizaremos en cada lógica empieza por determinar el conjunto de caracteres básicos o alfabeto. Estos caracteres se utilizan para construir las cadenas o palabras que permitirán describir el conocimiento sobre el dominio. Formalmente, llamaremos *alfabeto* a un conjunto numerable de símbolos $\nabla$. Un *lenguaje* $L$ sobre $\nabla$ es un subconjunto no vacío del *lenguaje universal* sobre $\nabla$, es decir:

$$L \subseteq \nabla^* = \bigcup_{n \in \mathbb{N}} \nabla^n$$

Llamaremos a los elementos de $L$ *fórmulas bien formadas* (en adelante, *fbf*). Un lenguaje $L$ puede, por tanto, determinarse por:

(i) Un conjunto de símbolos, llamado *alfabeto* del lenguaje.

(ii) Un conjunto de *reglas de formación* que determinan qué cadenas de símbolos del alfabeto son fbfs del lenguaje. Estas reglas de formación constituyen la *gramática* o *sintaxis* del lenguaje.

Si ambos conjuntos están definidos sin hacer referencia alguna al significado de los símbolos, el lenguaje se denomina *lenguaje formal*.

A un lenguaje formal se le denomina *lenguaje objeto* y al lenguaje utilizado para describirlo se le denomina *metalenguaje*.

En términos de clausuras inductivas (libremente generadas): dado un alfabeto $\nabla$, un conjunto no vacío $X$ de *átomos*, donde $X \subset \nabla^*$, y un conjunto de operadores o conectivos $F = \{op_i : X^{\alpha(op_i)} \to X\}$, llamamos lenguaje $L$ a la clausura inductiva (libremente generada) del conjunto $X$ para el conjunto de operadores $F$, es decir $L = X^+$.

En términos de álgebras abstractas, decimos que el álgebra abstracta $\mathcal{L} = (L, F)$ es un lenguaje si es libremente generada para un cierto subconjunto $X$ de $L$ a cuyos elementos llamamos *átomos*.

### 1.3.2. La Semántica

Para introducir el concepto de "verdad" debemos hacer corresponder las fórmulas del lenguaje con su "significado", este significado es un elemento de un conjunto de valores, llamados valores semánticos, que en la lógica clásica está compuesto únicamente por "verdadero" y "falso".

Por otra parte, como ya hemos comentado, nuestro interés está centrado en estudiar la validez de los razonamientos, para lo cual debemos introducir el concepto de deducción semántica.

La semántica tiene como fin dar *significado* a las fórmulas del lenguaje a partir de su estructura sintáctica y establecer la noción (semántica) de *deducción*.

Cuando ponemos en relación las fórmulas del lenguaje con los entes (concretos o abstractos) designados por ellas, damos el paso de la sintaxis a la semántica.

Describiremos la semántica en términos de álgebras abstractas:

Sea un lenguaje $\mathcal{L} = (L, F_L)$, un álgebra similar $\mathcal{S} = (S, F_S)$ a $\mathcal{L}$ y sea $D \subseteq S$. A la terna $\mathcal{M} = (S, D, F_S)$ se la llama *matriz* para el lenguaje $\mathcal{L}$. El conjunto $S$ se denomina *conjunto de valores semánticos* y el conjunto $D$ se denomina *conjunto de valores semánticos destacados*.

En particular, si $|S| = k \in \mathbb{N}^\star$, $D = \{d_1, d_2, \ldots d_m\} \subseteq S$ y $F_S = \{f_1, f_2, \ldots, f_n\}$ escribiremos la matriz $\mathcal{M}$ del modo $\mathcal{M} = (S, d_1, d_2, \ldots, d_m, f_1, f_2, \ldots, f_n)$, y denominaremos al par $\mathfrak{L} = (\mathcal{L}, \mathcal{M})$ una *lógica k-valuada* con valores destacados $D$.

Sea una lógica $\mathfrak{L} = (\mathcal{L}, \mathcal{M})$ donde: $\mathcal{L} = (L, F_L)$ y $\mathcal{M} = (S, D, F_S)$ y sea $X$ el conjunto de generadores de $\mathcal{L}$:

Se llama *interpretación* a cualquier función $i : X \to S$. Esta función puede extenderse de forma única por el teorema 1.3 a un homomorfismo de álgebras $I : \mathcal{L} \to \mathcal{M}$.

Una fórmula $A \in L$ se dice *satisfacible* si existe una interpretación $I$ tal que $I(A) \in D$. En este caso, se dice que la interpretación $I$ *satisface* a $A$ o que $I$ es un *modelo* para $A$.

Un conjunto de fórmulas $\Omega \subseteq L$ se dice *satisfacible* si existe una interpretación $I$ tal que $I(A) \in D$ para cualquier $A \in \Omega$, a la interpretación $I$ se le llama modelo para $\Omega$ y se dice que $I$ *satisface* a $\Omega$.

Una fórmula $A \in L$ se dice *válida* si cualquier interpretación $I$ es un modelo para $A$.

Dos fórmulas $A, B \in L$ se dicen *lógicamente equivalentes*, denotado $A \equiv B$ si para cualquier interpretación $I$ se tiene que $I(A) = I(B)$.

Dado un conjunto $\Omega \subseteq L$ y una fórmula $A \in L$, se dice que $A$ se *deriva, deduce* o *infiere semánticamente* de $\Omega$, denotado $\Omega \models A$, si todo modelo para $\Omega$ es un modelo para $A$.

Este desarrollo semántico sobre un lenguaje es lo que se conoce como una *teoría de modelos* para el lenguaje.

### 1.3.3. Teoría de la Demostración

La introducción de reglas de inferencias como patrones formales que permiten deducir unas fórmulas a partir de otras y que manipulan las distintas componentes del lenguaje de modo puramente sintáctico es lo que se denomina una teoría de

la demostración para un lenguaje. Este mecanismo deductivo será el objeto de este aspecto de la lógica.

Para un *sistema axiomático* sobre un lenguaje $L$, el mecanismo deductivo viene dado por un conjunto numerable de fórmulas de $L$, llamadas *axiomas* y un conjunto de *reglas de inferencia, deducción o transformación* que establecen cuándo una fbf de $L$ es "consecuencia inmediata" de una o varias fbfs de $L$.

En un sistema axiomático $\mathcal{S}$, con lenguaje asociado $L$, se dice que la secuencia $\langle A_1, A_2, \ldots, A_n\rangle$ con $A_i \in L$ es una *demostración* de longitud $n$ de $A_n$ si y sólo si cada una de las fórmulas $A_i$ es un axioma o una consecuencia inmediata de algún subconjunto de $\{A_1, A_2, \ldots A_{i-1}\}$.

Una fórmula de $L$ se dice que es un *teorema* en un sistema axiomático $\mathcal{S}$, si existe para ella una demostración. Obviamente, los axiomas de $\mathcal{S}$ son teoremas de $\mathcal{S}$.

En un sistema axiomático $\mathcal{S}$, con un lenguaje asociado $L$, se dice que la secuencia $\langle A_1, A_2, \ldots, A_n\rangle$ con $A_i \in L$ es una *deducción o derivación* de $A_n$ a partir de un conjunto de fórmulas $\Omega \subseteq L$ si y sólo si cada una de las fórmulas $A_i$ es una fórmula de $\Omega$, o un axioma, o una consecuencia inmediata de algún subconjunto de $\{A_1, A_2, \ldots A_{i-1}\}$. Este hecho se representa mediante $\Omega \vdash A_n$.

Dado un sistema axiomático $\mathcal{S}$, si $\Omega \vdash A$ entonces toda deducción de $A$ desde $\Omega$ es una demostración de $A$ en un sistema $\mathcal{T}$ tal que $\mathcal{T}$ tiene las mismas reglas de inferencia que $\mathcal{S}$ y sus axiomas son los axiomas de $\mathcal{S}$ junto con las fórmulas de $\Omega$.

Los tres aspectos: lenguaje, semántica y teoría de la demostración, constituyen una *teoría formal*.

Dada una teoría formal $\mathcal{T}$, una interpretación es un *modelo para* $\mathcal{T}$ si es un modelo para todos los teoremas de $\mathcal{T}$.

Una teoría formal se dice *correcta* si cumple que

$$\text{Si } \Omega \vdash A \text{ entonces } \Omega \models A$$

En particular, en una teoría correcta todo teorema es una fórmula válida.

Una teoría formal se dice *completa* si cumple que

$$\text{Si } \Omega \models A \text{ entonces } \Omega \vdash A$$

En particular, en una teoría completa toda fórmula válida es un teorema.

En una teoría formal es deseable, aunque desgraciadamente no siempre alcanzable, que ésta sea correcta y completa. Cuando esto sucede se establece un camino de "ida y vuelta" entre la teoría de modelos y la teoría de la demostración, es decir, se dispone de la identificación de la derivabilidad semántica y la derivabilidad sintáctica.

Es fácil conseguir que una teoría formal sea correcta, ya que basta con que los axiomas sean fórmulas válidas y las reglas de inferencia "respeten" la validez, es decir, que las "consecuencias inmediatas" de fórmulas válidas sean fórmulas válidas. Por el contrario, conseguir que una teoría formal sea completa es generalmente bastante más difícil ya que hemos de asegurar que no hemos olvidado ningún axioma ni ninguna regla necesaria para abarcar todas las fórmulas válidas. Aún más, la completitud no siempre es alcanzable, esto se debe a que con ella se pretende nada menos que *caracterizar sintácticamente la validez*, es decir, nuestra comprensión (recogida en la semántica) sobre ciertos aspectos del mundo real; lo cuál es a veces imposible, por excesivamente ambicioso.

En el mejor de los casos, aunque dispongamos de una teoría completa, la mayoría de los teoremas de completitud para un sistema de demostración dado prueban que si una fórmula (o inferencia) es válida entonces existe para ella una demostración en el sistema, pero no indican cómo encontrarla.

Consecuentemente, para hacer de la Lógica una herramienta no sólo atractiva sino también efectiva, es preciso considerar un cuarto aspecto, el aspecto computacional.

### 1.3.4. La Automatización de las Deducciones

Este aspecto es también de naturaleza sintáctica y está estrechamente ligado al aparato deductivo. Es la componente intrínsecamente computacional, pretende no sólo poder establecer qué deducciones son lícitas sino además disponer de un procedimiento de decisión (*algoritmo*) tal que, dada una fórmula $A$, decida en un número finito de etapas si $A$ es o no válida o, centrándonos en los razonamientos, si $A$ se deduce o no de un conjunto de fórmulas $\Omega$. Se denomina *Demostración Automática de Teoremas*, a la búsqueda de procedimientos de decisión de este tipo y ATPs (Automated Theorem Provers) a los algoritmos .

Si una lógica dispone de un ATP se dice que es *decidible*. La Lógica Proposicional es decidible, pero Church y Turing demostraron independientemente (Church, 1936; Turing, 1936; Mendelson, 1987) en 1936 que la Lógica de Predicados de Primer Orden no lo es. Para la Lógica de Predicados de Primer Orden, la demostración automática de teoremas sólo podrá aspirar a encontrar procedimientos llamados de *semidecisión*, es decir procedimientos que garantizan la respuesta afirmativa si su entrada es una fórmula válida o un razonamiento correcto, pero cuando la entrada no es válida puede dar respuesta o no terminar. Una lógica con un procedimiento de semidecisión se dice lógica *semidecidible*.

En los casos en los que sea imposible encontrar un procedimiento de decisión debemos conformarnos con encontrar un procedimiento de semidecisión, pero incluso en el mejor de los casos, y con una lógica decidible, si el tiempo necesitado para obtener una respuesta es excesivo, estaremos muy lejos de nuestro objetivo

de poder aplicar estos procedimientos de forma práctica. Por tanto la búsqueda de un ATP eficiente para las lógicas que deseemos utilizar es la traducción de nuestro objetivo en términos de demostración automática de teoremas.

La sospecha de que no existe un procedimiento que ejecute en tiempo polinómico la decisión de si una fórmula es o no válida, ni siquiera para la Lógica Proposicional Clásica, nos lleva a centrar nuestro objetivo en encontrar ATPs que sólo tomen un tiempo excesivo para determinar la validez de una fórmula o inferencia en casos excepcionales, y que se ejecuten satisfactoriamente para la mayoría de las entradas.

Un aspecto en el que nuestro grupo de investigación GIMAC ha centrado su mayor esfuerzo es en la consecución de ATPs que no empleen un tiempo "excesivo" en determinar la validez de un razonamiento. Más concretamente, nuestro objetivo es que todo desarrollo teórico que podamos aportar en el área de la demostración automática conlleve aplicaciones prácticas de las mismas.

## 1.4. Tipos de Lógicas

La Lógica Clásica Proposicional es el tipo de lógica más elemental, el más conocido y posiblemente el más utilizado. Como es bien conocido, esta lógica se caracteriza porque:

- es independiente del contexto, es decir, considera únicamente construcciones declarativas (proposiciones) sobre las que podemos pronunciarnos acerca de su verdad o falsedad sin recurrir a consideraciones de *contexto* alguno, ni a consideraciones sobre la estructura interna de estas proposiciones.
- es veritativa funcional, es decir, contempla la verdad composicionalmente: la verdad de una construcción compuesta queda determinada por la verdad de sus componentes.
- verifica la ley del tercio excluso, es decir, las proposiciones tienen que ser verdaderas o falsas, y no hay más posibilidades. Por tanto la proposición "$p$ o no es el caso que $p$" es siempre verdadera (válida) sea cual sea la proposición $p$ que se elija.

El hecho de no considerar la estructura interna de los enunciados (proposiciones) hace que en la Lógica Clásica Proposicional el enunciado

"Si todo hombre es mortal entonces algún hombre es mortal"

no sea válido, ya que este enunciado se formalizaría como "si p entonces q", que no es una fórmula válida. Es necesario, pues, ampliar la Lógica Clásica

Proposicional de forma que pueda hacerse un análisis interno de los enunciados en el que se distinga el qué se predica y de quién o qué se predica, es decir, necesitamos considerar la Lógica Clásica de Predicados.

Al igual que se ha necesitado una lógica distinta para capturar un concepto de "verdad" que la Lógica Clásica Proposicional no capturaba, el análisis eficaz y adecuado de otros razonamientos requiere otros tipos de lógicas, conocidas como lógicas no clásicas.

En general se denomina lógica no clásica a cualquier lógica distinta de la Lógica Clásica Proposicional y de la Lógica Clásica de Predicados. Estas lógicas incumplen uno o más puntos de los anteriormente comentados; así podemos considerar lógicas en los que se permita considerar contextos temporales, de creencia, de necesidad, etc.

Las lógicas no clásicas han adquirido gran protagonismo en las Ciencias de la Computación al ser utilizadas por disciplinas tales como control automático, simulación de circuitos digitales, planificación, diagnosis, comprensión del lenguaje natural, sistemas expertos, verificación de programas, etc. Normalmente es en este tipo de lógicas donde el aspecto de una mecanización eficiente de las deducciones tiene una importancia capital, ya que una mayor complejidad en el tipo de lógica para admitir nuevos aspectos, trae normalmente aparejado un considerable incremento en la complejidad de estas deducciones.

Siguiendo a Susan Haak (Haak, 1978) podemos clasificar las lógicas no clásicas en dos grandes grupos:

- Extensiones de la lógica clásica. Pueden hablar de cosas de las que la lógica clásica no puede, extendiendo el vocabulario básico de ésta. Por lo tanto, añaden nuevas leyes a las de la lógica clásica.

  Ejemplos de este tipo de lógicas son aquéllas en las que el análisis de los razonamientos consideran contextos de tiempo (lógicas *temporales*), de necesidad y posibilidad (lógicas *modales*), de creencia (lógicas *doxásticas*), etc.

- Desviaciones o rivales de la lógica clásica. Utilizan el mismo vocabulario que la lógica clásica, pero no mantienen algunas de las leyes de ésta.

  Un ejemplo destacado de lógica rival lo constituyen las lógicas multivaluadas objetivo de este trabajo.

  Otro ejemplo significativo es el de la lógica *Intuicionista* o *constructivista*. Esta lógica fue introducida por Brouwer, aunque las nociones de verdad y validez intuicionistas fueron axiomatizadas por Heyting. Para un intuicionista, $A$ es verdadero sólo si es posible dar "constructivamente" una realización de $A$ o probar "constructivamente" la existencia de $A$.

Ya desde muy antiguo se criticaba a las lógicas clásicas el hecho de que las cosas tuviesen que ser necesariamente verdaderas o falsas, principalmente desde dos puntos de vista, de un lado porque las cosas pueden ser "verdades a medias" y del otro porque aunque en el fondo consideremos que las cosas son verdaderas o falsas podemos desconocer si una proposición determinada lo es y sin embargo realizar razonamientos donde intervenga dicha proposición. Este planteamiento da lugar a lógicas donde no se verifica la ley del tercio excluso, es decir hay más de dos valores de verdad. Es a este tipo de lógicas, concretamente a algunas lógicas trivaluadas (hay tres valores semánticos posibles) a las que dedicaremos nuestra atención en este trabajo.

# Capítulo 2

# Lógicas multivaluadas

## 2.1. Algunas notas históricas

Los orígenes de las lógicas multivaluadas se remontan hasta los tratados de filosofía de la antigua Grecia.

Aristóteles fue el primero en poner objeciones a la ley del tercero excluido $A \vee \neg A$ considerada como verdad indiscutible en la lógica clásica. La idea de aceptar enunciados que no son ni absolutamente verdaderos ni absolutamente falsos provocó discusiones entre los epicúreos y los estoicos. Los epicúreos rechazaban totalmente el determinismo, admitiendo la posibilidad de que de dos enunciados, uno de ellos negación del otro, ninguno de ellos sea verdadero; en particular, cuando los enunciados se refieren a sucesos futuros. Por su parte, los estoicos, y en particular Crisipo de Soli (279-206 a.C.), segundo fundador de la Estoa, representan el punto de vista del determinismo extremo con la defensa a ultranza de la bivalencia. Diógenes Laercio escribió que, en opinión de la mayoría de los lógicos posteriores a Crisipo, si los dioses usaran algún tipo de lógica, usarían la lógica de Crisipo. Esta es la razón por la que las lógicas multivaluadas y, más en general las lógicas no clásicas, se denominan a veces lógicas no crisipianas.

Más recientemente, podemos considerar como pioneros de las lógicas multivaluadas a G. Boole (1815-1964), C.S. Peirce (1839-1914) y N. A. Vasiliev y como actuales autores de su rigurosa fundamentación a J. Łukasiewicz (1878-1956), que elaboró en 1920 una lógica trivalente, y a E. Post (1897-1954) que en el mismo año proporcionó un marco general de lógicas polivalentes con un conjunto finito de valores de verdad, concretamente, una lógica $n$-valuada funcionalmente completa. A estos autores, independientemente, se deben las primeras descripciones sistemáticas de sistemas lógicos multivaluados; el primero de ellos, motivado por razones filosóficas, y el segundo por consideraciones matemáticas.

El trabajo de Łukasiewicz en 1920 fue el punto de partida para la construcción de lógicas multivaluadas, así como de sistemas infinito-valuados.

Posteriormente, Łukasiewicz en su obra 'Sobre el Determinismo" argumenta que la consideración de enunciados sobre el futuro requiere rechazar la ley del tercero excluido, por tanto, es preciso añadir un tercer valor de verdad leído como "posible". Hay enunciados de los que no sólo no conocemos su valor de verdad, sino que ahora no lo poseen. Łukasiewicz recoge así el pensamiento de Aristóteles: "Si uno supone que enunciados sobre el futuro son ahora verdaderos o falsos, uno se ve abocado al fatalismo". La lógica de Łukasiewicz introduce el valor semántico $\frac{1}{2}$ para tratar enunciados en un futuro contingente. El sistema lógico resultante fue desarrollado por Łukasiewicz y sus colaboradores entre 1920 y 1930. Sus resultados teóricos han sido recogidos en el trabajo de Łukasiewicz y Tarski (Łukasiewicz y Tarski, 1930).

Sin embargo, el interés por estas lógicas ha sido escaso hasta fechas muy recientes. En particular, el interés en el área de la demostración automática en lógicas multivaluadas ha aumentado en los últimos diez años debido a su aplicación en computación. La razón de la escasa bibliografía sobre el tema hasta entonces se debe, básicamente, por una parte al convencimiento de que el interés sobre estas lógicas quedaba reducido exclusivamente a los matemáticos puros (en particular a aquellos interesados en el álgebra y en teoría espectral) y por otra a que diferentes aplicaciones requieren diferentes tipos de lógicas multivaluadas y a que es difícil comparar diferentes lógicas multivaluadas. Para destacar la desconexión de estas lógicas con una visión en el marco de la matemática aplicada, basta referirnos a comentarios que, hace tan sólo unos pocos años, podíamos encontrar sobre estas lógicas. Elegimos, por su radicalidad, el siguiente:

> "Cualquiera puede entender $\{0, 1\}$-interpretaciones, pero pocos, incluidos sus creadores, pueden entender las tablas de verdad multivaluadas ..." (D. Scott 1976)

Como destaca A. Urquhart (Urquhart, 1986), el interés en las lógicas multivaluadas se debe en gran parte al esfuerzo de los ingenieros y los científicos de la Computación. Así queda constancia en la historia sobre demostración automática de teoremas que, como hemos referido en la introducción, incluye H. Reiner en (Hähnle, 1993).

## 2.2. Lenguaje y Semántica

### 2.2.1. El Lenguaje

Para definir el lenguaje de una lógica proposicional multivaluada, al igual que en la Lógica Clásica Proposicional, partimos de un conjunto numerable de

símbolos de proposición $Q = \{p, q, \ldots, p_1, q_1, \ldots, p_n, q_n, \ldots\}$, un conjunto de símbolos de puntuación $P = \{(, ), [, ]\}$ y un conjunto finito de conectivas u operadores $Op$ que variará según la lógica multivaluada proposicional considerada.

En algunas ocasiones se utilizarán los mismos símbolos que para las conectivas de la Lógica Clásica Proposicional, es decir, las conectivas binarias (de aridad 2) $O_{bin} = \{\vee, \wedge, \rightarrow, \leftrightarrow\}$, la conectiva monaria (de aridad 1) $O_{un} = \{\neg\}$ y las constantes (de aridad 0) $O_{const} = \{\top, \bot\}$. Sea cual sea el caso, el alfabeto $\nabla$ estará constituido por la unión de los conjuntos de símbolos de proposición, símbolos de puntuación y símbolos de operación, es decir: $\nabla = Q \cup P \cup Op$, donde $Op = O_{const} \cup O_{un} \cap O_{bin}$.

En el caso de que todos los operadores sean de aridades 0, 1 y 2, el lenguaje $\mathcal{L}$ se obtiene como la clausura inductiva del conjunto $Q \cup O_{const}$ para el conjunto de conectivas $\{f_- \mid - \in O_{un}\} \cup \{f_{op} \mid op \in O_{bin}\}$, que se definen de la forma siguiente:

$$f_-(\alpha) = -\alpha$$

$$f_{op}(\alpha, \beta) = (\alpha\, op\, \beta)$$

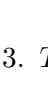
donde $\alpha, \beta$ son elementos del lenguaje universal sobre el alfabeto $\nabla$, es decir $\alpha, \beta \in \nabla^*$.

A las cadenas de $\mathcal{L}$ se les llama fórmulas bien formadas (*fbfs*). En la práctica se suprimen los paréntesis innecesarios, como por ejemplo los de inicio y fin de una *fbf*.

La clausura inductiva definida anteriormente resulta ser libremente generada por y lo tanto, como hemos indicado en la introducción, cada elemento $A$ del lenguaje $L$ se representa de modo único por un árbol sintáctico $T_A$ definido recursivamente como sigue:

1. $T_p$ es $p$ si $p \in Q$.

2. $T_{-A}$ es  donde $- \in O_{un}$.

3. $T_{AopB}$ es  donde $op \in O_{bin}$.

En el caso general de conectivas $n$-arias, obtendremos nodos con $n$ hijos.

En términos de álgebras abstractas el lenguaje viene dado por la subálgebra del álgebra $(\nabla, O_{bin} \cap O_{un})$ libremente generada por $Q \cup O_{const}$.

### 2.2.2. La Semántica

A diferencia de la Lógica Clásica Proposicional en la que el conjunto de valores semánticos $S$ se limitaba a dos valores $\{0, 1\}$, en las lógicas multivaluadas el conjunto $S$ tendrá $n$ valores, con $n > 2$. En general, elegiremos como valores $n$ números racionales equidistantes en el intervalo $[0, 1]$, es decir, para el caso de lógicas trivaluadas tomaremos como $S = \{0, \frac{1}{2}, 1\}$ y para $n$ arbitrario $S = \{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$. Para el conjunto de valores destacados $D$ se toma habitualmente los $k$ elementos mayores de $S$ con $(1 \leq k \leq n)$, concretamente $D = \{\frac{n-k}{n-1}, \frac{n-k+1}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$.

Para abreviar la notación, al conjunto $\{0, \frac{1}{2}, 1\}$ lo representaremos por **3** y, en general, al conjunto $\{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$ por $\mathbf{n}$.

En definitiva, la matriz $n$-valuada para un lenguaje $\mathcal{L} = (Q, op_1, op_2, \ldots, op_k)$ viene dada por un álgebra $\mathcal{M} = (\mathbf{n}, d_1, d_2, \ldots, d_m, op'_1, op'_2, \ldots, op'_k\}$ similar a $\mathcal{L}$. En la práctica se suele utilizar el mismo símbolo para $op$ y para $op'$.

Dados los conjuntos $S$ y $D$ definimos una interpretación $i$ como una función del conjunto de símbolos proposicionales en el conjunto de valores semánticos, es decir $i : Q \to S$. Denotaremos por $I$ la extensión homomórfica de $i$ a $\mathcal{L}$ $(I : \mathcal{L} \to S)$.

Como es habitual, una vez introducido el concepto de interpretación se definen los conceptos de modelo, satisfacibilidad y deducción semántica de la forma estándar.

Dadas dos fórmulas $A$ y $B$ se dice que son *lógicamente equivalentes* y se representa por $A \equiv B$ si para cualquier interpretación $I$ se tiene que $I(A) = I(B)$.

Como en la lógica clásica, también para la lógica multivaluada se dispone de un resultado (Teorema de equivalencia) que proporciona un mecanismo simple (llamado sustitución) mediante el cual se obtienen nuevas equivalencias, a partir de equivalencias conocidas. Para su formalización introducimos las siguiente notaciones.

**Notación:**

- $A[B]$ denota que $B$ es una subfórmula de $A$.
- Si $A[B]$ entonces $A[B/C]$ denota que $B$ es una subfórmula de $A$ y que se han sustituido en $A$ todas las ocurrencias de $B$ por la fórmula $C$.

A continuación utilizamos esta notación para expresar el conocido Teorema de Equivalencia:

**Teorema 2.1** *Si* $A[B]$ *y* $B \equiv C$ *entonces* $A \equiv A[B/C]$.

## 2.3. Lógicas Multivaluadas y Deducción

Antes de introducir los ejemplos más conocidos de lógicas multivaluadas, vamos a profundizar en la noción de deducción, objetivo esencial de nuestra investigación. Para ello, siguiendo a (E. Trillas y Terricabras, 1995), nos acercaremos a ella desde un punto de vista puramente algebraico.

Un modo de entender la deducción o inferencia lógica, el que más nos agrada, es como *generación de información* tanto explícita como *implícita*; es decir, si afirmamos que una fórmula $A$ se deduce de un conjunto $\Omega$ de fórmulas, entendemos que la información en $\Omega$ contiene explícita o implícitamente la información de $A$. Cuando afirmamos que $\Omega \cup \{A\} \models A$ estamos extrayendo información explícita y cuando afirmamos que $\Omega \models A$ con $A \notin \Omega$ estamos extrayendo información implícita. Obviamente, nuestro interés se centra en la extracción de información implícita.

Con esta visión de la deducción semántica, podemos leer $A \models B$ como "la información recogida por $A$ es mayor o igual que la información recogida por $B$". Estamos leyendo $\models$ como una relación binaria en el lenguaje $L$, la cual es claramente reflexiva y transitiva, es decir, es un preorden.

Vamos a formalizar esta lectura de la deducción en un marco general.

### 2.3.1. Operadores de Deducción y Preórdenes

Comenzamos expresando la deducción como operador que asocia a cada conjunto de *hipótesis* el conjunto de las *consecuencias* que se pueden deducir de ellas.

**Definición 2.1** Sea $E$ un conjunto no vacío, una función

$$Con : 2^E - \varnothing \longrightarrow 2^E - \varnothing$$

se denomina *operador de deducción o razonamiento* si satisface las propiedades siguientes:

1. $H \subseteq Con(H)$ para todo subconjunto $H$ de $E$.
2. Si $H \subseteq Con(H')$ entonces $Con(H) \subseteq Con(H')$.
3. $Con(Con(H)) \subseteq Con(H)$.

Si contemplamos $H$ como un conjunto de hipótesis y $Con(H)$ como el conjunto de *consecuencias* de $H$, por la primera condición, cada $h \in H$ es una consecuencia de $H$; la segunda condición es equivalente a la siguiente:

2b. Si $H \subseteq H'$ entonces $Con(H) \subseteq Con(H')$.

y asegura la monotonía, es decir, al aumentar el número de hipótesis aumenta el número de consecuencias y la tercera asegura que no hay más consecuencias que $Con(H)$. Es preciso pues destacar que estamos contemplando un operador de razonamiento *monótono*.

El teorema 1.1 nos permite la lectura de los operadores de deducción en términos de preórdenes:

**Teorema 2.2** *Sea $E$ un conjunto no vacío, una función*

$$Con : 2^E - \varnothing \longrightarrow 2^E - \varnothing$$

*es un* operador de deducción o consecuencia lógica *si y sólo si la relación $R$ en $E$ definida por $R = \{(a, b) \mid a \in E, b \in Con(\{a\})\}$ es un preorden.*

Demostración: Puesto que para todo subconjunto no vacío de $H$ de $E$ se tiene que $Con(H) = R(H)$, el resultado es consecuencia inmediata del teorema 1.1. *q.e.d.*

### 2.3.2. Deducción y Condicionales

Deseamos un formalismo adecuado que nos permita realizar deducciones. Como punto de partida, y casi inevitablemente, hemos de enfocar a la regla modus ponens:

**Definición 2.2** Sea $E$ un conjunto no vacío, $R$ una relación binaria en $E$ y $V$ un subconjunto no vacío de $E$. Decimos que la relación $R$ es un *condicional lógico respecto a $V$* (en adelante, $R$ es un $V$-condicional) o que $V$ es un conjunto de elementos verdaderos para $(E, R)$ (en adelante, $V$ es una $v$-parte de $(E, R)$), si se satisface la condición siguiente:

$$\text{si } a \in V \text{ y } (a, b) \in R, \text{ entonces } b \in V$$

Es decir, si $R(V) \subseteq V$ o lo que es lo mismo, $R$ respeta a $V$.

Una atenta lectura de la definición anterior delata su debilidad; en efecto, toda relación $V \times X$ tal que $V \times X \subseteq V \times V \subseteq E \times E$ es un $V$-condicional. Por tanto, existen $V$-condicionales que no son preórdenes y que, en consecuencia, no nos proporcionan operadores de deducción.

**Definición 2.3** Sea $E$ un conjunto no vacío, $R$ una relación binaria en $E$, y $F$ un subconjunto no vacío de $E$. Decimos que $F$ es un conjunto de elementos falsos para $(E, R)$ (en adelante, $F$ es una $f$-parte de $(E, R)$), si se satisface la condición siguiente:

$$\text{si } b \in F \text{ y } (a,b) \in R, \text{ entonces } a \in F$$

Es decir, si $R^{-1}(F) \subseteq F$, o lo que es lo mismo, $R^{-1}$ respeta a $F$.

El siguiente teorema nos indica respecto a qué subconjuntos de $E$ se tiene que un preorden es condicional lógico .

**Teorema 2.3** *Sea $E$ un conjunto y $R$ un preorden sobre $E$. Entonces,*

1. *$R$ es un condicional lógico para todo subconjunto $V$ no vacío de $E$ tal que $R(V) = V$.*
2. *Si $E_0$ es un subconjunto no vacío de $E$, $R$ es condicional lógico respecto a $V = R(E_0)$.*

DEMOSTRACIÓN:

1. Es inmediato de la Definición 2.2.
2. Basta advertir que por ser $R$ transitiva, se tiene que $R(V) = R(R(E_0)) \subseteq R(E_0) = V$.

*q.e.d.*

**Ejemplo 2.1**

- Sea $L$ un lenguaje de la Lógica Clásica Proposicional, $I : L \to \{0,1\}$ una interpretación y $V_I = \{A \in L \mid I(A) = 1\}$. Entonces la relación en $L$

  $R_I(A,B)$ si y sólo si $I(A) \leq I(B)$, es decir, $R_I(A,B)$ si y sólo si $A \models_I B$

  es un condicional lógico respecto a $V_I$.

- Sea $E = \mathbb{Q}$ el conjunto de los números racionales y $V = \mathbb{Z}$ el de los números enteros. Entonces la relación en $\mathbb{Q}$

  $$R_{\mathbb{Z}}(a,b) \text{ si y sólo si } a - b \in \mathbb{Z}$$

  es un condicional lógico respecto a $\mathbb{Z}$.

- Sea $E$ un espacio vectorial y $V = H$ un subespacio de $E$. Entonces la relación en $E$

  $$R_H(\vec{x},\vec{y}) \text{ si y sólo si } \vec{x} - \vec{y} \in H$$

  es un condicional lógico respecto a $H$.

Todos los condicionales lógicos del ejemplo anterior son preórdenes y, por lo tanto, proporcionan operadores de deducción.

### 2.3.3. Condicional Material

De lo expuesto hasta aquí, podemos concluir que para tener asegurado que capturamos toda la potencia deseada para la deducción necesitamos $V$-condicionales, pero que debemos tomar para cada $V$ el mayor de los $V$-condicionales. Como veremos a continuación, este análisis nos conduce al condicional material:

**Definición 2.4** Sea $E$ un conjunto no vacío, $\mathcal{P}$ una partición de $E$ y $\varnothing \neq V \in \mathcal{P}$. Llamamos *condicional material asociado a* $V$, a la relación en $E$ definida por

$$\rightarrow_V = V^c \times E \ \cup \ V \times V$$

En particular, si la partición en $E$ es $E = F \ \cup \ V$,

$$\rightarrow_V = F \times E \ \cup \ V \times V = F \times F \ \cup \ F \times V \ \cup \ V \times V$$

**Ejemplo 2.2** En el ejemplo 2.1, $R_I$ es un condicional material asociado a $V$, pero $R_{\mathbb{Z}}$ y $R_H$ no lo son.

**Teorema 2.4** *Sea $E$ un conjunto no vacío, $\mathcal{P}$ una partición de $E$ y $\varnothing \neq V \in \mathcal{P}$, entonces se tiene que:*

1. *$\rightarrow_V$ es un preorden.*
2. *$\rightarrow_V$ es un condicional lógico respecto a $V$.*
3. *Una relación $R$ en $E$ es un condicional lógico respecto a $V$ si y sólo si $R \subseteq \rightarrow_V$. Así pues, $\rightarrow_V$ es el mayor de los condicionales lógicos respecto a $V$.*

Demostración:

1. Es de comprobación inmediata por definición de $\rightarrow_V$.
2. Si $a \in V$ y $a \rightarrow_V b$, por definición de $\rightarrow_V$, se tiene que $(a, b) \in V \times V$, es decir, $b \in V$; por tanto $\rightarrow_V$ es un condicional lógico respecto a $V$.
3. Supongamos que $R \subseteq \rightarrow_V$. Entonces, si $a \in V$ y $(a, b) \in R$, se tiene también que $a \rightarrow_V b$ y, puesto que $\rightarrow_V$ es un condicional lógico respecto a $V$, se tiene que $b \in V$, es decir, $R$ es un condicional lógico respecto a $V$.

   Recíprocamente, supongamos que $R$ es un condicional lógico respecto a $V$ y sea $(a, b) \in R$. Tenemos dos casos posibles:

   - $a \in V$, en cuyo caso, por ser $R$ un $V$-condicional, se tiene que $b \in V$; es decir $(a, b) \in \rightarrow_V$.

- $a \in V^c$. En este caso se tiene que $(a, b) \in V^c \times E$; es decir, $(a, b) \in \rightarrow_V$.

*q.e.d.*

Como consecuencia de lo anterior podemos concluir lo que sigue:

Sea $E$ un conjunto no vacío, $V$ un subconjunto no vacío de $E$ y $\Rightarrow_V$ un $V$-condicional, y denotemos $a \Rightarrow_V b$ en lugar de $(a, b) \in \Rightarrow_V$, entonces, podemos asegurar que:

- Si $\Rightarrow_V$ es un preorden y $a \Rightarrow_V b$ entonces se tiene que, por el teorema 2.2, $b$ es consecuencia de $a$ respecto a $\Rightarrow_V$.
- Si $\Rightarrow_V$ no es un preorden y $a \Rightarrow_V b$ entonces se tiene que, por el teorema 2.4, $\Rightarrow_V \subseteq \rightarrow_V$ y puesto que $\rightarrow_V$ es un preorden, el teorema 2.2 asegura que $b$ es consecuencia de $a$ respecto a $\rightarrow_V$.

En definitiva, el condicional material suple toda la "lógica" que pueda faltarle a un $V$-condicional y es el condicional que corresponde a un razonamiento formal monótono.

### 2.3.4. Conjunciones y condicionales

Veamos ahora cuál es la conjunción adecuada al concepto de deducción:

**Definición 2.5** Sea $E$ un conjunto no vacío y $R$ una relación binaria en $E$. Una operación binaria $\cdot : E \times E \longrightarrow E$ se dice una *conjunción* respecto de $R$ si satisface que

$$(a, b) \in R \text{ si y sólo si existe } c \in E \text{ tal que } a = b \cdot c$$

**Definición 2.6** Sea $E$ un conjunto no vacío y sea $\mathcal{P}$ una partición de $E$ y $\varnothing \neq V \in \mathcal{P}$. Dada una relación binaria $R$ en $E$ y una conjunción $\cdot$ respecto a $R$, decimos que $\cdot$ es una *conjunción lógica* o una operación "y", para $V$ si se satisface que

$$a \cdot b \in V \text{ si y sólo si } a \in V \text{ y } b \in V$$

En particular, si la partición en $E$ es $E = F \cup V$, es equivalente a

$$a \cdot b \in F \text{ si y sólo si } a \in F \text{ ó } b \in F$$

**Teorema 2.5** *Si en $(E, R)$ hay una conjunción lógica para $V$ entonces $R \subseteq \rightarrow_V$.*

Demostración: Sea $\cdot$ una conjunción lógica para $V$. Entonces,

- Si $a \in V^c$ y $(a, b) \in R$, entonces $(a, b) \in V^c \times E \subseteq \rightarrow_V$.
- Si $a \in V$ y $(a, b) \in R$ y entonces existe $c \in E$ tal que $a = b \cdot c$ y puesto que $\cdot$ es una conjunción lógica, se tiene que $b \in V$ y por tanto, $(a, b) \in V \times V \subseteq \rightarrow_V$.

Tenemos pues que $R \subseteq \rightarrow_V$. *q.e.d.*

**Definición 2.7** Dada una estructura $(E, R)$, una conjunción $\cdot$ respecto a $R$ se dice que es una *conjunción ordenadora* si

$$(a, b) \in R \text{ si y sólo si } a = b \cdot a$$

**Teorema 2.6** *Si en $(E, R)$ existe una conjunción ordenadora $\cdot$, entonces:*

1. *$\cdot$ es idempotente si y sólo si $R$ es reflexiva.*
2. *Si $\cdot$ es conmutativa entonces $R$ es antisimétrica.*
3. *Si $\cdot$ es asociativa entonces $R$ es transitiva.*
4. *Si $\cdot$ tiene como elemento absorbente $0$ entonces se tiene que $(0, a) \in R$ para todo $a \in E$.*
5. *Si $\cdot$ tiene un elemento neutro $1$ entonces se tiene que $(a, 1) \in R$ para todo $a \in E$.*

Por lo tanto, una condición necesaria para que exista una conjunción ordenadora idempotente, asociativa y conmutativa respecto a $R$ es que $R$ sea un orden parcial.

**Ejemplo 2.3**

1. En la Lógica Clásica Proposicional, para $(L/\equiv, \models)$ se tiene que $\wedge$ es una conjunción ordenadora respecto de $\models$ ya que

$$A \models B \text{ si y sólo si } A \equiv B \wedge A$$

2. Para $(2^E, \subseteq)$ se tiene que $\cap$ es una conjunción ordenadora respecto de $\subseteq$ ya que

$$A \subseteq B \text{ si y sólo si } A = B \cap A$$

3. Para $(\mathbb{Z}, \leq)$ donde $\leq$ es el orden habitual en $\mathbb{Z}$, se tiene que $-$ es una conjunción (no ordenadora) respecto de $\leq$ ya que

$$a \leq b \text{ si y sólo si existe } c \in \mathbb{Z} \ (c = b - a) \text{ tal que } a = b - c$$

### 2.3.5. Disyunciones y condicionales

Dualmente, disponemos de la disyunción adecuada a la deducción.

**Definición 2.8** Sea $E$ un conjunto no vacío y $R$ una relación binaria en $E$. Una operación binaria $+ : E \times E \longrightarrow E$ se dice una *disyunción* respecto de $R$ si satisface que

$$(a, b) \in R \text{ si y sólo si existe } d \in E \text{ tal que } b = a + d$$

**Definición 2.9** Sea $E$ un conjunto no vacío y sea $\mathcal{P}$ una partición de $E$ y $\varnothing \neq V \in \mathcal{P}$. Dada una relación binaria $R$ en $E$ y una disyunción $+$ respecto a $R$, decimos que $+$ es una *disyunción lógica* o una operación "o" para $V$ si satisface que

$$a + b \in V \text{ si y sólo si } a \in V \text{ ó } b \in V$$

En particular, si la partición en $E$ es $E = F \cup V$, es equivalente a

$$a + b \in F \text{ si y sólo si } a \in F \text{ y } b \in F$$

**Teorema 2.7** *Si en $(E, R)$ hay una disyunción lógica respecto de $V$ entonces $R \subseteq \rightarrow_V$.*

Demostración: Sea $+$ una disyunción lógica respecto de $V$. Entonces,

- Si $b \in V^c$ y $(a, b) \in R$ existe $d \in V$ tal que $b = a + d$ y puesto que $+$ es una disyunción lógica, se tiene que $a \in V^c$ y por tanto, $(a, b) \in V^c \times E \subseteq \rightarrow_V$.
- Si $b \in V$, entonces $(a, b) \in V^c \times E \ \cup \ V \times V = \rightarrow_V$.

Tenemos pues que $R \subseteq \rightarrow_V$. *q.e.d.*

**Definición 2.10** Dada una estructura $(E, R)$, una disyunción $+$ respecto a $R$ se dice *ordenadora* si

$$(a, b) \in R \text{ si y sólo si } b = a + b$$

**Teorema 2.8** *Si en $(E, R)$ existe una disyunción ordenadora $+$, entonces:*

1. *$+$ es idempotente si y sólo si $R$ es reflexiva.*
2. *Si $+$ es conmutativa entonces $R$ es antisimétrica.*
3. *Si $+$ es asociativa entonces $R$ es transitiva.*

4. *Si* $+$ *tiene un neutro* $0$ *entonces se tiene que* $(0, a) \in R$ *para todo* $a \in E$.

5. *Si* $+$ *tiene como absorbente* $1$ *entonces se tiene que* $(a, 1) \in R$ *para todo* $a \in E$.

Por lo tanto, una condición necesaria para que $(E, R)$ tenga una disyunción ordenadora idempotente, asociativa y conmutativa es que $R$ sea un orden parcial.

**Ejemplo 2.4**

1. En la Lógica Clásica Proposicional, para $(L/\equiv, \models)$ se tiene que $\vee$ es una disyunción ordenadora respecto de $\models$ ya que

$$A \models B \text{ si y sólo si } B \equiv A \vee B$$

2. Para $(2^E, \subseteq)$ se tiene que $\cup$ es una disyunción ordenadora respecto de $\subseteq$ ya que

$$A \subseteq B \text{ si y sólo si } B = A \cup B$$

3. Para $(\mathbb{Z}, \leq)$ donde $\leq$ es el orden habitual en $\mathbb{Z}$, se tiene que $+$ es una disyunción (no ordenadora) respecto de $\leq$ ya que

$$a \leq b \text{ si y sólo si existe } d \in \mathbb{Z} \ (d = b - a) \text{ tal que } b = a + d$$

### 2.3.6. Negaciones

**Definición 2.11** Dada una estructura $(E, R)$, donde $E$ es conjunto no vacío y $R$ una relación binaria en $E$, llamamos *negación* respecto de $R$ a toda operación monaria $' : E \to E$ tal que

$$(a, b) \in R \text{ si y sólo si } (b', a') \in R$$

Una negación $'$ en $E$ se dice "fuerte" si para cada $a \in E$ se tiene que $a'' = a$.

**Ejemplo 2.5**

1. En la Lógica Clásica Proposicional, para $(L/\equiv, \models)$ se tiene que $\neg$ es una negación fuerte respecto de $\models$ ya que

$$A \models B \text{ si y sólo si } \neg B \models \neg A \text{ y } \neg\neg A$$

2. Para $(2^E, \subseteq)$, la operación de complementación $(\_)^c$ es una negación fuerte respecto de $\subseteq$ ya que

$$A \subseteq B \text{ si y sólo si } B^c \subseteq A^c \, y \, A^{cc}$$

3. Para $(\mathbb{Z}, \leq)$ donde $\leq$ es el orden habitual en $\mathbb{Z}$, se tiene que la aplicación $x' = -x$ es una negación fuerte respecto de $\leq$ ya que

$$a \leq b \text{ si y sólo si } -b \leq -a \text{ y } -(-a) = a$$

**Definición 2.12** Consideremos $E$ con la partición $E = V \cup F$. Dada una estructura $(E, R)$ y una negación $'$ en $E$, diremos que $'$ es una *negación lógica* si se satisface que

$$a \in V \text{ si y sólo si } a' \in F$$

**Definición 2.13** Sea $(E, R)$ una estructura y $'$ una *negación* en $E$ respecto de $R$. Llamamos *función de verdad* en $(E, R, ')$ a toda función $v : E \to \mathbf{n}$ tal que

1. Si $(a, b) \in R$ entonces $v(a) \leq v(b)$.
2. $v(a) = 0$ si y sólo si $v(a') = 1$.

Advirtamos que una condición suficiente, pero no necesaria, para que la segunda condición sea cierta es que se verifique

2b) $v(a') = 1 - v(a)$.

Esta condición obliga a que $'$ sea fuerte.

**Teorema 2.9** *Sea una función $v : E \to \mathbf{n}$ en $(E, R, ')$ que satisface que si $(a, b) \in R$ se tiene que $v(a) \leq v(b)$, entonces para cada $i \in \mathbf{n}$, el conjunto*

$$v^{-1}([i, 1]) = \{a \in E \mid i \leq v(a) \leq 1\}$$

*es un conjunto de elementos verdaderos para $R$, es decir, $R$ es un condicional lógico respecto de $v^{-1}([i, 1])$.*

DEMOSTRACIÓN: Si $a \in v^{-1}([i, 1])$ y $(a, b) \in R$, puesto que $v(a) \leq v(b)$, se tiene que $v(b) \in [i, 1]$, es decir, $b \in v^{-1}([i, 1])$. *q.e.d.*

**Teorema 2.10** *Si una función $v : E \to \mathbf{n}$ en $(E, R, ')$ satisface que si $(a, b) \in R$ se tiene que $v(a) \leq v(b)$, entonces para cada $i \in \mathbf{n}$, se tiene que el conjunto*

$$v^{-1}([0, i]) = \{a \in E \mid 0 \leq v(a) \leq i\}$$

*es un conjunto de elementos falsos para $R$, es decir, $v^{-1}([i, 1])$ es cerrado respecto a $R^{-1}$.*

DEMOSTRACIÓN: En efecto, si $(a,b) \in R$ y $b \in v^{-1}([0,i])$, puesto que $v(a) \leq v(b)$, se tiene que $v(a) \in [0,i]$, es decir, $a \in v^{-1}([0,i])$. *q.e.d.*

**Definición 2.14** Sea $(E,R)$ una estructura, $'$ una *negación* en $E$ respecto de $R$ y $v : E \to \mathbf{n}$ una función de verdad en $(E,R,')$ que satisface 2b. Un elemento $i \in \mathbf{n}$ se dice que es un *valor destacado* de la lógica $n$-valuada definida en $(E,R,')$ por $v$ si $i > \frac{1}{2}$.

Es obvio que si consideramos que $\{a \in E \mid v(a) \in [i,1]\}$ es el conjunto de los elementos verdaderos y que $\{a \in E \mid v(a) \in [0,1-i]\}$ es el de los elementos falsos entonces $\{a \in E \mid v(a) \in (1-i,i)\}$ es el conjunto de los elementos con un valor de verdad intermedio.

**Ejemplo 2.6** Si $v : E \to \mathbf{5}$ y tomamos como valor destacado $\frac{3}{4}$, obtenemos una partición de $E$ en el conjunto $D = \{\frac{3}{4}, 1\}$, el conjunto de elementos con valor en $\{0, \frac{1}{4}\}$ y el conjunto de elementos con valor $\frac{1}{2}$. Tendríamos así una lógica trivaluada. Si tomamos como valor destacado 1, obtenemos una partición de $E$ en $D = \{1\}$, el conjunto de elementos con valor 0 y el conjunto de elementos con valor en $\{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. Si consideramos este último conjunto como unión disjunta de $v^{-1}(\frac{1}{4})$, $v^{-1}(\frac{1}{2})$ y $v^{-1}(\frac{3}{4})$, obtenemos una lógica 5-valuada.

## 2.4. Sistemas trivaluados

Consideremos ahora una partición $E = V \cup P \cup F$ y en $E$ una conjunción $\cdot$, una disyunción $+$ y una negación en $E$ que actúan como en el caso clásico respecto a $F$ y $V$. Debemos decidir cual ha de ser el comportamiento respecto a $P$. En principio tendremos que preguntarnos por la existencia de una aritmética en un conjunto de tres elementos $\{0, \frac{1}{2}, 1\}$ que refleje el cálculo lógico trivaluado deseado. Si existe, la correspondencia será del tipo $V \to 1$, $P \to \frac{1}{2}$ y $F \to 0$.

Una primera solución es la elección del siguiente conjunto de operaciones :

$$x + y = \text{máx}(x,y); \quad x \cdot y = \text{mín}(x,y); \quad x' = 1 - x$$

que proporcionan el llamado cálculo trivalente *estándar*. Tenemos así las tablas:

| | $'$ | $+$ | 0 | $\frac{1}{2}$ | 1 | $\wedge$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | $\frac{1}{2}$ | 1 | 0 | 0 | 0 | 0 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | $\frac{1}{2}$ | 0 | $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | $\frac{1}{2}$ | 1 |

En este cálculo, puesto que

$$\begin{array}{rcl} 1-\text{mín}\{1-x,1-y\} & = & \text{máx}\{x,y\} \\ 1-\text{máx}\{1-x,1-y\} & = & \text{mín}\{x,y\} \end{array}$$

tenemos las leyes de Morgan

$$\begin{array}{rcl} (a'\cdot b')' & = & a+b \\ (a'+b')' & = & a\cdot b \end{array}$$

y puesto que $1-(1-x)=x$, disponemos también de la ley de la doble negación $a''=a$, es decir, $'$ es fuerte. Así mismo, se dispone de las leyes conmutativas, asociativas, idempotentes y distributivas. $(E,\cdot,+,',0,1)$ es pues un álgebra casi-booleana en la que no valen las leyes:

- Tercero excluido: $a'+a\in V$
- De no contradicción: $a'\cdot a\in F$

ya que $\text{máx}(\frac{1}{2},\frac{1}{2}')=\text{máx}(\frac{1}{2},\frac{1}{2})=\frac{1}{2}$ y $\text{mín}(\frac{1}{2},\frac{1}{2}')=\text{mín}(\frac{1}{2},\frac{1}{2})=\frac{1}{2}$. Estas son las dos únicas leyes del álgebra de Boole que no estarán presentes.

Otra solución es la elección de otro conjunto de operaciones denominados operadores de Łukasiewicz:

$$x+y=\text{mín}(1,x+y);\quad x\cdot y=\text{máx}(0,x+y-1);\quad x'=1-x$$

Tenemos así las tablas:

| | $'$ | $+$ | 0 | $\frac{1}{2}$ | 1 | $\wedge$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | $\frac{1}{2}$ | 1 | 0 | 0 | 0 | 0 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | 1 | $\frac{1}{2}$ | 0 | 0 | $\frac{1}{2}$ |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | $\frac{1}{2}$ | 1 |

En este cálculo, tenemos las leyes de Morgan, la ley de la doble negación y las leyes conmutativas y asociativas pero no las leyes del tercero excluido, la de no contradicción, ni tampoco las leyes idempotentes, ni las leyes distributivas.

En general, considerando una partición de $E$ en $n$ clases, para su modelización numérica, disponemos al menos de los dos sistemas:

$$x+y=\text{máx}(x,y);\quad x\cdot y=\text{mín}(x,y);\quad x'=1-x$$

y

$$x+y=\text{mín}(1,x+y);\quad x\cdot y=\text{máx}(0,x+y-1);\quad x'=1-x$$

De ellos, el primero es el único posible que nos proporciona las leyes idempotentes y distributivas.

### 2.4.1. Razonamiento no monótono

Una de las características del razonamiento formal con el condicional material, es la monotonía: *al aumentar las premisas no se pierden conclusiones*:

> Si $a \rightarrow_V b$, se tiene que $a \cdot c \rightarrow_V b$ para cualquier $c \in E$.

En efecto, $(a \cdot c)' + b = (a' + c') + b = (a' + b) + c' \in V$, por lo tanto $a' + b \in V$. Por otra parte, existen $V$-condicionales no monótonos, basta por ejemplo considerar $\Rightarrow_V = V \times V$. Sin embargo, la mayoría de los sistemas lógicos no monótonos poseen una propiedad de *monotonía restringida*, entendiendo por tal propiedad que satisfacen:

1. Reflexiva: $a \Rightarrow a$ para todo $a \in E$.

2. Monotonía restringida: Si $c \Rightarrow a$ y $c \Rightarrow b$, entonces $c \cdot b \Rightarrow a$.

En el ejemplo anterior, $V \times V$ es un $V$-condicional no monótono. pero sí es monótono en el sentido restringido.

## 2.5. Ejemplos

En la introducción, hemos hecho mención de la dificultad de una clasificación de las lógicas no clásicas. Existe un problema parecido respecto a la caracterización de las lógicas multivaluadas multivaluadas (Avron, 1991). Por esta razón, la elección de los ejemplos es un tanto subjetiva. Nosotros hemos elegido para los tres primeros ejemplos, las lógicas de Łukasiewicz, de Bochvar y de Kleene. La razón, como indica (Turner, 1985), es que son representativas de las diversas lecturas de los valores de verdad no clásicos (intermedios) introducidos por las lógicas multivaluadas.

En relación al comentario anterior, A. Urquhart afirma:

> ...en mi opinión, lo que caracteriza a una lógica multivaluada no es tanto su aparato formal semántico sino las relaciones de los nuevos valores de verdad con su interpretación intuitiva. A este respecto, las ideas básicas que tienen en común los sistemas de Łukasiewicz, Bochvar y Kleene son las siguientes:
>
> - Añaden a los valores de verdad clásicos uno o más valores de verdad con significados como "posible", "sin significado" o "indeterminado". Usualmente, estos valores de verdad son ordenados linealmente.

- Las reglas de asignación de valores de verdad a una fórmula compuesta satisfacen una regla de funcionalidad generalizada; el valor asignado a una fórmula compuesta es una función de los valores asignados a su componente.

### 2.5.1. Lógica trivaluada de Łukasiewicz ($Ł_3$)

Empezamos con la lógica trivaluada de Łukasiewicz ya que es históricamente la primera (junto con la de Post) y es la que más importancia ha tenido en el desarrollo de las lógicas multivaluadas (Byrd, 1979), (Iturrioz, 1976) . Como hemos indicado, la lógica de Łukasiewicz introduce el valor semántico $\frac{1}{2}$ para tratar afirmaciones sobre situaciones que implican incertidumbre y sobre un "futuro abierto" , es decir, afirmaciones que no son verdaderas ni falsas sino indeterminadas en algún sentido. No sólo es que no conozcamos su valor de verdad sino que no lo poseen. Por tanto, el valor semántico $\frac{1}{2}$ denota un "hueco" de verdad.

El sistema original 3-valuado proposicional de Łukasiewicz utiliza las conectivas $\neg$ y $\hookrightarrow$ pensadas para generalizar la implicación material y la negación de la lógica clásica. Las funciones asignadas a estos operadores vienen dadas por:

$$\neg i = 1 - i, \qquad i \hookrightarrow j = \text{mín}\{1, 1 - i + j\}$$

Por lo tanto, sus tablas de verdad son:

| | $\neg$ | $\hookrightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | 1 |
| 1 | 0 | 1 | 0 | $\frac{1}{2}$ | 1 |

Łukasiewicz lee el valor de verdad $\frac{1}{2}$ como "posible". La asignación que provoca reflexión es el valor 1 asignado a $I(A \hookrightarrow B)$ cuando $I(A) = \frac{1}{2}$ e $I(B) = \frac{1}{2}$. Como afirma A. Urquhart en (Urquhart, 1986): "...Desafortunadamente, Łukasiewicz no fue muy explícito sobre este punto crucial...". Él deseaba que $A \hookrightarrow A$ fuera una tautología 3-valuada para toda fbf y hasta aquí todo es aceptable. Sin embargo, es más difícil justificar esta asignación de modo compatible con una lógica de lo "posible": la razón es que la incertidumbre no es veritativo-funcional.

Habitualmente, se consideran además en $Ł_3$ las conectivas $\vee, \wedge$ y $\leftrightarrow$ con $\vee$ y $\wedge$ como en el cálculo estándar y con la siguiente función asignada a $\leftrightarrow$:

$i \leftrightarrow j = \text{mín}\{\text{mín}\{1, 1 - i + j\}, \text{mín}\{1, 1 - j + i\}\}$ para todo $i, j \in \mathbf{3}$

Por tanto, la tabla para $\leftrightarrow$ es:

| $\leftrightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 1 | $\frac{1}{2}$ | 0 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | $\frac{1}{2}$ |
| 1 | 0 | $\frac{1}{2}$ | 1 |

Estas conectivas pueden ser introducidas como conectivas definidas ya que para todo $i, j \in \mathbf{3}$ se tiene que

$i \vee j = (i \hookrightarrow j) \hookrightarrow j$

$i \wedge j = \neg(\neg i \hookrightarrow \neg j)$

$i \leftrightarrow j = (i \hookrightarrow j) \wedge (j \hookrightarrow i)$

En términos de álgebras abstractas: La lógica trivaluada de Łukasiewicz viene dada por un par $(\mathcal{L}, \mathcal{M})$ en el que el álgebra $\mathcal{L} = (Q, \neg, \hookrightarrow, \vee, \wedge, \leftrightarrow)$ tiene un tipo de similaridad $\langle 1, 2, 2, 2, 2\rangle$ y tiene una matriz asociada $\mathcal{M} = (\mathbf{3}, 1, \neg, \hookrightarrow, \vee, \wedge, \leftrightarrow)$.

Esta lógica fue axiomatizada en 1931 por M. Wajsberg como sigue:

**Axiomas**

1. $A \hookrightarrow (B \hookrightarrow A)$
2. $(A \hookrightarrow B) \hookrightarrow ((B \hookrightarrow C) \hookrightarrow (A \hookrightarrow C))$
3. $(\neg B \hookrightarrow \neg A) \hookrightarrow (A \hookrightarrow B)$
4. $((A \hookrightarrow \neg A) \hookrightarrow A) \hookrightarrow A$

**Regla de Inferencia**

La única regla de inferencia es la regla Modus Ponens:

$$\frac{A, \quad A \hookrightarrow B}{\text{B}}$$

Este sistema es correcto y completo. Sin embargo, $Ł_3$ no es funcionalmente completo. En 1936 Jerzy Słupecki extendió $Ł_3$ añadiendo la conectiva monaria $T$ definida por la tabla:

| | $T$ |
|---|---|
| 0 | $\frac{1}{2}$ |
| $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | $\frac{1}{2}$ |

y regido por los axiomas:

5) $T\,A \hookrightarrow \neg T\,A$

6) $\neg T\,A \hookrightarrow T\,A$

La lógica resultante, conocida como lógica de Łukasiewicz-Słupecki, denotada $Ł_3S$, es funcionalmente completa.

Otras axiomatizaciones para $Ł_3$ pueden verse en (Tokarz, 1974), (Iturrioz, 1977)

### Lógica n-valuada de Łukasiewicz ($Ł_n$)

Łukasiewicz generalizó su lógica 3-valuada a $n$ valores así como a un sistema infinito-valuado en 1922. El propio Łukasiewicz expresó su preferencia filosófica por la lógica infinito valuada.

La lógica $n$-valuada de Łukasiewicz sigue fielmente lo expuesto para el caso $n = 3$ y sólo hay que cambiar el conjunto de valores semánticos $S = \mathbf{n}$, así, vendrá dada por un par $(\mathcal{L}, \mathcal{M})$ en el que el álgebra $\mathcal{L} = (Q, \neg, \hookrightarrow, \vee, \wedge, \leftrightarrow)$ tiene un tipo de similaridad $\langle 1, 2, 2, 2, 2\rangle$ y tiene una matriz asociada $\mathcal{M} = (\mathbf{n}, 1, \neg, \hookrightarrow, \vee, \wedge, \leftrightarrow)$, donde estos últimos operadores vienen dados por: $\neg i = 1 - i$, $i \hookrightarrow j = \text{mín}\{1, 1 - i + j\}$, $i \vee j = \text{máx}\{i, j\}$, $i \wedge j = \text{mín}\{i, j\}$ y $i \leftrightarrow j = \text{mín}\{\text{mín}\{1, 1 - i + j\}, \text{mín}\{1, 1 - j + i\}\}$, con $i, j \in \mathbf{n}$.

Introducimos ahora las conectivas monarias $J_0, J_{\frac{1}{n-1}} \ldots, J_1$, llamadas *afirmaciones de valores de verdad*, cuya semántica viene dada por

$$I(J_k(A)) = \begin{cases} 1 & \text{si } I(A) = k \\ 0 & \text{si } I(A) \neq k \end{cases}$$

**Lema 2.1** *Para todo $k$ tal que $k \in \mathbf{n}$, se tiene que $J_k$ es definible en $Ł_n$.*

## 2.5.2. La lógica de Bochvar

El trabajo del lógico ruso D.A. Bochvar (Bochvar, 1939) representa una nueva motivación filosófica para las lógicas multivaluadas; él las utiliza como un modo de evitar las paradojas semánticas.

Para tratar con enunciados tales como: "Esta frase es falsa" (Paradoja del mentiroso), Bochvar adopta una estrategia que supone una lógica diferente. Según Bochvar, tales afirmaciones no son ni verdaderas ni falsas sino paradójicas o "sin significado" y a ellas se les asocia un tercer valor semántico $\frac{1}{2}$. Estas ideas condujeron a Bochvar a introducir tres lógicas 3-valuadas, denotadas $\mathcal{B}_3^1$, $\mathcal{B}_3^2$ y $\mathcal{B}_3^3$. Sus matrices son:

- $\{\{0, \frac{1}{2}, 1\}, \{1\}, \neg, \wedge, \vee, \rightarrow\}$ para $\mathcal{B}_3^1$ y $\mathcal{B}_3^3$.
- $\{\{0, \frac{1}{2}, 1\}, \{\frac{1}{2}, 1\}, \neg, \wedge, \vee, \rightarrow\}$ para $\mathcal{B}_3^2$.

Las tablas de verdad para $\mathcal{B}_3^1$ y $\mathcal{B}_3^2$ son tales que el valor semántico $\frac{1}{2}$ es, en algún sentido, "infeccioso": en la definición de las conectivas, si alguna componente toma el valor $\frac{1}{2}$, el total toma el valor $\frac{1}{2}$ y, cuando intervienen únicamente los valores 1 y 0, el valor asignado es el mismo que en la lógica clásica. Así, por ejemplo, las tablas para $\neg$ y para $\wedge$ son:

| | $\neg$ | $\wedge$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | $\frac{1}{2}$ | 0 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 | 1 | 0 | $\frac{1}{2}$ | 1 |

Por su parte, las tablas para $\mathcal{B}_3^3$ son:

| | $\neg$ | $\wedge$ | 0 | $\frac{1}{2}$ | 1 | $\vee$ | 0 | $\frac{1}{2}$ | 1 | $\rightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| $\frac{1}{2}$ | 1 | $\frac{1}{2}$ | 0 | 0 | 0 | $\frac{1}{2}$ | 0 | 0 | 1 | $\frac{1}{2}$ | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

En $\mathcal{B}_3^1$ no existen tautologías y en $\mathcal{B}_3^2$ y $\mathcal{B}_3^3$ el conjunto de tautologías coincide con el conjunto de tautologías de la lógica clásica proposicional así como los sistemas axiomáticos.

Las ideas de Bochvar fueron extendidas por S. Halldén (Halldén, 1949) quien formalizó una versión de la lógica proposicional 3-valuada que contiene, además de las conectivas proposicionales usuales, una conectiva monaria $S$. $S\,A$ se lee como "$A$ es una proposición significativa" y toma el valor 1 si $A$ toma el valor 1 o el valor 0, y toma el valor 0 si $A$ toma el valor $\frac{1}{2}$. Una breve descripción de esta lógica puede verse en (Bolc y Borowick, 1992).

El trabajo de Halldén fue extendido por Goddard y Routley (Goddar y Routley, 1973) incluyendo no sólo un análisis sintáctico y semántico de las lógicas proposicionales de la significación sino también formalizaciones de las extensiones a primer orden y a orden superior.

En la lógica introducida en (Delahaye y Thibau, 1991), la implicación externa $\rightarrow$ de Bochvar se usa entre la cabeza y el cuerpo de las cláusulas en los programas lógicos.

### 2.5.3. Lógica trivaluada fuerte de Kleene ($KF_3$)

En 1938, S.C. Kleene introdujo una nueva lógica 3-valuada (Kleene, 1938), (Kleene, 1952). Su motivación fue la formalización de la "ignorancia parcial"; una de las más claras interpretaciones de la lógica de Kleene emplea alguna forma de metáfora epistemológica (lo sé o no): A un enunciado se le asigna el tercer valor semántico, $\frac{1}{2}$, si "no se conoce si es verdadero o falso", pero es ciertamente verdadero o falso (pensemos en un detective que ha de pronunciarse sobre si alguien es o no un asesino). Específicamente, Kleene estaba interesado en la teoría de las funciones recursivas. En dicha teoría, si pensamos en una máquina diseñada para responder "verdadero" o "falso" a ciertas cuestiones, para algunas entradas la máquina será incapaz de proporcionar una respuesta, bien porque entra en un bucle infinito o porque ha agotado su capacidad computacional. En este caso podemos pensar que la respuesta de la máquina es "indefinido" o $\frac{1}{2}$.

De hecho, Kleene introdujo dos lógicas trivaluadas que son conocidas en la bibliografía actual como lógica fuerte y débil respectivamente

En términos de álgebras abstractas, la lógica trivaluada fuerte de Kleene viene dada por un par $(\mathcal{L}, \mathcal{M})$ en el que el álgebra $\mathcal{L} = (Q, \neg, \rightarrow, \vee, \wedge, \leftrightarrow)$ tiene un tipo de similaridad $\langle 1, 2, 2, 2, 2\rangle$ y tiene una matriz asociada $\mathcal{M} = (\mathbf{3}, 1, \neg, \rightarrow, \vee, \wedge, \leftrightarrow)$, los operadores $\neg, \wedge$ y $\vee$ se definen como en el cálculo estándar y los restantes se definen mediante las tablas:

| $\rightarrow$ | 0 | $\frac{1}{2}$ | 1 | $\leftrightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | $\frac{1}{2}$ | 0 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 | $\frac{1}{2}$ | 1 | 1 | 0 | $\frac{1}{2}$ | 1 |

En esta lógica el conjunto de tautologías es vacío.

En la lógica introducida en (Delahaye y Thibau, 1991) la implicación material $\rightarrow$ de Kleene se usa en el cuerpo de las cláusulas en los programas lógicos.

### 2.5.4. Lógica trivaluada débil de Kleene ($KD_3$)

Esta lógica viene dada por un par $(\mathcal{L}, \mathcal{M})$ en el que el álgebra $\mathcal{L} = (Q, \neg, \rightarrow, \vee, \wedge, \leftrightarrow, \Leftrightarrow)$ tiene un tipo de similaridad $\langle 1, 2, 2, 2, 2\rangle$ y tiene una matriz asociada $\mathcal{M} = (\mathbf{3}, 1, \neg, \rightarrow, \vee, \wedge, \leftrightarrow, \Leftrightarrow)$, los operadores $\neg, \vee$ y $\wedge$ se definen como en el cálculo estándar, $\rightarrow$ y $\leftrightarrow$ mediante las tablas

| $\rightarrow$ | 0 | $\frac{1}{2}$ | 1 | $\leftrightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | $\frac{1}{2}$ | 0 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | 1 | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | $\frac{1}{2}$ |
| 1 | 0 | $\frac{1}{2}$ | 1 | 1 | 0 | $\frac{1}{2}$ | 1 |

y el operador añadido $\Leftrightarrow$ se define según la tabla siguiente :

| $\Leftrightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| $\frac{1}{2}$ | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |

Otros ejemplos de lógicas trivaluadas, tales como la lógica de H. Reichenbach (motivada por algunos aspectos filosóficos de la mecánica cuántica), la lógica de J. Słupecki (con motivaciones puramente lógicas) y la lógica de B. Sobociński (con $D = \{\frac{1}{2}, 1\}$, la lógica LPF (extensión de la lógica de Kleene desarrollada en el proyecto VDM (H. Barringer y Jones, 1984), (Jones, 1986)) y la lógica $\mathrm{RM}_3$ (la más fuerte de las lógicas de la relevancia) pueden verse en (Bolc y Borowick, 1992), (Avron, 1991), (Anderson y N.D.Belnap, 1975) y (Dunn, 1986).

Introducimos por último un ejemplo más, tanto por su interés histórico como por su utilidad teórica: l lógica de Post.

### 2.5.5. Lógica de Post

Como hemos indicado, independientemente de Łukasiewicz, E. Post da en (Post, 1920) una lógica $n$-valuada que utiliza como conectivas primitivas una conectiva monaria $\sigma$ y una binaria $\vee$; la conectiva $\vee$ es la disyunción estándar y la conectiva monaria $\sigma$ es un "desplazamiento ascendente cíclico de los valores de verdad":

$$\sigma i = \ \text{mín}\,\{1, i + \tfrac{1}{n-1}\} \quad \text{para todo } i \in \mathbf{n}$$

El destacado trabajo de Post incluye no sólo una demostración de que su lógica es funcionalmente completa, sino además un método general para obtener una axiomatización completa del sistema con

$$S = \ \mathbf{n} \text{ y } D = \{\frac{n-m}{n-1}, \frac{n-m+1}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$$

**Teorema 2.11** *La lógica n-valuada de Post es funcionalmente completa*

Una demostración de este teorema puede verse en (Urquhart, 1986).

## 2.6. Aplicaciones de las lógicas multivaluadas

Terminamos este capítulo con una breve referencia a las aplicaciones de las lógicas multivaluadas.

Las lógicas multivaluadas, desde que en 1920 fueron formuladas por primera vez por Łukasiewicz, fueron asociadas a problemas de indefinición, incompletitud y no terminación. En particular, las lógicas $n$-valuadas con $n > 3$ fueron en sus orígenes un producto de especulaciones más o menos formales. Sin embargo, ciertas lógicas 3-valuadas fueron diseñados con propósitos muy concretos. Entre ellas cabe destacar las introducidas para resolver ciertos problemas filosóficos en conexión con la mecánica cuántica. Estos problemas fueron objeto de estudio por parte de G. Birkhoff y J. Von Neumann (Birkhoff y von Neumann, 1936), y posteriormente por H.Reinchenbach (Reichenbach, 1962), (Reichenbach, 1964). En los trabajos de Słupecki, Sobocinski y Webb (Słupecki, 1972) el cálculo trivaluado se usa para analizar y decidir ciertas cuestiones de naturaleza puramente lógica.

Por su parte, la lógica 3-valuada de S. C. Kleene, introducida en (Kleene, 1952) como herramienta para cuestiones relativas a propiedades de terminación de funciones recursivas, dio lugar al uso de las lógicas multivaluadas en el estudio de los lenguajes de programación.

Las lógicas multivaluadas en general y las trivaluadas en particular han adquirido especial relevancia en los últimos años, tanto por su interés en sí mismo (Hodes, 1989), como por su potencial aplicabilidad en diversas áreas en Ciencias de la Computación.

Entre las áreas de las Ciencias de la Computación interesadas en las lógicas multivaluadas cabe destacar el área de verificación, especificación y síntesis de programas ( (Jones, 1986), (Blikle, 1991), (Fitting, 1986), (Fitting, 1990a), (Kunen, 1987), (Delahaye y Thibau, 1991), (Baaz y Zach, 1992), (Przymusinski, 1991)). En particular, H.Rasiowa (Rasiowa, 1974) y B. Kirkernd han destacado la aplicabilidad de las lógicas 3-valuadas en el área de corrección de programas y son ya numerosos los trabajos sobre semántica de los lenguajes de programación en los que se hace uso de tales lógicas.

Otra de las áreas que más reclaman en la actualidad la consideración de lógicas multivaluadas es la Inteligencia Artificial ( (Turner, 1985), (Colmerauer y Pique, 1981), (de Bessonet, 1991)). Como indica Sombé, las bases de conocimiento se construyen a partir de las creencias de los expertos y la modificación de estas creencias es una actividad esencial tanto para el hombre como para todo sistema evolutivo. Dada una base de conocimiento, podemos señalar tres tipos destacados de modificaciones:

- *Expansión:* Debido a la necesidad de introducir nueva información que no modifica la información existente.
- *Revisión:* Al introducir nueva información que modifica la información existente.
- *Contracción:* Al suprimir información existente.

Estas modificaciones recogen estados de *ignorancia parcial* que ponen de manifiesto las limitaciones de la lógica clásica y que encuentran en las lógicas multivaluadas una herramienta de gran utilidad.

La aplicación de lógicas multivaluadas al tratamiento de información incompleta ha sido estudiada por el profesor Mundici, relacionándolas con el tratamiento en la recepción de señales con ruido. Ligada a esta aplicación, en (Mundici, 1996), (Mundici, 1990) y (Mundici, 1992) puede verse el paralelismo entre las lógicas multivaluadas y el juego de Ulam. Otras aplicaciones pueden encontrarse en (Mundici, 1993) y (Mundici, 1991).

Los libros (Hähnle, 1993) y (Bolc y Borowick, 1992) recogen variedad de ejemplos de aplicaciones, tanto de naturaleza teórica como de naturaleza práctica. Estas últimas requieren avances significativos en el área de demostración automática en el que se enmarca este trabajo.

# Capítulo 3

# Sistemas de demostración

Dada una lógica (es decir, un lenguaje y una teoría de modelos) existen, en general, diversos sistemas de demostración que podemos clasificar en:

- Sistemas de deducción.
- Sistemas de refutación.

En un sistema de deducción, la verificación de una fórmula o de una inferencia se realiza directamente, mediante lo que podríamos llamar mecanismos de *razonamiento hacia adelante* que conducen finalmente a la fórmula a verificar (o van de las hipótesis a la conclusión en el caso de la verificación de una inferencia). Este es el caso de los sistemas tipo Hilbert y los sistemas tipo Gentzen para los que, en el caso de las lógicas multivaluadas existe una amplia bibliografía (ver por ejemplo (Bolc y Borowick, 1992), (Urquhart, 1986)).

En un sistema de refutación, por el contrario, se realiza un *razonamiento hacia atrás* haciendo uso de la conexión entre los conceptos semánticos de validez e insatisfacibilidad:

$$A \text{ es válida si y sólo si } \neg A \text{ es insatisfacible}$$

$$\Omega \models A \text{ si y sólo si } \Omega \cup \{\neg A\} \text{ es insatisfacible}$$

y se verifica la fórmula (o la inferencia), intentando "refutar" la negación de la fórmula (o el conjunto de las hipótesis y la negación de la conclusión).

La mayoría de los sistemas de refutación, en claro contraste con los sistemas de deducción, son procedimientos automáticos.

Hasta nuestros días, la mayor parte de los trabajos sobre demostración automática de teoremas se basan en dos métodos de refutación:

1. El método de construcción de modelos, basados en trabajos de Gentzen (Gentzen, 1935).

2. El método de resolución, debido a Robinson (Robinson, 1965) basado en técnicas desarrolladas por Herbrand (Herbrand, 1967).

El cálculo de secuentes de Gentzen supuso la primera ocasión en que se utilizó la relación de *consecuencia lógica* como un cálculo. Su sistema $LK^{-}$ (Gentzen, 1935) ha dado lugar a diversos sistemas de demostración para diversas lógicas, los cuales se conocen globalmente como *tablas semánticas* ( (Oppacher y Suen, 1988), (Carnielli, 1987), (Fitting, 1983)) y cuya primera formulación se debe a Beth en 1955 ( (Beth, 1959), (Beth, 1967)). Otras formulaciones pueden encontrarse en (Smullyan, 1968), (Jeffrey, 1967), (Gallier, 1987), (Oppacher y Suen, 1988), (Fitting, 1994), (Fitting, 1990b), (Hähnle y Schmitt, 1994).

Los trabajos de Herbrand y Gentzen están estrechamente relacionados, pero su desarrollo y aplicación han sido muy diferentes. Los ATPs basados en las tablas semánticas ( (Bibel, 1982), (Andrews, 1981)) son comparativamente muy escasos respecto a los basados en resolución, y ello es debido, básicamente, a los trabajos de Kowalski (Kowalski, 1974) y Colmerauer (et al, 1973), los cuales impulsaron en la década de los 70 la idea de que la Lógica puede ser usada como un lenguaje de programación. Esta idea fue revolucionaria a causa de que, hasta 1972, la Lógica sólo había sido usada como lenguaje declarativo de especificación.

En el caso de las lógicas multivaluadas, al igual que para las demás, la mayoría de los demostradores existentes también están basados en diversas versiones de la regla de resolución. Como hemos indicado en la introducción, un análisis de algunos métodos basados en resolución para las lógicas multivaluadas puede verse en (Hähnle, 1993).

El nuevo método introducido en este trabajo, al igual que las tablas semánticas, está basado en los trabajos de Gentzen. Por esta razón, incluimos aquí una breve visión de los métodos más destacados en esta línea.

Damos a continuación un breve resumen del método de las tablas semánticas para la lógica clásica proposicional

## 3.1. El método de las tablas semánticas

Como sistema de refutación, para verificar la validez de una fórmula $A$, se determina si $\neg A$ es insatisfacible. Para ello se organiza la búsqueda sistemática de un modelo para $\neg A$. Si la búsqueda tiene éxito, la fórmula $A$ no es válida y si la búsqueda fracasa, la fórmula $A$ es válida.

En síntesis, el método puede ser descrito como sigue:

Dada la fórmula $A$ cuya validez se quiere comprobar, se toma como punto de partida un árbol con $\neg A$ como único nodo

Este árbol, denominado *árbol inicial asociado a* $\neg A$ se irá ampliando sucesivamente mediante ciertas reglas de extensión, basadas únicamente en la estructura sintáctica de las fórmulas. Cada árbol así obtenido se denomina *árbol asociado a* $A$. Las reglas de extensión son diseñadas de modo que:

- La aplicación de cada una de dichas reglas genera un árbol binario cuyos nodos están etiquetados con fbfs.
- Cada regla introduce fbfs que son subfórmulas propias de fbfs existentes en el árbol al que se aplica.

Para la descripción del método utilizaremos la *notación uniforme* debida a Smullyan (Smullyan, 1968), que consiste en agrupar las fórmulas por tipos, consiguiendo con ello una notable simplificación en la exposición del método. Concretamente, las fbfs se clasifican en: *literales*, de *tipo* $\alpha$ (o de comportamiento conjuntivo) y de *tipo* $\beta$ (o de comportamiento disyuntivo). Cada fórmula de tipo $\alpha$ o de tipo $\beta$ tiene asociada dos *componentes*, denotadas $\alpha_1, \alpha_2$ y $\beta_1, \beta_2$ respectivamente. Las siguientes tablas muestran las fórmulas de tipo $\alpha$ y de tipo $\beta$ junto con sus componentes.

| $\alpha$ | $\alpha_1$ | $\alpha_2$ |
|---|---|---|
| $A \wedge B$ | $A$ | $B$ |
| $\neg(A \vee B)$ | $\neg A$ | $\neg B$ |
| $\neg(A \rightarrow B)$ | $A$ | $\neg B$ |
| $\neg\neg A$ | $A$ | $A$ |

| $\beta$ | $\beta_1$ | $\beta_2$ |
|---|---|---|
| $A \vee B$ | $A$ | $B$ |
| $\neg(A \wedge B)$ | $\neg A$ | $\neg B$ |
| $A \rightarrow B$ | $\neg A$ | $B$ |

**Reglas de Extensión**

El árbol inicial asociado a una fbf $\neg A$, denotado $T_{\neg A}$, es extendido sucesivamente, para obtener árboles asociados a $\neg A$, mediante las siguientes reglas:

($\alpha$) Si $\rho_B$ denota la rama determinada por el nodo hoja $B$ y una $\alpha-$fórmula ocurre en $\rho_B$, extendemos dicha rama añadiendo dos nodos etiquetados con sus componentes $\alpha_1$, $\alpha_2$ respectivamente (o un nodo etiquetado con la componente común, si $\alpha_1 \neq \alpha_2$).

($\beta$) Si $\rho_B$ denota la rama determinada por el nodo hoja $B$ y una $\beta-$fórmula ocurre en $\rho_B$, extendemos dicha rama añadiendo dos nodos etiquetados con sus componentes $\beta_1$ y $\beta_2$ respectivamente.

**Definición 3.1** Sea $A$ una fbf. Un árbol $T$ se dice que es un *árbol para* $\neg A$, si existe una secuencia de árboles $\mathrm{T}_1, \ldots, \mathrm{T}_n$ tal que:

- $\mathrm{T}_1$ es el árbol inicial asociado a $\neg A$.
- Cada árbol $\mathrm{T}_i$ $(2 \leq i \leq n)$ es un árbol asociado a $\neg A$ que es *extensión inmediata* de $\mathrm{T}_{i-1}$, es decir, $\mathrm{T}_i$ se obtiene de $\mathrm{T}_{i-1}$ por aplicación de una regla de extensión a uno de sus nodos.
- $\mathrm{T}_n = T$.

**Definición 3.2** Sea $T$ un árbol para $\neg A$. Una rama de $T$ se dice *cerrada*, si en ella ocurren una fórmula $B$ y su negación $\neg B$. En caso contrario, se dice que la rama es *abierta*. El árbol $T$ se dice *cerrado* si todas sus ramas son cerradas.

Una rama $\rho$ de un árbol $T$ para $\neg A$ se dice *completa* si satisface las siguientes condiciones:

1. Si la fórmula $\alpha$ ocurre en $\rho$, también sus componentes $\alpha_1$ y $\alpha_2$ ocurren en $\rho$.
2. Si la fórmula $\beta$ ocurre en $\rho$ entonces, o la componente $\beta_1$ o la componente $\beta_2$ ocurre en $\rho$.

Un árbol $T$ para $\neg A$ se dice *terminado* si toda rama es cerrada o completa.

**Teorema 3.1** *Para toda fbf, A, mediante un número finito de extensiones inmediatas se obtiene un árbol terminado para* $\neg A$.

**Definición 3.3** Dada una fbf $A$, una *refutación* para $\neg A$ es un árbol cerrado para $\neg A$. Una fórmula $A$ se dice *demostrable* si existe una refutación para $\neg A$.

**Definición 3.4** Un árbol $T$ para $\Omega$ se dice *satisfacible* si el conjunto de las fbfs que etiquetan los nodos de alguna de las ramas de $T$ es satisfacible. Una rama tal se dice que es *satisfacible*.

**Teorema 3.2 (de existencia de modelo)** *Toda rama completa y abierta* $\rho$ *de un árbol* $T$ *para* $\neg A$ *es satisfacible.*

El teorema de existencia de modelos asegura que en un árbol terminado para $\neg A$, cada rama abierta $\rho$, de existir, proporciona un modelo para $\neg A$. Concretamente, toda interpretación $I$ tal que $I(p_i) = 1$ si $p_i$ ocurre en $\rho$ e $I(p_i) = 0$ si $\neg p_i$ ocurre en $\rho$.

Vemos pues que no es necesario que el modelo generado asigne valores de verdad específicos a todos los símbolos proposicionales que intervienen en $\Omega$ y que ramas distintas pueden generar el mismo modelo.

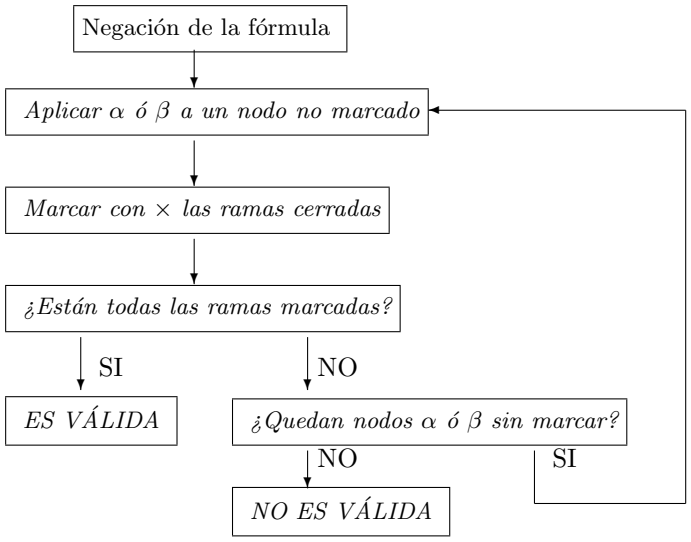
**Teorema 3.3 (Corrección y completitud)** *El método de las tablas semánticas es correcto y completo, es decir, una fbf es válida si y sólo si existe una refutación para* $\neg A$.

### Descripción del método

En la construcción de un árbol para refutar un conjunto de fórmulas, se tienen en cuenta las siguientes consideraciones:

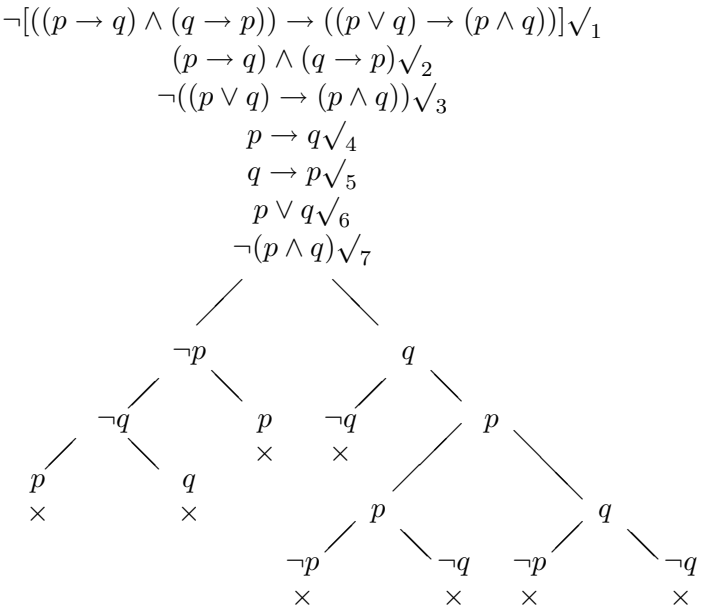
- Para evitar ramificaciones innecesarias, se da prioridad a la regla ($\alpha$).
- La regla ($\alpha$) o ($\beta$) se aplica sobre cada nodo una sola vez. Para indicar que la $\alpha$-fórmula o $\beta$-fórmula correspondiente no será usada de nuevo, tras su aplicación, se marca el nodo con $\surd$.
- No se realiza ninguna extensión sobre las ramas cerradas. Para indicar que una rama es cerrada se marca con $\times$.

Damos a continuación un diagrama de flujo para la construcción de un árbol de refutación:



En el siguiente ejemplo hacemos uso del método para probar la validez de una fórmula

**Ejemplo 3.1** La fórmula $((p \to q) \wedge (q \to p)) \to ((p \vee q) \to (p \wedge q))$ es válida.




## 3.2. Método de los Árboles Finitamente Generados

La generalización a las lógicas $n$-valuadas proposicionales del método de las tablas de Beth se debe a S.J. Surma (Surma, 1984) y W. Suchon (Suchoń, 1974) que lo introdujeron independientemente en 1974; su extensión a las lógicas $n$-valuadas de primer orden se debe a Walter A. Carnielli (Carnielli, 1985), (Carnielli, 1987).

En esta sección presentamos brevemente la generalización para el caso proposicional.

Sea $\mathcal{L} = (Q, *_1, *_2, \ldots, *_k)$ un lenguaje para una lógica proposicional $n$-valuada y $\mathcal{M} = (\mathbf{n}, d_1, d_2, \ldots, d_m, *'_1, *'_2, \ldots, *'_k)$ una matriz $n$-valuada para $\mathcal{L}$. Suponemos que disponemos de las $n$ conectivas monarias de *afirmaciones de valores de verdad*: $J_0, J_{\frac{1}{n-1}} \ldots, J_1$, y que las fbfs están etiquetadas con una etiqueta $J_i$ para algún $i \in \mathbf{n}$.

Sea $*$ una conectiva $m$-aria del lenguaje. La regla de eliminación de la conectiva $*$ en una fbf del tipo $J_i * (A_1, \ldots, A_m)$, denotada $(*, i)$ se define como sigue:

$$\frac{J_i * (A_1, \ldots, A_m)}{\bigvee\{\bigwedge_{k=1}^{t} J_{j_k} A_{i_k} \mid j_k \in \mathbf{n}, t \leq m \text{ y } H_i(*, j_1 \ldots, j_t)\}}$$

donde $H_i(*, j_1 \ldots, j_t)$ denota la siguiente condición:

Existe un homomorfismo $\phi : \mathcal{L} \to \mathcal{M}$ con las propiedades siguientes:

- $\phi(A_{i_k}) = v_k$ para $1 \leq k \leq t$.
- Si $*'$ es el operador $m$-ario sobre $\mathbf{n}$ que es la interpretación de $*$, entonces

$$*'(v_1 \ldots, v_{i_1}, \ldots, v_{i_2}, \ldots, v_{i_k}, \ldots, v_m) = i$$

- $t$ es el menor índice que satisface las dos condiciones anteriores.

**Ejemplo 3.2** Consideremos $\mathbf{n} = \mathbf{3}$ y sea $*$ definida por la tabla

| $*$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | 1 |
| 1 | 0 | $\frac{1}{2}$ | 1 |

Entonces,

$$\frac{J_0(A * B)}{J_1 A \wedge J_0 B}$$

$$\frac{J_{\frac{1}{2}}(A * B)}{(J_{\frac{1}{2}} A \wedge J_0 B) \vee (J_1 A \wedge J_{\frac{1}{2}} B)}$$

$$\frac{J_1(A * B)}{(J_0 A \wedge J_0 B) \vee (J_0 A \wedge J_{\frac{1}{2}} B) \vee \ldots \vee (J_{\frac{1}{2}} A \wedge J_1 B) \vee (J_1 A \wedge J_1 B)}$$

Definidas las reglas $(*, i)$, se llama *árbol asociado* a la fórmula $J_i*(A_1, \ldots, A_m)$ al árbol construido como sigue:

La fórmula $J_i*(A_1, \ldots, A_m)$ es la raíz, y la construcción continúa del siguiente modo:

- Los sucesores inmediatos de la raíz son las conclusiones de la regla $(*, i)$ aplicada a la fórmula raíz.

- En cada etapa de construcción se procede como sigue:

  - Si todas las hojas del árbol son fórmulas atómicas, se finaliza la construcción del árbol.
  - En otro caso, elegimos algún nodo $N$ que es una fórmula no atómica. Si esta fórmula es de la forma $J_jB$, se añaden en cada una de las hojas descendientes de $N$ las conclusiones de la regla $(*, j)$ aplicada a $J_jB$ y se marca $N$ como usado.

**Definición 3.5** El árbol asociado a $J_i * (A_1, \ldots, A_m)$ es cerrado si todas las ramas son cerradas, es decir, si en cada rama existen dos nodos $J_ip$ $J_kp$ con $i \neq k$, o bien existe un nodo no marcado para el que no es aplicable ninguna regla. En caso contrario, el árbol es abierto.

Obviamente, se tiene que

- Para todo átomo $p \in Q$ los árboles asociados a las fórmulas $J_ip$ son abiertos.
- El árbol asociado a una fórmula $J_i * (A_1, \ldots, A_m)$ es cerrado si es falsa la hipótesis de que existe un modelo para tal fórmula.

**Teorema 3.4** *Una fbf $A$ es válida si y sólo si todos los árboles asociados a $J_i * (A_1, \ldots, A_m)$ para $i \notin D$ son cerrados.*

**Ejemplo 3.3** Probemos que la fórmula $p \hookrightarrow (q \hookrightarrow p)$ es válida en la lógica trivaluada de Łukasiewicz con conectivas $\neg$ y $\hookrightarrow$, matriz $(\mathbf{3}, \{1\}, \neg, \hookrightarrow)$ y tablas de verdad:

| | $\neg$ | $\hookrightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 | 1 |
| 1 | 0 | 1 | 0 | $\frac{1}{2}$ | 1 |

Las reglas de expansión para las fórmulas $J_i(A \hookrightarrow B)$ son:

$$\frac{J_0(A \hookrightarrow B)}{J_1A \wedge J_0B}$$

$$\frac{J_{\frac{1}{2}}(A \hookrightarrow B)}{(J_{\frac{1}{2}}A \wedge J_0B) \vee (J_1A \wedge J_{\frac{1}{2}}B)}$$

$$\frac{J_1(A \hookrightarrow B)}{\bigvee\{(J_iA \wedge J_kB) \mid i \leq k; i,k \in \mathbf{3}\}}$$

Las reglas de expansión para las fórmulas $J_i\ \neg A$ son:

$$\frac{J_0\neg A}{J_1A} \qquad \frac{J_{\frac{1}{2}}\neg A}{J_{\frac{1}{2}}A} \qquad \frac{J_1\neg A}{J_0A}$$

El árbol asociado a la fórmula $J_0(p \hookrightarrow (q \hookrightarrow p))$ es el árbol cerrado de la figura **??**

$$\begin{array}{c} J_0(p \hookrightarrow (q \hookrightarrow p)) \\ \downarrow \\ J_1p \\ \downarrow \\ J_0(q \hookrightarrow p) \\ \downarrow \\ J_1q \\ \downarrow \\ J_0p \\ \times \end{array}$$

Figura 3.1: El árbol $J_0(p \hookrightarrow (q \hookrightarrow p))$.

El árbol asociado a la fórmula $J_0(p \hookrightarrow (q \hookrightarrow p))$ es el árbol cerrado de la figura **??**

**Definición 3.6** Un conjunto finito o numerable de fórmulas etiquetadas $\Omega$ es un *conjunto de Hintikka* si satisface las siguientes condiciones

1. Para todo símbolo proposicional $p \in Q$, si $J_ip \in \Omega$ y $k \neq i$ se tiene que $J_kp \notin \Omega$.
2. Si $A = J_i * (A_1, \ldots, A_m) \in \Omega$ entonces al aplicar la regla $(*, i)$ a $A$ se tiene que al menos una de las conclusiones pertenece a $\Omega$.

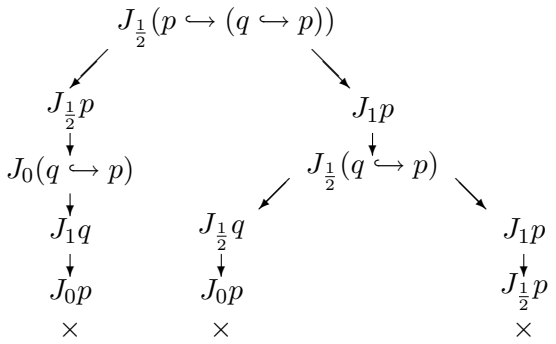



Figura 3.2: El árbol $J_{\frac{1}{2}}(p \hookrightarrow (q \hookrightarrow p))$.

Un conjunto de Hintikka $\Omega$ se dice *saturado* si satisface además de las condiciones 1 y 2 las condiciones siguientes:

3. Para todo símbolo proposicional $p \in Q$, existe $i \in \mathbf{n}$ tal que $J_i\, p \in \Omega$.

1. Si $A = J_i * (A_1, \ldots, A_m)$ entonces $A \in \Omega$ si y sólo si al aplicar la regla $(*, i)$ a $A$ se tiene que al menos una de las conclusiones pertenece a $\Omega$.

**Lema 3.1** *Todo conjunto de Hintikka puede ser extendido a un conjunto de Hintikka saturado.*

**Teorema 3.5** *Para todo conjunto de Hintikka saturado $\Omega$ existe una interpretación $I$ tal que $I(A) = i \in \mathbf{n}$ si y sólo si $A \in \Omega$; en particular, si existe un árbol abierto para $J_i A$, entonces existe una valuación $I$ tal que $I(A) = i$.*

Como destaca R. Hähnle en (Hähnle, 1993), no son pocos los obstáculos para que el método expuesto pueda ser usado en un demostrador de teoremas con la mínima eficiencia requerida. Incluso al nivel proposicional que nos ocupa, podemos destacar entre otros los siguientes:

- El número de árboles que es necesario construir es $\mathbf{n} - |D|$ y por lo tanto, en general, $\mathcal{O}(n)$, ya que $D$ es pequeño.
- El número medio de ramas generadas al aplicar una regla es $\frac{n^k}{n} = n^{k-1}$ y el número de fórmulas nuevas para conectivas de aridad $k$ es $\frac{kn^k}{n} = kn^{k-1}$.
- Como se puede observar en los árboles de las figuras **??** y **??**, si consideramos las fórmulas que etiquetan los nodos eliminando las conectivas monarias $J_i$,

todas las fórmulas obtenidas en el primer árbol se repiten en el segundo y en la misma posición. El análisis de otros ejemplos muestra que existe siempre un alto grado de redundancia en los árboles correspondiente a los diversos valores no destacados.

## 3.3. Conjuntos de valores de verdad como signos

R. Hähnle en (Hähnle, 1993) introduce un nuevo demostrador para las lógicas multivaluadas que utiliza una nueva técnica para eliminar las redundancias en el método de las tablas expuesto en la sección anterior. El objetivo es la búsqueda de todos los valores no destacados en paralelo. Ello requiere poder expresar con una sola etiqueta que una fórmula $A$ puede tomar diversos valores de verdad. Por ejemplo, en el caso de la lógica trivaluada se consigue expresar con una sola etiqueta que

$$\text{el valor de verdad de la fórmula } A \text{ es } 0 \text{ o } \tfrac{1}{2}$$

Para ello, se admiten subconjuntos del conjunto de valores de verdad como etiquetas.

**Definición 3.7** El conjunto base de signos es $\overline{S} = 2^{\mathbf{n}}$.

En principio se supone que el conjunto de etiquetas $\mathcal{E}_{\mathcal{L}}$ en una lógica $\mathcal{L}$ siempre satisface que

$$\{\{i\} |\ i \in S\} \subseteq \mathcal{E}_{\mathcal{L}} \subseteq \overline{S}$$

Esta condición se requiere para asegurar la corrección de las reglas. Posteriormente, se puede dar una condición más débil que mantiene la corrección de las mismas.

**Ejemplo 3.4** Para la lógica trivaluada débil de Kleene $\mathrm{KD}_3$ un conjunto de signos es

$$\mathcal{E}_{KD_3} = \{\{0\}, \{\frac{1}{2}\}, \{1\}, \{0, \frac{1}{2}\}, \{\frac{1}{2}, 1\}\}$$

$\{0,1\}A$ expresa que la fórmula $A$ tiene o bien el valor de verdad 0 o bien el valor$\frac{1}{2}$.

**Definición 3.8** Sea $\mathcal{L} = (Q, *_1, *_2, \ldots, *_k)$ un lenguaje para una lógica proposicional $n$-valuada y $\mathcal{M} = (\mathbf{n}, d_1, d_2, \ldots, d_m, *'_1, *'_2, \ldots, *'_k\}$ una matriz $n$-valuada para $\mathcal{L}$. Entonces se llama *álgebra de signos*, $\mathcal{M}_{\mathbf{n}} = \{\mathbf{n}, *''_1, *''_2, \ldots, *''_k\}$, al álgebra abstracta similar a $\mathcal{M}$ cuyas operaciones $*''_i$ se definen como sigue:

Si $(S_1, S_2, \ldots, S_m)$ es una $m$-upla de elementos de $2^{\mathbf{n}}$:

$$*''_i(S_1, S_2, \ldots, S_m) = \bigcup \{*'_i(j_1, \ldots, j_m) |\quad j_k \in S_k, 1 \leq k \leq m\}$$

El álgebra $\mathcal{M}_{\mathbf{n}}$ define la semántica de $\mathcal{L}$ en términos de conjuntos de valores de verdad.

**Definición 3.9** Sea $\mathcal{L}$ una lógica proposicional multivaluada, $A$ una fórmula de $\mathcal{L}$

$$A = *(A_1, A_2, \ldots, A_m)$$

y sean $*'$ y $*''$ las interpretaciones de $*$ en $\mathcal{M}$ y $\mathcal{M}_{\mathcal{S}}$ respectivamente.

Una *$\mathcal{L}$-regla para una tabla* es una función $\pi_{S,*}$ que asigna a la fórmula etiquetada $S\,A$ un árbol con raíz $S * (A_1, A_2, \ldots, A_m)$ llamada *premisa* y subárboles lineales (que se denota $\circ \ldots \circ$) $S_1\,A_{i_1} \circ \ldots \circ S_t\,A_{i_t}$ llamados *extensiones*, tales que $S_1, \ldots, S_t \in \mathcal{E}_{\mathcal{S}}$, donde $1 \leq t \leq m$ y cada uno de estas extensiones satisface la condición $H_{\mathcal{S}}(*; (S_1, i_1), \ldots, (S_t, i_t))$ definida a continuación.

El conjunto de todas las extensiones se denomina *conclusión* para una regla si se satisface:

**(T0a)** Para todo $(j_1, \ldots, j_m) \in *^{-1}(S)$ existe una extensión $S_1 *_{i_1} \circ \ldots \circ S_t *_{i_t}$ con $j_{i_k} \in S_k$ para todo $k$ tal que $1 \leq k \leq t$, y

**(T0b)** No existe ningún conjunto de extensiones con menos elementos y que satisfaga $(TOa)$

$H_{\mathcal{S}}(*; (S_1, i_1), \ldots, (S_t, i_t))$ se satisface si y sólo si existe un homomorfismo

$$h : \mathcal{L} \longrightarrow \mathcal{M}_{\mathcal{S}}$$

satisfaciendo las siguientes propiedades:

**(T1)** $h(*_{i_k}) = S_k$ para $1 \leq k \leq t$

**(T2)** $*''(S'_1, \ldots, S'_m) \subseteq S$ si $S'_{i_k} = S_k$ para todo $1 \leq k \leq t$ y el resto de los $S'_j$ son arbitrarios.

**(T3)** Para ningún $1 \leq k \leq t$ existe un $S'_k$ con $S_k \subsetneq S'_k$ que satisfaga $(T1)$ y $(T2)$.

**(T4)** No existe $t'$ con $t' < t$ que satisfaga $(T1)$ y $(T2)$.

En el caso de que no exista ningún homomorfismo de este tipo, no existe ninguna regla para $S\,A$.

$(T0a)$ especifica la corrección de las reglas respecto a $\mathcal{M}_{\mathcal{S}}$. Cada entrada de la tabla de verdad que es un elemento de $S$ ha de ser recogida por alguna extensión.

$(T0b)$ minimiza el número de extensiones. Así, aseguramos que de las tres posibles reglas correctas para $\{1\}(A \vee B)$ siguientes, nunca se selecciona la de la derecha:

$$\frac{\{1\}(A \vee B)}{\{1\}A \mid \{0\}A \mid \{1\}B} \qquad \frac{\{1\}(A \vee B)}{\{1\}A \mid \{1\}A} \qquad \frac{\{1\}(A \vee B)}{\begin{array}{c|c|c}\{1\}A & \{1\}A & \{0\}A \\ \{0\}B & \{0\}B & \{1\}B\end{array}}$$

Figura 3.3:

Puesto que $h$ es un homomorfismo, respeta la semántica de $\mathcal{L}$ y (T1) propaga esta propiedad a la regla mediante $h$.

(T2) garantiza la completitud respecto a $\mathcal{M}_{\mathcal{S}}$.

(T3) Hace a $S_k$ maximal y favorece, por ejemplo, que de las dos reglas de la siguiente figura, se seleccione la regla de la izquierda en lugar de la derecha.

$$\frac{\{\frac{1}{2}\}(A \vee B)}{\{0, \frac{1}{2}\}A \mid \{0\}A \mid \{1\}B} \qquad \frac{\{1\}(A \vee B)}{\{1\}A \mid \{1\}A} \qquad \frac{\{1\}(A \vee B)}{\begin{array}{c|c|c}\{1\}A & \{1\}A & \{0\}A \\ \{0\}B & \{0\}B & \{1\}B\end{array}}$$

Figura 3.4:

(T4) Minimiza el número de fórmulas en las extensiones. Favorece, por ejemplo, que de las reglas de la figura **??** se elija la de la izquierda.

(T0a), (T1) y (T2) garantizan la existencia de reglas correctas y completas y, de las diferentes reglas correctas y completas posibles, la definición 3.9 selecciona una con la ayuda de las condiciones (T0b), (T3) y (T4), que aún no es única salvo el orden de las fórmulas en las extensiones y el orden de las extensiones en la conclusión.

Las fórmulas signadas para las que no existe regla son intrínsecamente contradictorias y se incluyen en el *conjunto de contradicciones* que recoge las condiciones de cierre para las ramas en este nuevo método. Este conjunto considera, en lugar de pares de literales complementarios, un conjunto finito de fórmulas para las que no existe ningún valor de verdad común.

**Definición 3.10** El conjunto de *contradicciones* de $L^n_{M+}$ usando conjuntos de valores semánticos es

$$\begin{array}{rcl} Contr & = & \{\{S_1A, \ldots S_rA\} \mid 1 \leq r, \bigcap S_j = \varnothing, A \text{ es una fbf } \} \cup \\ & & \{\{SA\} \mid \text{ no existe regla para } SA\} \end{array}$$

### 3.3.1. Reglas de extensión

Para mostrar ejemplos de como ejecuta el método, damos a continuación algunas de las reglas de extensión para una lógica trivaluada. Un sistema completo puede verse en (Hähnle, 1993).

Consideraremos las conectivas $\neg, \sim, \rightarrow, \vee$ y $\wedge$ con similaridad $\langle 1, 1, 2, 2, 2\rangle$. Llamaremos a $\neg$ *negación fuerte*, a $\sim$ *negación débil*, a $\rightarrow$ *implicación débil*, a $\vee$ *disyunción* y a $\wedge$ *conjunción*. La semántica de estas conectivas se define del modo siguiente:

- $\neg i = 1 - i$
- $\sim i = \begin{cases} 1 & \text{si } i \in S - D, \text{ es decir, si } i = 0 \text{ ó } \frac{1}{2} \\ 0 & \text{si } i \in D, \text{ es decir, si } i = 1 \end{cases}$
- $i \rightarrow j = \begin{cases} 1 & \text{si } i \in S - D, \text{ es decir, si } i = 0 \text{ ó } \frac{1}{2} \\ j & \text{si } i \in D, \text{ es decir, si } i = 1 \end{cases}$
- $i \wedge j = \text{mín}\{i, j\}$.
- $i \vee j = \text{máx}\{i, j\}$.

por lo tanto, sus tablas de verdad son:

| | $\neg$ |
|---|---|
| 0 | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 |

| | $\sim$ |
|---|---|
| 0 | 1 |
| $\frac{1}{2}$ | 1 |
| 1 | 0 |

| $\rightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| $\frac{1}{2}$ | 1 | 1 | 1 |
| 1 | 0 | $\frac{1}{2}$ | 1 |

| $\vee$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 0 | $\frac{1}{2}$ | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 |
| 1 | 1 | 1 | 1 |

| $\wedge$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| $\frac{1}{2}$ | 0 | $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 | $\frac{1}{2}$ | 1 |

Las reglas para estas conectivas son:

**Conjunción**

$$\frac{\{0\}(A \wedge B)}{\{0\}A \mid \{0\}B} \qquad \frac{\{\frac{1}{2}\}(A \wedge B)}{\begin{array}{c|c} \{\frac{1}{2}, 1\}A & \{\frac{1}{2}\}B \\ \{\frac{1}{2}\}B & \{\frac{1}{2}, 1\}B \end{array}} \qquad \frac{\{1\}(A \wedge B)}{\begin{array}{c} \{1\}A \\ \{1\}B \end{array}}$$

$$\frac{\{0, \frac{1}{2}\}(A \wedge B)}{\{0, \frac{1}{2}\}A \mid \{0, \frac{1}{2}\}B} \qquad \frac{\{\frac{1}{2}, 1\}(A \wedge B)}{\begin{array}{c} \{\frac{1}{2}, 1\}A \\ \{\frac{1}{2}, 1\}B \end{array}}$$

**Disyunción**

$$\frac{\{0\}(A \vee B)}{\begin{array}{c}\{0\}A\\ \{0\}B\end{array}} \qquad \frac{\{\frac{1}{2}\}(A \vee B)}{\begin{array}{c|c}\{0,\frac{1}{2}\}A & \{\frac{1}{2}\}B\\ \{\frac{1}{2}\}B & \{0,\frac{1}{2}\}B\end{array}} \qquad \frac{\{1\}(A \vee B)}{\{1\}A \mid \{1\}B}$$

$$\frac{\{0,\frac{1}{2}\}(A \vee B)}{\begin{array}{c}\{0,\frac{1}{2}\}A\\ \{0,\frac{1}{2}\}B\end{array}} \qquad \frac{\{\frac{1}{2},1\}(A \vee B)}{\{\frac{1}{2},1\}A \mid \{\frac{1}{2},1\}B}$$

**Implicación Débil**

$$\frac{\{0\}(A \to B)}{\begin{array}{c}\{1\}A\\ \{0\}B\end{array}} \qquad \frac{\{\frac{1}{2}\}(A \vee B)}{\begin{array}{c}\{1\}A\\ \{\frac{1}{2}\}B\end{array}} \qquad \frac{\{1\}(A \to B)}{\{0,\frac{1}{2}\}A \mid \{1\}B}$$

$$\frac{\{0,\frac{1}{2}\}(A \vee B)}{\begin{array}{c}\{0,\frac{1}{2}\}A\\ \{0,\frac{1}{2}\}B\end{array}} \qquad \frac{\{\frac{1}{2},1\}(A \vee B)}{\{\frac{1}{2},1\}A \mid \{\frac{1}{2},1\}B}$$

**Negación Fuerte**

$$\frac{\{0\}\neg A}{\{1\}A} \qquad \frac{\{\frac{1}{2}\}\neg A}{\{\frac{1}{2}\}A} \qquad \frac{\{1\}\neg A}{\{0\}A} \qquad \frac{\{0,\frac{1}{2}\}\neg A}{\{\frac{1}{2},1\}A} \qquad \frac{\{\frac{1}{2},1\}\neg A}{\{0,\frac{1}{2}\}A}$$

**Negación Débil**

$$\frac{\{0\} \sim A}{\{1\}A} \qquad \frac{\{1\}\neg A}{\{0,\frac{1}{2}\}A} \qquad \frac{\{0,\frac{1}{2}\}\neg A}{\{1\}A} \qquad \frac{\{\frac{1}{2},1\}\neg A}{\{0,\frac{1}{2}\}A}$$

**Ejemplo 3.5** Aplicamos ahora el método a la fórmula $\neg p \rightarrow (\sim p \wedge \neg p)$.

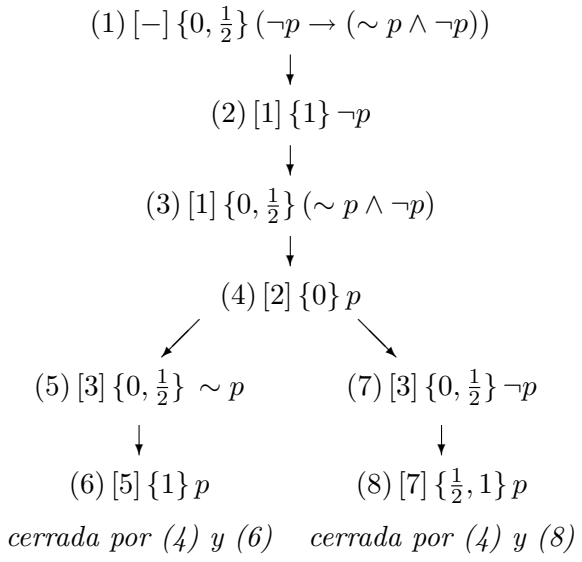



Figura 3.5: El árbol para $\neg p \rightarrow (\sim p \wedge \neg p)$.

**Teorema 3.6** *El sistema de* conjuntos de valores de verdad como signos *es correcto y completo.*

# Capítulo 4

# TAS-M3: Un ATP para Lógicas Trivaluadas

## 4.1. La metodología TAS

La metodología TAS para el desarrollo de ATPs fue introducida en la tesis doctoral de D. Francisco Sanz (Sanz, 1992). Uno de los objetivos de la nueva metodología era que fuese flexible y que pudiese extenderse a lógicas no clásicas, el presente trabajo es una prueba más de que ese objetivo fue cumplido.

Al igual que el resto de los demostradores desarrollados en el marco de la metodología TAS (Aguilera *et al.*, 1995a) por nuestro grupo de investigación GIMAC (marcas (Aguilera *et al.*, 1994d; Aguilera *et al.*, 1994c; Aguilera *et al.*, 1995b; Aguilera *et al.*, 1994a; Aguilera *et al.*, 1994b; Aguilera *et al.*, 1993; Ojeda, 1996) para la lógica clásica y (de Guzmán *et al.*, 1995; de Guzmán y Enciso, 1995; Ojeda *et al.*, 1996; Enciso, 1995) para lógicas modales y temporales), y siguiendo la filosofía general del método, el demostrador TAS-M**3** es un demostrador por refutación, es decir, la entrada del método es la negación fuerte de la fórmula, y mediante sucesivas transformaciones de su árbol sintáctico (es un método de reescritura), se va reduciendo el tamaño de la fórmula para evitar tantas distribuciones de $\wedge$ sobre $\vee$ como sea posible.

El diseño del algoritmo hereda las propiedades características de la metodología TAS, es decir, es eficiente, flexible, fácilmente adaptable y paralelizable. Esta última propiedad, permite que la única parte del algoritmo con complejidad exponencial en el peor caso, pueda ejecutarse de forma paralela.

Como ya hemos comentado, las características señaladas en el párrafo anterior han sido destacadas en (Hähnle, 1993) como las deseables para un demostrador. A lo largo del desarrollo de este capítulo, quedará patente que TAS-M**3** las satisface.

La idea clave del método es el uso de interpretaciones parciales unitarias asociadas a cada nodo del árbol sintáctico de la fórmula. Estas interpretaciones, agrupadas en los denominados conjuntos $\Delta$, permiten efectuar *reducciones* (lo que caracteriza a los métodos TAS) en la fórmula en estudio, utilizando distintos procesos.

Los procesos utilizados en el método no se ejecutan uno tras otro sin más, sino que la fórmula es analizada y, dinámicamente, se decide el proceso a aplicar.

## 4.2. La Lógica M3

La lógica M**3** que utilizamos es una extensión de la lógica introducida por Rosser y Turquette (Rosser y Turquette, 1952) y utilizada en (Hähnle, 1993). M**3** viene dada por el par $(\mathcal{L}, \mathcal{M})$ en el que el álgebra $\mathcal{L} = (Q, \bot, \oslash, \top, \neg, \sim, \vee, \wedge, \rightarrow)$ tiene como tipo de similaridad $\langle 0, 0, 0, 1, 1, 2, 2, 2\rangle$ y como matriz asociada $\mathcal{M} = (\mathbf{3}, 1, \bot, \oslash, \top, \neg, \sim, \vee, \wedge, \rightarrow)$. Llamaremos a los elementos de $Q$ *símbolos proposicionales*, a $\neg$ *negación fuerte*, a $\sim$ *negación débil*, a $\vee$ *disyunción*, a $\wedge$ *conjunción* y a $\rightarrow$ *implicación débil*. La semántica de estas conectivas se define del modo siguiente:

- $\bot, \oslash$ y $\top$ (conectivas de aridad 0) tienen como función de verdad asociada, las funciones constantes $0, \frac{1}{2}$ y 1 respectivamente.
- $\neg i = 1 - i$
- $\sim i = \begin{cases} 1 & \text{si } i \in S - D, \text{ es decir, si } i = 0 \text{ ó } \frac{1}{2} \\ 0 & \text{si } i \in D, \text{ es decir, si } i = 1 \end{cases}$
- $i \wedge j = \text{mín}\{i, j\}$.
- $i \vee j = \text{máx}\{i, j\}$.
- $i \rightarrow j = \begin{cases} 1 & \text{si } i \in S - D, \text{ es decir, si } i = 0 \text{ ó } \frac{1}{2} \\ j & \text{si } i \in D, \text{ es decir, si } i = 1 \end{cases}$

por lo tanto, sus tablas de verdad son:

| | $\neg$ |
|---|---|
| 0 | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 |

| | $\sim$ |
|---|---|
| 0 | 1 |
| $\frac{1}{2}$ | 1 |
| 1 | 0 |

| $\vee$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 0 | $\frac{1}{2}$ | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 1 |
| 1 | 1 | 1 | 1 |

| $\wedge$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| $\frac{1}{2}$ | 0 | $\frac{1}{2}$ | $\frac{1}{2}$ |
| 1 | 0 | $\frac{1}{2}$ | 1 |

| $\rightarrow$ | 0 | $\frac{1}{2}$ | 1 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| $\frac{1}{2}$ | 1 | 1 | 1 |
| 1 | 0 | $\frac{1}{2}$ | 1 |

Las conectivas monarias de afirmación de valores de verdad $J_i$ (introducidas en la sección 2.5.1) pueden definirse en función del conjunto de conectivas $\{\bot, \oslash, \top, \neg, \sim, \vee, \wedge, \rightarrow\}$ del siguiente modo:

$$\begin{array}{lcl} J_0 A & \equiv & \neg \sim \neg A \\ J_{\frac{1}{2}} A & \equiv & \sim A \wedge \sim \neg A \\ J_1 A & \equiv & \neg \sim A \end{array}$$

**Teorema 4.1** *La lógica M***3** *es funcionalmente completa.*

DEMOSTRACIÓN: Siguiendo el estilo de demostración de Post, demostraremos que todo operador $f : \mathbf{3}^n \rightarrow \mathbf{3}$ puede expresarse como composición de elementos del conjunto $\Theta = \{\bot, \oslash, \top, \neg, \sim, \vee, \wedge, \rightarrow\}$. Realizaremos la demostración por inducción sobre la aridad $n$ del operador $f$.

La propiedad es cierta en el caso base ($n = 0$) ya que los únicos operadores de aridad 0 que pueden definirse sobre **3** son precisamente $\bot, \oslash$ y $\top$, todos ellos pertenecientes al conjunto $\Theta$.

Supondremos cierta la propiedad para el caso $n = k$ y demostraremos que es cierta cuando $n = k + 1$.

Sea $f : \mathbf{3}^{k+1} \rightarrow \mathbf{3}$.

$$\begin{array}{lcl} f(x_1, x_2, \ldots, x_k, x_{k+1}) & = & (f(x_1, x_2, \ldots, x_k, 0) \wedge J_0(x_{k+1})) \vee \\ & & (f(x_1, x_2, \ldots, x_k, \frac{1}{2}) \wedge J_{\frac{1}{2}}(x_{k+1})) \vee \\ & & (f(x_1, x_2, \ldots, x_k, 1) \wedge J_1(x_{k+1})) \\ & = & (f_0(x_1, x_2, \ldots, x_k) \wedge J_0(x_{k+1})) \vee \\ & & (f_{\frac{1}{2}}(x_1, x_2, \ldots, x_k) \wedge J_{\frac{1}{2}}(x_{k+1})) \vee \\ & & (f_1(x_1, x_2, \ldots, x_k) \wedge J_1(x_{k+1})) \end{array}$$

donde cada $f_i$, con $i \in \mathbf{3}$, es la restricción de $f$ a una función $f_i : \mathbf{3}^k \rightarrow \mathbf{3}$ definida como sigue:

$$f_i(x_1, x_2, \ldots, x_k) = f(x_1, x_2, \ldots, x_k, i)$$

Puesto que, por hipótesis de inducción, las $f_i$ con $i \in \mathbf{3}$ cumplen la propiedad y las afirmaciones de valores de verdad $J_i$ con $i \in \mathbf{3}$ también pueden ser expresadas utilizando únicamente operadores del conjunto $\Theta$, se tiene lo que queríamos demostrar. *q.e.d.*

El hecho de que esta lógica sea funcionalmente completa permite definir cualquier operador trivaluado en función de los operadores seleccionados, así por ejemplo para la negación $\sigma$ de la lógica trivaluada de Post se tiene:

$$\sigma A \equiv (\oslash \wedge \neg \sim \neg A) \vee (\sim A \wedge \sim \neg A)$$

y para la implicación en la lógica trivaluada de Łukasiewicz:

$$A \supset B \equiv (\neg A \vee B) \vee ((\sim A \wedge \sim \neg A) \wedge (\sim B \wedge \sim \neg B))$$

## 4.3. El Método

Como hemos indicado, TAS-M**3** es un método de refutación; para analizar la validez de una fórmula $A$, utilizamos el teorema 4.2 en el que hacemos uso de la definición siguiente:

**Definición 4.1** Dada una fórmula $A$ en M**3** diremos que $A$ es *casisatisfacible* si existe una interpretación $I$ tal que $I(A) \in \{\frac{1}{2}, 1\}$.

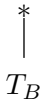
La *casisatisfacibilidad* juega aquí un papel similar al de la satisfacibilidad en la lógica clásica, según nos muestra el siguiente teorema, de demostración trivial:

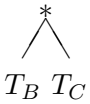
**Teorema 4.2** *Dada una fbf $A$ en M**3** se tiene que*

- *$A$ es no válida si y sólo si $\neg A$ es casisatisfacible.*
- *$A = \bigvee_{i \in I} A_i$ es casisatisfacible si y sólo si $A_i$ es casisatisfacible para algún $i \in I$*

**Definición 4.2** El *árbol sintáctico* de una fbf $A$ de M**3** es el árbol binario, recursivamente definido como sigue:

1. Si $A \in Q \cup \{\bot, \oslash, \top\}$, entonces $T_A$ es $A$.

2. Si $A = *B$ con $* \in \{\neg, \sim\}$, entonces $T_A$ es 


3. Si $A = B * C$ con $* \in \{\vee, \wedge, \rightarrow\}$, entonces $T_A$ es 


TAS-M**3** tiene como entrada la negación fuerte de una fórmula, o más concretamente, el árbol sintáctico de la negación fuerte de la fórmula. Analizando la estructura de este árbol sintáctico, el método, dinámicamente, realiza transformaciones, reescribiendo el mismo, intentando en lo posible evitar distribuciones mediante la disminución del tamaño de la fórmula.

Supondremos que en la fórmula cuya validez se desea analizar, y por tanto en la entrada de TAS-M**3**, no intervienen las constantes $\bot$, $\oslash$ y $\top$. Más adelante

mostraremos que esto no supone pérdida de generalidad, ya que, si $\bot$, $\oslash$ y $\top$ intervienen en la fórmula, la ejecución del método las elimina proporcionando una fórmula que es casisatisfacible si y sólo si lo es la fórmula de partida.

El diagrama de flujo del método es el mismo que para los otros demostradores TAS, lo que cambia es la descripción de los procesos de transformación de árboles que intervienen, así, el flujo del algoritmo puede verse en la figura **??**

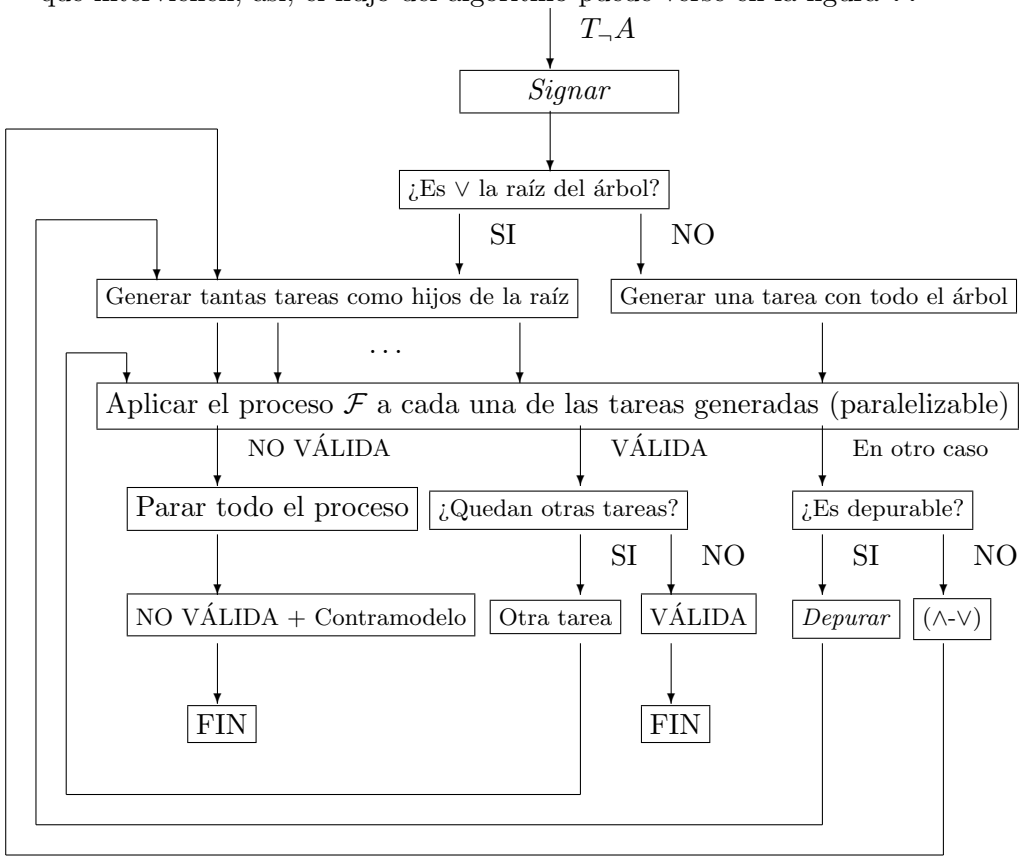



Figura 4.1: El método TAS-M**3**

## 4.4. El proceso *Signar*

El proceso *Signar* constituye, tal como se puede ver en la figura **??**, la primera etapa del algoritmo. El objetivo de esta etapa es transformar el árbol sintáctico de la fórmula de entrada en el de otra lógicamente equivalente a la primera, pero en

la que no ocurra $\rightarrow$, y en la que, en términos de árboles sintácticos, las conectivas $\vee$ y $\wedge$ no tienen como ascendiente una conectiva monaria. Una fórmula tal se dice que está en *forma normal unaria*, este concepto se introducirá formalmente más tarde.

Para realizar la transformación indicada sobre la fórmula de entrada, *Signar* hace uso de las siguientes leyes:

1. Leyes de eliminación de la conectiva $\rightarrow$:

$$\begin{aligned} A \rightarrow B &\equiv \sim A \vee B \\ \neg(A \rightarrow B) &\equiv \neg \sim A \wedge \neg B \\ \sim (A \rightarrow B) &\equiv \neg \sim A \wedge \sim B \end{aligned}$$

2. Leyes de Morgan:

$$\begin{aligned} &a)\ \neg(A \wedge B) \equiv \neg A \vee \neg B \\ &b)\ \neg(A \vee B) \equiv \neg A \wedge \neg B \\ &c)\ \sim (A \wedge B) \equiv \sim A \vee \sim B \\ &d)\ \sim (A \vee B) \equiv \sim A \wedge \sim B \end{aligned}$$

3. Leyes de negaciones múltiples:

$$a)\ \neg^k A \equiv \begin{cases} A & \text{si } k \text{ es par} \\ \neg A & \text{si } k \text{ es impar} \end{cases}$$

$$b)\ \sim^k A \equiv \begin{cases} \neg \sim A & \text{si } k \text{ es par} \\ \sim A & \text{si } k \text{ es impar} \end{cases}$$

$$\begin{aligned} &c)\ \sim \neg \sim A \equiv \sim A \\ &d)\ \neg \sim \neg \sim A \equiv \neg \sim A \\ &e)\ \sim \neg \sim \neg A \equiv \sim \neg A \end{aligned}$$

A partir de las leyes de negaciones múltiples obtenemos que las únicas secuencias de $\neg$ y $\sim$, salvo equivalencia, son las secuencias pertenecientes al conjunto siguiente, donde $\epsilon$ denota la secuencia vacía:

$$\{\epsilon, \neg, \sim, \neg \sim, \sim \neg, \neg \sim \neg\}$$

en definitiva, por la semántica de M**3**, podemos considerar las secuencias pertenecientes al conjunto

$$\{\epsilon, \neg, \neg J_1, J_1, \neg J_0, J_0\}$$

Sin embargo, si deseamos disponer en M**3** de la potencia del método (en definitiva la drástica disminución de distribuciones requeridas) disponible en el caso

proposicional clásico, requerimos, como justificaremos a lo largo de este trabajo, introducir además las secuencias $J_{\frac{1}{2}}$ y $\neg J_{\frac{1}{2}}$ que si bien no intervienen en la fórmula de entrada, sí serán generadas por la ejecución del método. En consecuencia, consideraremos la siguiente definición:

**Definición 4.3** Llamaremos *literal trivaluado* a toda fbf de la forma $\alpha p$ donde $p \in Q$ y $\alpha \in \Lambda$ con

$$\Lambda = \{\epsilon, J_0, J_{\frac{1}{2}}, J_1, \neg, \neg J_0, \neg J_{\frac{1}{2}}, \neg J_1\}$$

Dado un literal trivaluado $\alpha p$, llamaremos a $\alpha$ el *prefijo del literal* y a $p$ el *sufijo del literal*.

**Lema 4.1** *Para todo $b \in \mathbf{3}$ y todo símbolo proposicional $p$ se tiene que*

$$J_b p \wedge \neg J_b p \equiv \bot \quad y \quad J_b p \vee \neg J_b p \equiv \top$$

DEMOSTRACIÓN: Inmediato por la semántica de las conectivas $J_b$. *q.e.d.*

**Lema 4.2** *Todo literal trivaluado es satisfacible y, por lo tanto, casisatisfacible.*

DEMOSTRACIÓN: La demostración es trivial ya que basta comprobar que en cada columna de la tabla siguiente hay al menos un 1.

| | $\epsilon$ | $\neg$ | $J_0$ | $\neg J_0$ | $J_{\frac{1}{2}}$ | $\neg J_{\frac{1}{2}}$ | $J_1$ | $\neg J_1$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

*q.e.d.*

**Definición 4.4** Una fórmula de M**3** está en *forma normal unaria* o, abreviadamente, es una fnu si pertenece a la clausura inductiva libremente generada del conjunto de literales trivaluados

$$\{\alpha p \mid p \in Q \text{ y } \alpha \in \{\epsilon, J_0, J_{\frac{1}{2}}, J_1, \neg, \neg J_0, \neg J_{\frac{1}{2}}, \neg J_1\}\}$$

para el conjunto de constructores $\{C_\wedge, C_\vee\}$ definidos como sigue:

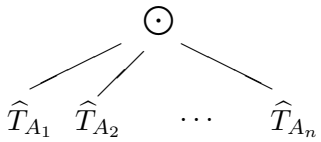
Para dos cadenas $X$ e $Y$ cualesquiera del alfabeto de M**3**,

$$\begin{aligned} C_\wedge(X,Y) &= (X \wedge Y) \\ C_\vee(X,Y) &= (X \vee Y) \end{aligned}$$

Las leyes asociativas permiten considerar como fbfs a expresiones de la forma $A_1 \vee \cdots \vee A_n$ y $A_1 \wedge \cdots \wedge A_n$ (abreviadamente $\bigvee_{i=1}^{n} A_i$ y $\bigwedge_{i=1}^{n} A_i$). Este hecho, junto con la consideración de los nuevos prefijos de literal trivaluado en $\Lambda$, nos llevan a la siguiente definición de árbol sintáctico de una fnu:

**Definición 4.5** Sea $A$ una fnu, su *árbol sintáctico generalizado*, denotado por $\widehat{T}_A$, se define recursivamente como sigue:

1. Si $A = \bigodot_{i=1}^{n} A_i$, donde $\bigodot$ es $\bigwedge$ o $\bigvee$, entonces $\widehat{T}_A$ es




2. $\widehat{T}_A = A$ en otro caso, es decir, si $A$ es un literal trivaluado o una constante.

Obsérvese que en un árbol sintáctico generalizado, a diferencia de lo que ocurre en un árbol sintáctico, sus hojas son literales trivaluados o constantes.

Por abuso del lenguaje, hablaremos sólo de árboles sintácticos y utilizaremos únicamente la representación $T_A$, siendo el contexto el que determinará si se trata de árboles sintácticos generalizados o no.

### Cómo trabaja el proceso *Signar*

El proceso *Signar* transforma el árbol sintáctico de una fórmula en otro correspondiente a una fnu. Para ello, este proceso utiliza un proceso recursivo que se aplica a la raíz del árbol sintáctico de la fórmula de entrada y desciende hasta las hojas, eliminando las conectivas $\rightarrow$ y bajando las negaciones (transmitiéndolas a las hojas):

**Definición 4.6** Dada una fórmula $A$ y $\kappa \in \{-3, -2, -1, 1, 2, 3\}$, $A(\kappa)$ se define recursivamente como sigue:

| | |
|---|---|
| $p(1) = p$ | $p(-1) = \neg p$ |
| $p(2) = \neg J_0 p$ | $p(-2) = J_0 p$ |
| $p(3) = J_1 p$ | $p(-3) = \neg J_1 p$ |
| $(A * B)(\kappa) = A(\kappa) * B(\kappa)$ | si $\kappa > 0$ y $* \in \{\vee, \wedge\}$ |
| $(A * B)(\kappa) = A(\kappa) \overline{*} B(\kappa)$ | si $\kappa < 0$, $* \in \{\vee, \wedge\}$, $\overline{\vee} = \wedge$ y $\overline{\wedge} = \vee$ |
| $(A \rightarrow B)(\kappa) = A(-3) \vee B(\kappa)$ | si $\kappa > 0$ |
| $(A \rightarrow B)(\kappa) = A(3) \wedge B(\kappa)$ | si $\kappa < 0$ |
| $(\sim A)(\kappa) = A(-3)$ | si $\kappa > 0$ |
| $(\sim A)(\kappa) = A(3)$ | si $\kappa < 0$ |
| $(\neg A)(\kappa) = A(\gamma(\kappa))$ | donde $\gamma$ es la función definida como sigue: |

$$\gamma : \{-3,-2,-1,1,2,3\} \longrightarrow \{-3,-2,-1,1,2,3\},$$
$$\gamma(1) = -1; \qquad \gamma(-1) = 1$$
$$\gamma(2) = -3; \qquad \gamma(-2) = 3$$
$$\gamma(3) = -2; \qquad \gamma(-3) = 2$$

**Definición 4.7** Dada una fórmula $A$, *Signar* su árbol sintáctico $T_A$ es obtener el árbol sintáctico asociado a $A\,(1)$.

**Ejemplo 4.1**

Dada la fórmula $A = \neg(\neg p \to (\sim p \wedge \neg p))$, el proceso *Signar* obtiene a partir de su árbol sintáctico $T_A$, el árbol $T_{A(1)}$ de la figura **??**

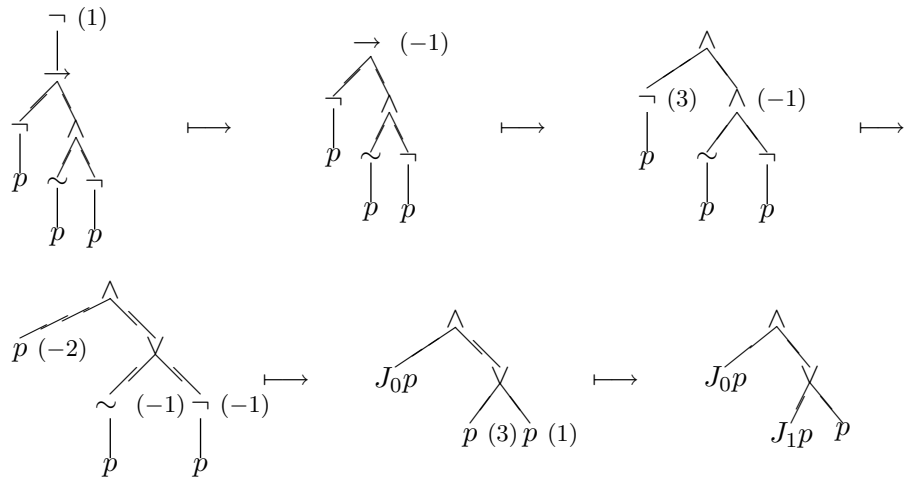



Figura 4.2: Traza del proceso *Signar*.

Advirtamos que, aunque para una mayor comprensión del proceso la figura muestra su traza, incluyendo cada uno de las acciones realizadas, todas ellas se ejecutan en un solo recorrido del árbol.

**Teorema 4.3** *El proceso* Signar *conserva el significado, es decir, para toda fbf $A$ se tiene que $Signar(A) \equiv A$.*

Demostración: Para demostrar este teorema basta considerar que:

La definición de $(A * B)(\kappa)$, con $* \in \{\wedge, \vee\}$ y $\kappa \in \{-3,-2,-1,1,2,3\}$, no es más que la traducción de las leyes de Morgan.

La definición de $(A \to B)(\kappa)$ no es más que la traducción de las leyes de eliminación de la conectiva $\to$.

1. La justificación de las definiciones de $(\sim A)(\kappa)$ y $(\neg A)(\kappa)$ viene dada por la siguiente tabla, que muestra la composición de las conectivas monarias $\neg$ y $\sim$:

| $\circ$ | $\epsilon$ | $\sim\neg$ | $\neg\sim$ | $\neg$ | $\neg\sim\neg$ | $\sim$ |
|---|---|---|---|---|---|---|
| $\neg$ | $\neg$ | $\neg\sim\neg$ | $\sim$ | $\epsilon$ | $\sim\neg$ | $\neg\sim$ |
| $\sim$ | $\sim$ | $\neg\sim\neg$ | $\sim$ | $\sim\neg$ | $\sim\neg$ | $\neg\sim$ |

y las equivalencias:

$$\begin{aligned} J_0 A &\equiv \neg\sim\neg A \\ J_1 A &\equiv \neg\sim A \end{aligned}$$

*q.e.d.*

## 4.5. La generación de tareas

Como ya hemos mencionado, TAS-M**3** es un método por refutación, esta característica permite realizar transformaciones que sólo conserven la casisatisfacibilidad, sin necesidad de conservar también la equivalencia lógica. Una de las estrategias más claras basadas en este hecho se puede aplicar cuando el método está analizando un árbol $T_B$ para determinar la casisatisfacibilidad de $B$ y tenga raíz $\vee$, en este caso, si comprobamos que la fórmula asociada a algún subárbol hijo de la raíz de $T_B$ es casisatisfacible, tendremos inmediatamente la casisatisfacibilidad de $B$. De este modo, si la fórmula $B$ es equicasisatisfacible con una fórmula $\neg A$ entonces la casisatisfacibilidad de $B$ implicará la no validez de $A$.

Así pues, cuando la raíz del árbol analizado por TAS-M**3** sea $\vee$, podemos aplicar el método independientemente a cada uno de los subárboles hijos de la raíz, en lo que llamaremos una tarea. Estas tareas pueden ejecutarse en paralelo, lo que aumenta la eficiencia. Basta con que al procesar una tarea se asegure que la fórmula asociada a su árbol es casisatisfacible (en realidad, al ser un método por refutación, la salida de la tarea será "NO VÁLIDA") para que el método termine con salida "NO VÁLIDA". Por el contrario si todas las tareas contestan "VÁLIDA", la salida general será "VÁLIDA".

Esta estrategia, nos permite estudiar cada uno de los hijos de una raíz $\vee$ como una tarea independiente, lo que además de permitir su paralelización, disminuye el tamaño de los árboles a los que se aplica el método (este hecho entre otras cosas aumenta la probabilidad de que tanto el proceso $\mathcal{S}$ como el proceso de 0-reducción completa actúen).

Observando el diagrama de flujo de TAS-M**3**, se puede comprobar que esta estrategia no sólo se aplica cuando el árbol inicial, tras el signado tiene raíz $\vee$, sino que cada vez que actúan los procesos ($\wedge$-$\vee$) y *Depurar* vuelve a aplicarse.

A esta estrategia de "división del trabajo" la llamamos generación de tareas, y obviamente si la raíz del árbol no es $\vee$, se estudia el árbol completo como única tarea.

A cada una de las tareas generadas se le aplica el proceso $\mathcal{F}$, este proceso (descrito en la sección 4.8) es el núcleo donde se realizan las reducciones. Veamos ahora cómo se generan las tareas.

1. Si la raíz del árbol es $\vee$, se generan tantas tareas como hijos tenga la raíz. Cada tarea corresponde a un hijo de la raíz, y será la entrada del proceso $\mathcal{F}$.

2. En otro caso se genera una única tarea correspondiente a todo el árbol de entrada, a la que se le aplica el proceso $\mathcal{F}$.

La salida general de las tareas generadas depende de sus salidas particulares de la siguiente forma:

- "NO VÁLIDA" si alguna de las tareas generadas responde con salida parcial "NO VÁLIDA".
- "VÁLIDA" si todas las tareas generadas responden "VÁLIDA".

## 4.6. Los conjuntos Δ

La base semántica de TAS-M**3** es la misma que la de todos los métodos TAS, la utilización de interpretaciones parciales unitarias de una fórmula y su negación. Para ello, se hace uso, como herramienta fundamental, de los conjuntos Δ, que permiten almacenar información para la detección de subfórmulas equivalentes a constantes, o bien sustituir la fórmula por otra de menor tamaño. Específicamente, la principal idea que justifica la definición de los conjuntos Δ es la de detectar literales trivaluados que satisfagan las siguientes propiedades:

1. Si $\models (l \rightarrow A)$ entonces $A \equiv l \vee B$
2. Si $\models (A \rightarrow l)$ entonces $A \equiv l \wedge B$

donde $B$ es una fórmula de menor tamaño que $A$.

Como podemos ver en el Apéndice A, para la Lógica Clásica, existen dos tipos de conjuntos Δ, uno para cada valor de verdad $\Delta_0$ y $\Delta_1$. En el caso de lógicas trivaluadas existirán tres conjuntos Δ asociados a cada fbf. Llamaremos a estos conjuntos $\Delta_0$, $\Delta_{\frac{1}{2}}$ y $\Delta_1$. Los elementos de los conjuntos Δ son de la forma $p\,b$ con $p \in Q$ y $b \in \mathbf{3}$. Si $A$ es una fbf, el hecho de que $pb_1 \in \Delta_{b_2}(A)$, con

$b_1, b_2 \in \mathbf{3}$ significa que para cualquier interpretación $I$ tal que $I(p) = b_1$ se tiene que $I(A) = b_2$.

El cálculo de los conjuntos $\Delta$ se realiza mediante un proceso recursivo que constituye la definición formal de estos conjuntos.

**Definición 4.8** Sea $p$ un símbolo proposicional y sean $A$ y $A_i$ fnus. Los *conjuntos* $\Delta_0$, $\Delta_{\frac{1}{2}}$ y $\Delta_1$ se definen recursivamente de la siguiente forma:

$$\Delta_0(p) = \{p0\}; \quad \Delta_{\frac{1}{2}}(p) = \{p\tfrac{1}{2}\}; \quad \Delta_1(p) = \{p1\}$$

La definición para el resto de los casos hace uso de la semántica de los literales trivaluados, así como de la semántica de las conectivas $\wedge$ y $\vee$:

$$\begin{array}{lll}
\Delta_0(\neg p) = \{p1\}; & \Delta_{\frac{1}{2}}(\neg p) = \{p\tfrac{1}{2}\}; & \Delta_1(\neg p) = \{p0\} \\
\Delta_0(J_0 p) = \{p\tfrac{1}{2}, p1\}; & \Delta_{\frac{1}{2}}(J_0 p) = \varnothing; & \Delta_1(J_0 p) = \{p0\} \\
\Delta_0(\neg J_0 p) = \{p0\}; & \Delta_{\frac{1}{2}}(\neg J_0 p) = \varnothing; & \Delta_1(\neg J_0 p) = \{p\tfrac{1}{2}, p1\} \\
\Delta_0(J_{\frac{1}{2}} p) = \{p0, p1\}; & \Delta_{\frac{1}{2}}(J_{\frac{1}{2}} p) = \varnothing; & \Delta_1(J_{\frac{1}{2}} p) = \{p\tfrac{1}{2}\} \\
\Delta_0(\neg J_{\frac{1}{2}} p) = \{p\tfrac{1}{2}\}; & \Delta_{\frac{1}{2}}(\neg J_{\frac{1}{2}} p) = \varnothing; & \Delta_1(\neg J_{\frac{1}{2}} p) = \{p0, p1\} \\
\Delta_0(J_1 p) = \{p0, p\tfrac{1}{2}\}; & \Delta_{\frac{1}{2}}(J_1 p) = \varnothing; & \Delta_1(J_1 p) = \{p1\} \\
\Delta_0(\neg J_1 p) = \{p1\}; & \Delta_{\frac{1}{2}}(\neg J_1 p) = \varnothing; & \Delta_1(\neg J_1 p) = \{p0, p\tfrac{1}{2}\}; \\
\Delta_0(\bigwedge_{i=1}^{n} A_i) = \bigcup_{i=1}^{n} \Delta_0(A_i); & & \Delta_1(\bigwedge_{i=1}^{n} A_i) = \bigcap_{i=1}^{n} \Delta_1(A_i) \\
\Delta_0(\bigvee_{i=1}^{n} A_i) = \bigcap \Delta_0(A_i); & & \Delta_1(\bigvee_{i=1}^{n} A_i) = \bigcup \Delta_1(A_i)
\end{array}$$

Queda por definir $\Delta_{\frac{1}{2}}(\bigwedge_{i=1}^{n} A_i)$ y $\Delta_{\frac{1}{2}}(\bigvee_{i=1}^{n} A_i)$: La definición para el caso $n = 2$ es:

- $\Delta_{\frac{1}{2}}(A_1 \wedge A_2) = [\Delta_{\frac{1}{2}}(A_1) \cap (\Delta_{\frac{1}{2}}(A_2) \cup \Delta_1(A_2))] \cup (\Delta_1(A_1) \cap \Delta_{\frac{1}{2}}(A_2))$
- $\Delta_{\frac{1}{2}}(A_1 \vee A_2) = [\Delta_{\frac{1}{2}}(A_1) \cap (\Delta_{\frac{1}{2}}(A_2) \cup \Delta_0(A_2))] \cup (\Delta_0(A_1) \cap \Delta_{\frac{1}{2}}(A_2))$

Para el caso $n > 2$ se utiliza la definición anterior asociando a izquierda, es decir,

$$\Delta_{\frac{1}{2}}(\bigwedge_{i=1}^{n} A_i) = \Delta_{\frac{1}{2}}((\bigwedge_{i=1}^{n-1} A_i) \wedge A_n); \qquad \Delta_{\frac{1}{2}}(\bigvee_{i=1}^{n} A_i) = \Delta_{\frac{1}{2}}((\bigvee_{i=1}^{n-1} A_i) \vee A_n))$$

**Nota:**

La dificultad que puede aportar al método el uso de $\Delta_{\frac{1}{2}}$ es tan sólo aparente como nos muestra el siguiente resultado, consecuencia inmediata de la definición:

- Si $l$ es un literal trivaluado cuyo prefijo no es $\epsilon$ ni $\neg$, entonces $\Delta_{\frac{1}{2}}(l) = \varnothing$.
- Sean $A_1, \ldots, A_n$ fnus tales que $\Delta_{\frac{1}{2}}(A_i) = \varnothing$ para todo $i$, $1 \leq i \leq n$, entonces $\Delta_{\frac{1}{2}}(\bigwedge_{i=1}^{n} A_i) = \Delta_{\frac{1}{2}}(\bigvee_{i=1}^{n} A_i) = \varnothing$.

**Definición 4.9** Dada una fnu, $A$, llamamos *Etiquetar* al proceso recursivo que, partiendo de la raíz de $T_A$, asocia a cada nodo $N$ de $T_A$ la terna

$$(\Delta_0(B), \Delta_{\frac{1}{2}}(B), \Delta_1(B))$$

, donde $B$ es la subfórmula de $A$ determinada por $N$.

**Ejemplo 4.2** Sea $T_B$ es el árbol obtenido al finalizar el proceso *Signar* del ejemplo 4.1. Puesto que la raíz de este árbol no es $\vee$ la tarea generada tras el signado es todo el árbol. La figura **??** muestra el proceso de etiquetado de este árbol $T_B$.

## 4.7. Información en los conjuntos Δ

Los conjuntos Δ asociados a una fórmula $A$ pueden permitirnos asegurar su equivalencia a una constante, su equivalencia a un literal trivaluado, o transformarla en otra fórmula $B$ de menor tamaño y tal que $A$ es casisatisfacible si y sólo si $B$ es casisatisfacible (en adelante, diremos que $A$ y $B$ son equicasisatisfacibles).

En esta sección introducimos los teoremas 4.4, 4.6, 4.5, que nos proporcionan un primer modo de extraer información de los conjuntos Δ. Como podremos comprobar en el resto del trabajo, la información total aportada por los conjuntos Δ es de una utilidad sorprendente y justifica el que lo consideremos como la herramienta fundamental.

**Notación**

Si $A$ es una fnu, denotamos

$$\mathcal{I}_b^{unit}(A) = \{pb' \mid \text{si una interpretación } I \text{ es tal que } I(p) = b' \text{ entonces, } I(A) = b\}$$

donde $b, b' \in \mathbf{3}$

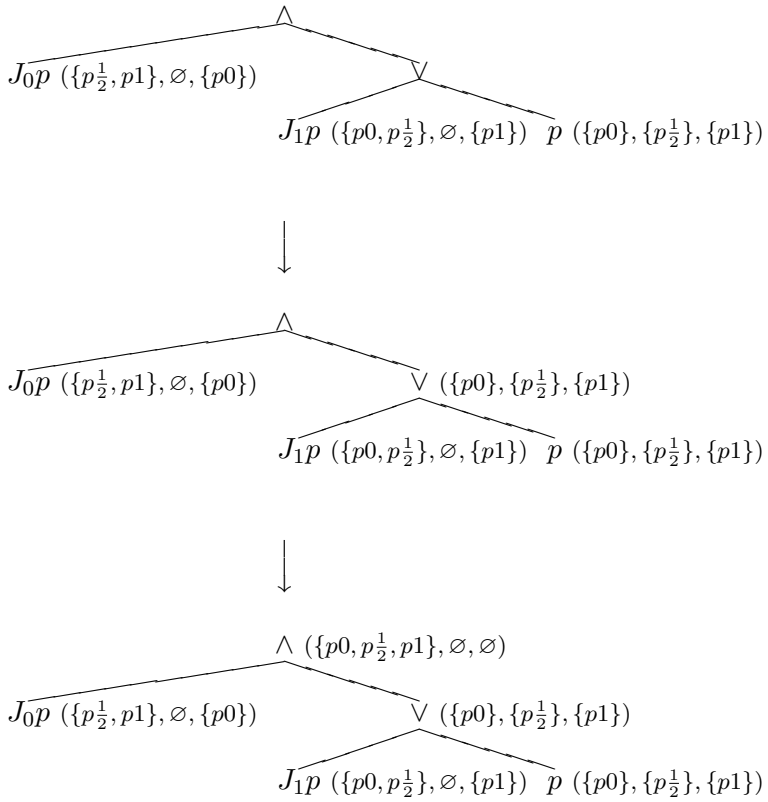



Figura 4.3: Traza del proceso *Etiquetar*

### 4.7.1. Información en $\Delta_0$:

El siguiente lema nos muestra el contenido semántico del conjunto $\Delta_0$:

**Lema 4.3** *Si $A$ es una fnu, entonces $\Delta_0(A) = \mathcal{I}_0^{unit}(A)$, es decir,*

$$\Delta_0(A) = \{pb \mid \text{ si una interpretación } I \text{ es tal que } I(p) = b \text{ entonces, } I(A) = 0\}$$

DEMOSTRACIÓN: Lo demostramos por inducción estructural.

- Si $A$ es un literal es inmediato.
- Si $A = A_1 \wedge A_2$, por definición de $\Delta_0$ se tiene que $\Delta_0(A) = \Delta_0(A_1) \cup \Delta_0(A_2)$. Por otra parte, es obvio que $\mathcal{I}_0^{unit}(A_1) \cup \mathcal{I}_0^{unit}(A_2) = \mathcal{I}_0^{unit}(A)$. Por lo tanto, por hipótesis de inducción, se tiene que

$$\Delta_0(A) \quad = \quad \mathcal{I}_0^{unit}(A_1) \cup \mathcal{I}_0^{unit}(A_2) \quad = \quad \mathcal{I}_0^{unit}(A)$$

- Si $A = A_1 \vee A_2$, por definición de $\Delta_0$ se tiene que $\Delta_0(A) = \Delta_0(A_1) \cap \Delta_0(A_2)$ y puesto que $\mathcal{I}_0^{unit}(A_1) \cap \mathcal{I}_0^{unit}(A_2) = \mathcal{I}_0^{unit}(A)$, se tiene, por hipótesis de inducción, que

$$\Delta_0(A) \quad = \quad \mathcal{I}_0^{unit}(A_1) \cap \mathcal{I}_0^{unit}(A_2) \quad = \quad \mathcal{I}_0^{unit}(A)$$

*q.e.d.*

**Teorema 4.4** *Sea $A$ una fnu y $p$ un símbolo proposicional, entonces:*

| | | | |
|---|---|---|---|
| (1) | $p0 \in \Delta_0(A)$ | *si y sólo si* | $A \equiv A \wedge \neg J_0 p$ |
| (2) | $p\frac{1}{2} \in \Delta_0(A)$ | *si y sólo si* | $A \equiv A \wedge \neg J_{\frac{1}{2}} p$ |
| (3) | $p1 \in \Delta_0(A)$ | *si y sólo si* | $A \equiv A \wedge \neg J_1 p$ |
| (4) | $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$ | *si y sólo si* | $A \equiv A \wedge J_1 p$ |
| (5) | $\{p0, p1\} \subseteq \Delta_0(A)$ | *si y sólo si* | $A \equiv A \wedge J_{\frac{1}{2}} p$ |
| (6) | $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$ | *si y sólo si* | $A \equiv A \wedge J_0 p$ |
| (7) | *Si* $\{p0, p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$ | *entonces* | $A \equiv \bot$ |

Demostración:

(I) En los apartados (1) a (6), que las equivalencias afirmadas a la derecha aseguran las afirmaciones sobre $\Delta_0$ establecidas a la izquierda, es consecuencia inmediata de la definición de $\Delta_0$ para los literales y para $\wedge$, y del Lema 4.3. Desarrollamos la demostración para el ítem (1). La demostración para los restantes es análoga.

Si $A \equiv A \wedge \neg J_0 p$, puesto que $p0 \in \Delta_0(\neg J_0 p)$, se tiene que $p0 \in \Delta_0(A \wedge \neg J_0 p)$ y, por el Lema 4.3, $p0 \in \mathcal{I}_0^{unit}(A \wedge \neg J_0 p)$. Ahora, por ser $A \equiv A \wedge \neg J_0 p$, podemos asegurar que $p0 \in \mathcal{I}_0^{unit}(A)$ y, de nuevo por el Lema 4.3, $p0 \in \Delta_0(A)$

(II) Demostramos ahora, para cada item, que las afirmaciones sobre $\Delta_0$ establecidas a la izquierda aseguran las equivalencias afirmadas a la derecha.

1. Suponemos que $p0 \in \Delta_0(A)$. Sea $I$ una interpretación arbitraria. Hemos de considerar dos casos:

   - Si $I(p) = 0$ entonces el lema 4.3 asegura que $I(A) = 0$. Por tanto, $I(A) = I(A \wedge \neg J_0 p) = 0$.
   - Si $I(p) \in \{\frac{1}{2}, 1\}$ entonces $I(\neg J_0 p) = 1$. Por tanto, $I(A) = I(A \wedge \neg J_0 p)$.

2. Suponemos que $p\frac{1}{2} \in \Delta_0(A)$. Sea $I$ una interpretación arbitraria. Hemos de considerar dos casos:

   - Si $I(p) = \frac{1}{2}$ entonces el lema 4.3 asegura que $I(A) = 0$. Por tanto, $I(A) = I(A \wedge \neg J_{\frac{1}{2}}p) = 0$.
   - Si $I(p) \in \{0, 1\}$ entonces $I(\neg J_{\frac{1}{2}}p) = 1$. Por tanto, $I(A) = I(A \wedge \neg J_{\frac{1}{2}}p)$.

3. Suponemos que $p1 \in \Delta_0(A)$. Sea $I$ una interpretación arbitraria. Hemos de considerar dos casos:

   - Si $I(p) = 1$ entonces el lema 4.3 asegura que $I(A) = 0$. Por tanto, $I(A) = I(A \wedge \neg J_1 p) = 0$.
   - Si $I(p) \in \{0, \frac{1}{2}\}$ entonces $I(\neg J_1 p) = 1$. Por tanto, $I(A) = I(A \wedge \neg J_1 p)$.

4. Suponemos que $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$. Sea $I$ una interpretación arbitraria. Hemos de considerar dos casos:

   - Si $I(p) \in \{0, \frac{1}{2}\}$ entonces el lema 4.3 asegura que $I(A) = 0$. Por tanto, $I(A) = I(A \wedge J_1 p) = 0$.
   - Si $I(p) = 1$ entonces $I(J_1 p) = 1$. Por tanto, $I(A) = I(A \wedge J_1 p)$.

5. Suponemos que $\{p0, p1\} \subseteq \Delta_0(A)$. Sea $I$ una interpretación arbitraria. Hemos de considerar tres casos:

   - Si $I(p) \in \{0, 1\}$ entonces el lema 4.3 asegura que $I(A) = 0$. Por tanto, $I(A) = I(A \wedge J_{\frac{1}{2}}p) = 0$.
   - Si $I(p) = \frac{1}{2}$ entonces $I(J_{\frac{1}{2}}p) = 1$. Por tanto, $I(A) = I(A \wedge J_{\frac{1}{2}}p)$.

6. Suponemos que $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$. Sea $I$ una interpretación arbitraria. Hemos de considerar tres casos:

   - Si $I(p) \in \{\frac{1}{2}, 1\}$ entonces el lema 4.3 asegura que $I(A) = 0$. Por tanto, $I(A) = I(A \wedge J_0 p) = 0$.
   - Si $I(p) = 0$ entonces $I(J_0 p) = 1$. Por tanto, $I(A) = I(A \wedge J_0 p)$.

7. Es inmediato por el lema 4.3 y la semántica de $\perp$.

*q.e.d.*

### 4.7.2. Información en $\Delta_1$:

El siguiente lema nos muestra el contenido semántico del conjunto $\Delta_1$:

**Lema 4.4** *Sea $A$ una fnu, entonces $\Delta_1(A) = \mathcal{I}_1^{unit}(A)$, es decir*

$\Delta_1(A) = \{pb \mid$ *si una interpretación $I$ es tal que $I(p) = b$ entonces, $I(A) = 1\}$*

Demostración: Lo demostramos por inducción estructural.

- Si $A$ es un literal es inmediato.
- Si $A = A_1 \wedge A_2$, por definición de $\Delta_1$ se tiene que $\Delta_1(A) = \Delta_1(A_1) \cap \Delta_1(A_2)$. Por otra parte, es obvio que $\mathcal{I}_1^{unit}(A_1) \cap \mathcal{I}_1^{unit}(A_2) = \mathcal{I}_1^{unit}(A)$. Por lo tanto, por hipótesis de inducción, se tiene que

$$\Delta_1(A) \quad = \quad \mathcal{I}_1^{unit}(A_1) \cap \mathcal{I}_1^{unit}(A_2) \quad = \quad \mathcal{I}_1^{unit}(A)$$

- Si $A = A_1 \vee A_2$, por definición de $\Delta_1$ se tiene que $\Delta_1(A) = \Delta_1(A_1) \cup \Delta_1(A_2)$ y puesto que $\mathcal{I}_1^{unit}(A_1) \cup \mathcal{I}_1^{unit}(A_2) = \mathcal{I}_1^{unit}(A)$, se tiene, por hipótesis de inducción, que

$$\Delta_1(A) \quad = \quad \mathcal{I}_1^{unit}(A_1) \cup \mathcal{I}_1^{unit}(A_2) \quad = \quad \mathcal{I}_1^{unit}(A)$$

*q.e.d.*

**Teorema 4.5** *Sea $A$ una fnu y $p$ un símbolo proposicional, entonces:*

| | | | |
|---|---|---|---|
| (1) | $p0 \in \Delta_1(A)$ | *si y sólo si* | $A \equiv A \vee J_0 p$ |
| (2) | $p\frac{1}{2} \in \Delta_1(A)$ | *si y sólo si* | $A \equiv A \vee J_{\frac{1}{2}} p$ |
| (3) | $p1 \in \Delta_1(A)$ | *si y sólo si* | $A \equiv A \vee J_1 p$ |
| (4) | $\{p0, p\frac{1}{2}\} \subseteq \Delta_1(A)$ | *si y sólo si* | $A \equiv A \vee \neg J_1 p$ |
| (5) | $\{p0, p1\} \subseteq \Delta_1(A)$ | *si y sólo si* | $A \equiv A \vee \neg J_{\frac{1}{2}} p$ |
| (6) | $\{p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$ | *si y sólo si* | $A \equiv A \vee \neg J_0 p$ |
| (7) | *Si* $\{p0, p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$ | *entonces* | $A \equiv \top$ |

Demostración: La demostración se obtiene paso a paso por dualidad de la demostración del teorema 4.4 a partir del lema 4.4. *q.e.d.*

### 4.7.3. Información en $\Delta_{\frac{1}{2}}$:

Al igual que para los otros conjuntos $\Delta$, el siguiente lema nos muestra el contenido semántico del conjunto $\Delta_{\frac{1}{2}}$:

**Lema 4.5** *Sea $A$ una fnu, entonces $\Delta_{\frac{1}{2}}(A) = \mathcal{I}^{unit}_{\frac{1}{2}}(A)$, es decir,*

$$\Delta_{\frac{1}{2}}(A) = \{pb \mid \text{ si una interpretación } I \text{ es tal que } I(p) = b \text{ entonces, } I(A) = \tfrac{1}{2}\}$$

Demostración: Lo demostramos por inducción estructural.

- Si $A$ es un literal es inmediato.
- Si $A = A_1 \wedge A_2$, por definición de $\Delta_{\frac{1}{2}}$ se tiene que

  $$\Delta_{\frac{1}{2}}(A_1 \wedge A_2) = [\Delta_{\frac{1}{2}}(A_1) \cap (\Delta_{\frac{1}{2}}(A_2) \cup \Delta_1(A_2))] \cup (\Delta_1(A_1) \cap \Delta_{\frac{1}{2}}(A_2))$$

  Por otra parte, es obvio que

  $$[\mathcal{I}^{unit}_{\frac{1}{2}}(A_1) \cap (\mathcal{I}^{unit}_{\frac{1}{2}}(A_2) \cup \mathcal{I}^{unit}_{1}(A_2))] \cup (\mathcal{I}^{unit}_{1}(A_1) \cap \mathcal{I}^{unit}_{\frac{1}{2}}(A_2)) = \mathcal{I}^{unit}_{\frac{1}{2}}(A)$$

  Por lo tanto, por el Lema 4.4 y por la hipótesis de inducción, se tiene que

  $$\begin{array}{rcl} \Delta_{\frac{1}{2}}(A) & = & [\mathcal{I}^{unit}_{\frac{1}{2}}(A_1) \cap (\mathcal{I}^{unit}_{\frac{1}{2}}(A_2) \cup \mathcal{I}^{unit}_{1}(A_2))] \cup (\mathcal{I}^{unit}_{1}(A_1) \cap \mathcal{I}^{unit}_{\frac{1}{2}}(A_2)) \\ & = & \mathcal{I}^{unit}_{\frac{1}{2}}(A) \end{array}$$

- Si $A = A_1 \vee A_2$, por definición de $\Delta_1$ se tiene que

  $$\Delta_{\frac{1}{2}}(A_1 \vee A_2) = [\Delta_{\frac{1}{2}}(A_1) \cap (\Delta_{\frac{1}{2}}(A_2) \cup \Delta_0(A_2))] \cup (\Delta_0(A_1) \cap \Delta_{\frac{1}{2}}(A_2))$$

  Por otra parte, es obvio que

  $$[\mathcal{I}^{unit}_{\frac{1}{2}}(A_1) \cap (\mathcal{I}^{unit}_{\frac{1}{2}}(A_2) \cup \mathcal{I}^{unit}_{0}(A_2))] \cup (\mathcal{I}^{unit}_{0}(A_1) \cap \mathcal{I}^{unit}_{\frac{1}{2}}(A_2)) = \mathcal{I}^{unit}_{\frac{1}{2}}(A)$$

  Por lo tanto, por el Lema 4.3 y por la hipótesis de inducción, se tiene que

  $$\begin{array}{rcl} \Delta_{\frac{1}{2}}(A) & = & [\mathcal{I}^{unit}_{\frac{1}{2}}(A_1) \cap (\mathcal{I}^{unit}_{\frac{1}{2}}(A_2) \cup \mathcal{I}^{unit}_{0}(A_2))] \cup (\mathcal{I}^{unit}_{0}(A_1) \cap \mathcal{I}^{unit}_{\frac{1}{2}}(A_2)) \\ & = & \mathcal{I}^{unit}_{\frac{1}{2}}(A) \end{array}$$

*q.e.d.*

**Teorema 4.6** *Sea A una fnu y p un símbolo proposicional, entonces:*

$$\text{Si } pb \in \Delta_{\frac{1}{2}}(A) \text{ entonces } b = \frac{1}{2}$$

DEMOSTRACIÓN:

Basta observar que:

(i) $\Delta_{\frac{1}{2}}$ aplicado a cualquier literal no contiene a $p0$ ni a $p1$.

(ii) $\Delta_{\frac{1}{2}}$ de la unión o intersección de varias subfórmulas es siempre un subconjunto de la unión de los $\Delta_{\frac{1}{2}}$ de dichas subfórmulas, por tanto ni $p0$ ni $p1$ pueden estar en $\Delta_{\frac{1}{2}}(A)$.

*q.e.d.*

## 4.8. El proceso $\mathcal{F}$

Las definiciones y resultados de la sección anterior se utilizan en el proceso $\mathcal{F}$ con el objetivo de disminuir la complejidad de la fórmula antes de distribuir. Concretamente, esta distribución, "filtra" la información de los conjuntos $\Delta$ para evitar tantas distribuciones de $\wedge$ sobre $\vee$ como sea posible.

El diagrama de flujo de este proceso aparece en la Figura **??**.

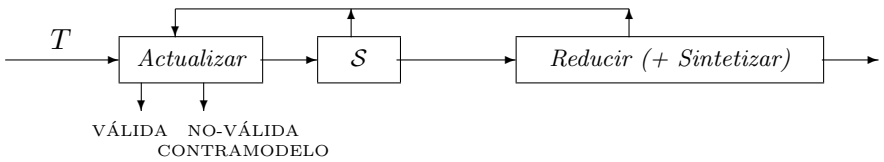



Figura 4.4: El proceso $\mathcal{F}$

Como puede verse en la figura **??**, la entrada de TAS-M**3** es el árbol sintáctico de la negación fuerte de la fórmula cuya validez se quiere determinar. Tras el proceso *Signar*, el árbol se ha transformado en el árbol sintáctico de una fórmula en forma normal unaria. Este árbol es la entrada del proceso $\mathcal{F}$, que actúa sobre la fórmula, reduciendo el tamaño de la misma o dando como salida el mensaje de fórmula VÁLIDA, o de fórmula NO VÁLIDA. En este último caso proporciona además un contramodelo, es decir una interpretación que asocia a la fórmula analizada el valor de verdad 0 o bien el valor de verdad $\frac{1}{2}$.

Pasamos ahora a describir el proceso $\mathcal{F}$.

### 4.8.1. El test de finalizabilidad

**Definición 4.10** Decimos que una fnu $A$ es *finalizable* si satisface una de las condiciones siguientes:

- $A$ es una de las constantes $\bot$, $\oslash$ o $\top$.
- $\Delta_{\frac{1}{2}}(A) \neq \varnothing$.
- $\Delta_1(A) \neq \varnothing$.

**Teorema 4.7** *Si una fnu $A$ es finalizable se tiene que:*

1. *Si $A$ es $\bot$ entonces $\neg A$ es válida*
2. *En otro caso, $A$ es casisatisfacible, es decir, $\neg A$ no es válida.*

Demostración:

1. Es inmediato.
2. *a*) Si $A$ es $\oslash$ o $\top$, es inmediato.
   *b*) Si $\Delta_{\frac{1}{2}}(A) \neq \varnothing$ y $pb \in \Delta_{\frac{1}{2}}(A)$, por el lema 4.5, toda interpretación $I$ tal que $I(p) = b$ verifica que $I(A) = \frac{1}{2}$. Por lo tanto, $A$ es casisatisfacible.
   *c*) Si $\Delta_1(A) \neq \varnothing$ y $pb \in \Delta_1(A)$, por el lema 4.4, toda interpretación $I$ tal que $I(p) = b$ verifica que $I(A) = 1$. Por lo tanto, $A$ es casisatisfacible.

*q.e.d.*

### 4.8.2. El proceso *Simplificar* o de reducción fuerte

El objetivo de este proceso es detectar, a partir de la información contenida en los conjuntos $\Delta$, la posibilidad de reducir drásticamente el tamaño de la fórmula. Específicamente, este proceso permitirá:

- Sustituir una subfórmula (posiblemente la fórmula completa) por la constante $\top$, por la constante $\bot$ o por un literal $l$. Las fórmulas en las que son posibles estas sustituciones las llamaremos 0-concluyentes, 1-concluyentes y $l$-simplificables, respectivamente.
- Podar una rama en el árbol sintáctico de la fórmula, en las fórmulas que llamaremos *podables*.

La posibilidad de realizar esta fuerte reducción de tamaño la proporcionan los Teoremas 4.4, 4.5 y 4.6 anteriores

Para definir formalmente el concepto de *Simplificar* necesitamos introducir algunas definiciones y resultados:

**Definición 4.11** Un *cubo trivaluado* es un literal trivaluado o una conjunción de literales trivaluados.

Una *cláusula trivaluada* es un literal trivaluado o una disyunción de literales trivaluados.

Una fbf se dice que está en *forma normal disyuntiva* (abreviadamente fnd) si es un cubo trivaluado o una disyunción de cubos trivaluados.

**Lema 4.6** *Para toda fnu $A$ en $M\mathbf{3}$ existe una fnu en forma normal disyuntiva $A'$ tal que*

1. $A \equiv A'$
2. *Para cada literal $l$ se tiene que*

$$l \text{ ocurre en } A \text{ si y sólo si } l \text{ ocurre en } A'.$$

Demostración: Es consecuencia inmediata de la ley distributiva de $\wedge$ sobre $\vee$ y del Teorema de Equivalencia 2.1. *q.e.d.*

**Definición 4.12** Un cubo trivaluado se dice *restringido* si no contiene dos literales con el mismo sufijo.

Una cláusula trivaluada se dice *restringida* si no contiene dos literales con el mismo sufijo.

En términos de árboles sintácticos, una cláusula trivaluada $C$ es restringida si:

$|\{p \in Q \mid pb \in \Delta_1(C) \text{ para algún } b \in \mathbf{3}\}| =$ grado de ramificación de la raíz de $T_C$

Análogamente, un cubo trivaluado $D$ es restringido si:

$|\{p \in Q \mid pb \in \Delta_0(D) \text{ para algún } b \in \mathbf{3}\}| =$ grado de ramificación de la raíz de $T_D$

**Notación**

- Si $p$ es un símbolo proposicional, $P_{\mathbf{3}}$ denota el conjunto $\{p0, p\frac{1}{2}, p1\}$.
- $\bar{0} = 1$, y $\bar{1} = 0$.

- Sea $l$ un literal trivaluado cuyo sufijo es $p$. Entonces denotamos

$$\overline{\Delta_b(l)} = \begin{cases} \{p\overline{b'}\} & \text{si } l = p \text{ o } l = \neg p \text{ y } \Delta_b(l) = \{pb'\} \\ P_{\mathbf{3}} - \Delta_b(l), & \text{en otro caso} \end{cases}$$

- Si $C = \bigvee_{i=1}^{n} l_i$ es una cláusula restringida, denotamos $\overline{\Delta_1(C)} = \bigcup_{i=1}^{n} \overline{\Delta_1(l_i)}$
- Si $D = \bigwedge_{i=1}^{n} l_i$ es un cubo restringido, denotamos $\overline{\Delta_0(D)} = \bigcup_{i=1}^{n} \overline{\Delta_0(l_i)}$

**Ejemplo 4.3** Para la cláusula restringida $C = p \vee \neg J_1 q \vee \neg r$,

$$\Delta_1(C) = \{p1, q0, q\frac{1}{2}, r0\}, \qquad \overline{\Delta_1(C)} = \{p0, q1, r1\}$$

**Lema 4.7**

1. *Si $l$ es un literal trivaluado y $b \in \{0,1\}$, entonces $\overline{\Delta_b(l)} = \Delta_{\bar{b}}(l)$.*
2. *Si $C = \bigvee_{i=1}^{n} l_i$ es una cláusula restringida, entonces $\overline{\Delta_1(C)} = \Delta_0(\bigwedge_{i=1}^{n} l_i)$*
3. *Si $D = \bigwedge_{i=1}^{n} l_i$ es un cubo restringido, entonces $\overline{\Delta_0(D)} = \Delta_0(\bigvee_{i=1}^{n} l_i)$*

DEMOSTRACIÓN:

1. es consecuencia inmediata de la definición de $\Delta_0$ y $\Delta_1$ para los literales trivaluados.

2. y 3. se siguen de 1 y de la definición de $\Delta_0$ y $\Delta_1$ para $\vee$ y $\wedge$.

*q.e.d.*

**Ejemplo 4.4** Para la cláusula restringida $C = p \vee \neg J_1 q \vee \neg r$ del ejemplo 4.3:

$$\overline{\Delta_1(C)} = \{p0, q1, r1\} = \Delta_0(p \wedge \neg J_1 q \wedge \neg r)$$

Para la cláusula restringida $C = p \vee J_1 q \vee \neg r$:

$$\overline{\Delta_1(C)} = \{p0, q0, q\frac{1}{2}, r1\} = \Delta_0(p \wedge J_1 q \wedge \neg r)$$

**Definición 4.13** Sea $A$ una fnu, diremos que

1. $A$ es *0-concluyente* si satisface una de las dos condiciones siguientes:
   a) Existe $p \in Q$ tal que $P_{\mathbf{3}} \subseteq \Delta_0(A)$.
   b) $A$ es de la forma $\bigwedge_{i=1}^{n} A_i$, algún $A_{i_0}$ con $1 \leq i_0 \leq n$ es una cláusula trivaluada restringida no literal y, para todo símbolo proposicional $p$ tal que $pb \in \Delta_1(A_{i_0})$ para algún $b \in \mathbf{3}$, se tiene que $P_{\mathbf{3}} - \overline{\Delta_1(A_{i_0})} \subseteq \Delta_0(A)$.
2. $A$ es *1-concluyente* si satisface una de las dos condiciones siguientes:
   a) Existe $p \in Q$ tal que $P_{\mathbf{3}} \subseteq \Delta_1(A)$.
   b) $A$ es de la forma $\bigvee_{i=1}^{n} A_i$, algún $A_{i_0}$ con $1 \leq i_0 \leq n$ es un cubo trivaluado restringido no literal y, para todo símbolo proposicional $p$ tal que $pb \in \Delta_0(A_{i_0})$ para algún $b \in \mathbf{3}$, se tiene que $P_{\mathbf{3}} - \overline{\Delta_0(A_{i_0})} \subseteq \Delta_1(A)$.
3. $A$ es $p$-simple si satisface las dos condiciones siguientes:
   a) $A$ no es un literal trivaluado ni una constante.
   b) $\Delta_0(A) \neq \varnothing$ y $\Delta_1(A) \neq \varnothing$.
   c) Existe $p \in Q$ tal que $\Delta_0(A) \cup \Delta_{\frac{1}{2}}(A) \cup \Delta_1(A) = P_{\mathbf{3}}$.
4. $A$ es *podable* si no es un literal trivaluado y satisface una de las condiciones siguientes:
   a) $A = \bigwedge_{i=1}^{n} A_i$ con algún $A_{i_0}$, $0 \leq i_0 \leq n$, tal que $\Delta_0(A_{i_0}) = \varnothing$ y existe un símbolo proposicional $p$ tal que $P_{\mathbf{3}} - \Delta_1(A_{i_0}) \subseteq \Delta_0(A)$.
   b) $A = \bigvee_{i=1}^{n} A_i$ con algún $A_{i_0}$, $0 \leq_{i_0} \leq n$, tal que $\Delta_1(A_{i_0}) = \varnothing$ y existe un símbolo proposicional $p$ tal que $P_{\mathbf{3}} - \Delta_0(A_{i_0}) \subseteq \Delta_1(A)$.

   En ambos casos decimos que $A$ es $A_{i_0}$-podable.
5. Diremos que una fnu $A$ es *simplificable* si tiene algún subárbol 0-concluyente, 1-concluyente, $p$-simple o podable.

El siguiente teorema confirma que las definiciones dadas tienen la utilidad esperada.

**Teorema 4.8** *Dada una fnu $A$, se tiene que:*

1. *Si* $A$ *es* 0-concluyente *entonces* $A \equiv \bot$.
2. *Si* $A$ *es* 1-concluyente *entonces* $A \equiv \top$.
3. *Si* $A$ *es* p-simple *se tiene que:*

(a) *Si* $\Delta_0(A) = \{p0\}$, $\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$, $\Delta_1(A) = \{p1\}$ *si y sólo si* $A \equiv p$
(b) *Si* $\Delta_0(A) = \{p1\}$, $\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$, $\Delta_1(A) = \{p0\}$ *si y sólo si* $A \equiv \neg p$
(c) *Si* $\Delta_0(A) = \{p\frac{1}{2}, p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$, $\Delta_1(A) = \{p0\}$ *si y sólo si* $A \equiv J_0 p$
(d) *Si* $\Delta_0(A) = \{p0\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$, $\Delta_1(A) = \{p\frac{1}{2}, p1\}$ *si y sólo si* $A \equiv \neg J_0 p$
(e) *Si* $\Delta_0(A) = \{p0, p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$, $\Delta_1(A) = \{p\frac{1}{2}\}$ *si y sólo si* $A \equiv J_{\frac{1}{2}} p$
(f) *Si* $\Delta_0(A) = \{p_{\frac{1}{2}}\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$, $\Delta_1(A) = \{p0, p1\}$ *si y sólo si* $A \equiv \neg J_{\frac{1}{2}} p$
(g) *Si* $\Delta_0(A) = \{p0, p\frac{1}{2}\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$, $\Delta_1(A) = \{p1\}$ *si y sólo si* $A \equiv J_1 p$
(h) *Si* $\Delta_0(A) = \{p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$, $\Delta_1(A) = \{p0, p\frac{1}{2}\}$ *si y sólo si* $A \equiv \neg J_1 p$

4. *Si* $A = \bigwedge_{i \in J} A_i$ *es* $A_{i_0}$*-podable, entonces* $A \equiv \bigwedge i \in J i \neq i_0 A_i$.
5. *Si* $A = \bigvee_{i \in J} A_i$ *es* $A_{i_0}$*-podable, entonces* $A \equiv \bigvee i \in J i \neq i_0 A_i$.

Demostración:

1. Sea $A$ una fnu 0-concluyente (para 1-concluyente la demostración es análoga).

   El caso $(a)$, es decir, si existe $p \in Q$ tal que $P_{\mathbf{3}} = \{p0, p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$, es el último ítem del teorema 4.4 y ha sido ya demostrado.

   Supongamos que $A = \bigwedge_{i=1}^{n} A_i$ y sea $A_{i_0}$ para algún $i_0$ con $1 \leq i_0 \leq n$ una cláusula trivaluada restringida, $A_{i_0} = \bigvee_{j=1}^{m} l_j$ donde dos cualesquiera de los literales trivaluados $l_j$ tienen sufijos diferentes. Entonces,

   $$A \quad = \quad \bigwedge_{i=1}^{n} A_i \equiv \bigwedge_{i \neq i_0} A_i \wedge A_{i_0} = \bigwedge_{i \neq i_0} A_i \wedge \bigvee_{j=1}^{m} l_j \equiv \bigvee_{j=1}^{m} (\bigwedge_{i \neq i_0} A_i \wedge l_j)$$

   Por ser $A_{i_0}$ una cláusula trivaluada restringida no literal, $\Delta_0(A_{i_0}) = \varnothing$ y por lo tanto, $\Delta_0(\bigwedge_{i=1}^{n} A_i) = \Delta_0(\bigwedge_{i \neq i_0} A_i)$. Por otra parte, para cada símbolo

proposicional $p$ que interviene en $\Delta_1(A_{i_0})$ la contribución de $p$ procede de un sólo literal $l = \alpha p$, es decir,

$$P_{\mathbf{3}} \cap \Delta_1(l) = P_{\mathbf{3}} \cap \Delta_1(A_{i_0})$$

Ahora, por hipótesis, se tiene que $P_{\mathbf{3}} - \overline{\Delta_1(A_{i_0})} \subseteq \Delta_0(A)$, y por el lema 4.7 se tiene que $\overline{\Delta_1(l)} = \Delta_0(l)$. Por lo tanto,

$$P_{\mathbf{3}} - \Delta_0(l) = P_{\mathbf{3}} - \overline{\Delta_1(l)}$$

Así pues,

$$P_{\mathbf{3}} - \Delta_0(l_j) \subseteq \Delta_0(A)$$

y podemos asegurar que cada uno de los elementos de la disyunción:

$$(\bigwedge_{i \neq i_0} A_i) \wedge l$$

satisface las condiciones del ítem $(a)$. Se tiene pues que $(\bigwedge_{i \neq i_0} A_i) \wedge l \equiv \bot$ para todo literal $l$ en $A_{i_0}$ y, por tanto, $A \equiv \bot$.

3. En todos los apartados, que las equivalencias afirmadas a la derecha aseguran las afirmaciones sobre los conjuntos $\Delta$ establecidas a la izquierda, es consecuencia inmediata de la definición de los conjuntos $\Delta$ para los literales trivaluados y de los lemas 4.3, 4.4 y 4.5.

   Demostramos ahora, para cada item, que las afirmaciones sobre los conjuntos $\Delta$ establecidas a la izquierda aseguran las equivalencias afirmadas a la derecha.

   (a) Si $\Delta_0(A) = \{p0\}$, $\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$ y $\Delta_1(A) = \{p1\}$ entonces, por los lemas 4.3, 4.4 y 4.5, para toda interpretación $I$, se tiene que $I(p) = b$ con $b \in \mathbf{3}$ asegura que $I(A) = b$. Por tanto $A \equiv p$.

   (b) Si $\Delta_0(A) = \{p1\}$, $\Delta_{\frac{1}{2}}(A) = \{p\frac{1}{2}\}$ y $\Delta_1(A) = \{p0\}$ entonces, por los lemas 4.3, 4.4 y 4.5, para toda interpretación $I$, se tiene que $I(\neg p) = b$ con $b \in \mathbf{3}$ asegura que $I(A) = b$. Por tanto, $A \equiv \neg p$.

   (c) Suponemos que $\Delta_0(A) = \{p\frac{1}{2}, p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ y $\Delta_1(A) = \{p0\}$.

   Ahora, de $\Delta_0(A) = \{p\frac{1}{2}, p1\}$ y el teorema 4.4, se tiene que $A \equiv A \wedge J_0 p$ y de $\Delta_1(A) = \{p0\}$ y el teorema 4.5, se tiene que $A \equiv A \vee J_0 p$ Por lo tanto, $A \equiv A \wedge J_0 p \equiv (A \vee J_0 p) \wedge J_0 p \equiv J_0 p$.

(d) Suponemos que $\Delta_0(A) = \{p0\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ y $\Delta_1(A) = \{p\frac{1}{2}, p1\}$.
Ahora, de $\Delta_0(A) = \{p0\}$ y el teorema 4.4, se tiene que $A \equiv A \wedge \neg J_0 p$ y de $\Delta_1(A) = \{p\frac{1}{2}, p1\}$ y el teorema 4.5, se tiene que $A \equiv A \vee \neg J_0 p$ Por lo tanto, $A \equiv A \wedge \neg J_0 p \equiv (A \vee \neg J_0 p) \wedge \neg J_0 p \equiv \neg J_0 p$.

(e) Suponemos que $\Delta_0(A) = \{p0, p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ y $\Delta_1(A) = \{p\frac{1}{2}\}$.
Ahora, de $\Delta_0(A) = \{p0, p1\}$ y el teorema 4.4, se tiene que $A \equiv A \wedge J_{\frac{1}{2}} p$ y de $\Delta_1(A) = \{p\frac{1}{2}\}$ y el teorema 4.5, se tiene que $A \equiv A \vee J_{\frac{1}{2}} p$ Por lo tanto, $A \equiv A \wedge J_{\frac{1}{2}} p \equiv (A \vee J_{\frac{1}{2}} p) \wedge J_{\frac{1}{2}} p \equiv J_{\frac{1}{2}} p$.

(f) Suponemos que $\Delta_0(A) = \{p\frac{1}{2}\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ y $\Delta_1(A) = \{p0, p1\}$.
Ahora, de $\Delta_0(A) = \{p\frac{1}{2}\}$ y el teorema 4.4, se tiene que $A \equiv A \wedge \neg J_{\frac{1}{2}} p$ y de $\Delta_1(A) = \{p0, p1\}$ y el teorema 4.5, se tiene que $A \equiv A \vee \neg J_{\frac{1}{2}} p$ Por lo tanto, $A \equiv A \wedge \neg J_{\frac{1}{2}} p \equiv (A \vee \neg J_{\frac{1}{2}} p) \wedge \neg J_{\frac{1}{2}} p \equiv \neg J_{\frac{1}{2}} p$.

(g) Suponemos que $\Delta_0(A) = \{p0, p\frac{1}{2}\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ y $\Delta_1(A) = \{p1\}$.
Ahora, de $\Delta_0(A) = \{p0, p\frac{1}{2}\}$ y el teorema 4.4, se tiene que $A \equiv A \wedge J_1 p$ y de $\Delta_1(A) = \{p1\}$ y el teorema 4.5, se tiene que $A \equiv A \vee J_1 p$ Por lo tanto, $A \equiv A \wedge J_1 p \equiv (A \vee J_1 p) \wedge J_1 p \equiv J_1 p$.

(h) Suponemos que $\Delta_0(A) = \{p1\}$, $\Delta_{\frac{1}{2}}(A) = \varnothing$ y $\Delta_1(A) = \{p0, p\frac{1}{2}\}$.
Ahora, de $\Delta_0(A) = \{p1\}$ y el teorema 4.4, se tiene que $A \equiv A \wedge \neg J_1 p$ y de $\Delta_1(A) = \{p0, p\frac{1}{2}\}$ y el teorema 4.5, se tiene que $A \equiv A \vee \neg J_1 p$ Por lo tanto, $A \equiv A \wedge \neg J_1 p \equiv (A \vee \neg J_1 p) \wedge \neg J_1 p \equiv \neg J_1 p$.

4. Debido a la conmutatividad y asociatividad de $\wedge$, basta probar el resultado para una fórmula $A = A_1 \wedge A_2$ que es $A_2$-podable.

   Por la hipótesis $\Delta_0(A_2) = \varnothing$, se tiene que $\Delta_0(A) = \Delta_0(A_1)$.

   Ahora tenemos que considerar 7 casos, todos aquellos en los que se verifica la hipótesis $P_{\mathbf{3}} - \Delta_1(A_2) \subseteq \Delta_0(A)$. Desarrollamos la demostración para dos de ellos. Los restantes se demuestran de modo análogo.

   - Supongamos que $p0 \in \Delta_1(A_2)$ y $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A) = \Delta_0(A_1)$, entonces, por el ítem (6) del Teorema 4.4 y el ítem (1) del Teorema 4.5:

   $$A \equiv A \wedge J_0 p = A_1 \wedge A_2 \wedge J_0 p \equiv A_1 \wedge (A_2 \vee J_0 p) \wedge J_0 p \equiv A_1 \wedge J_0 p \equiv A_1$$

   - Supongamos que $\{p0, p_1\} \subseteq \Delta_1(A_2)$ y $p\frac{1}{2} \in \Delta_0(A) = \Delta_0(A_1)$, entonces, por el ítem (2) del Teorema 4.7 y el ítem (5) del Teorema 4.5:

   $$\begin{aligned} A &\equiv A \wedge \neg J_{\frac{1}{2}} p = A_1 \wedge A_2 \wedge \neg J_{\frac{1}{2}} p \equiv A_1 \wedge (A_2 \vee \neg J_{\frac{1}{2}} p) \wedge \neg J_{\frac{1}{2}} p \\ &\equiv A_1 \wedge \neg J_{\frac{1}{2}} p \equiv A_1 \end{aligned}$$

5. Se demuestra de modo análogo al ítem anterior.

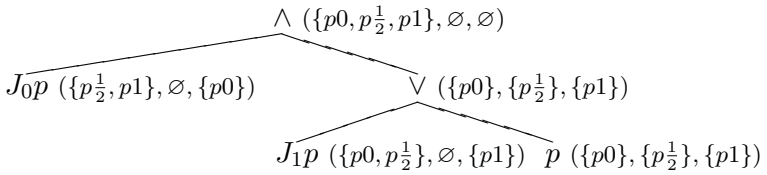
*q.e.d.*

Ahora podemos definir qué entendemos por *Simplificar* un árbol sintáctico de una fnu de M**3**.

**Definición 4.14** Dada una fnu $A$, por *Simplificar* $A$ entenderemos recorrer $T_A$ primero en profundidad, realizando la comprobación de si $T_A$ tiene un subárbol $b$-concluyente, $p$-simple o podable aplicando, en caso afirmativo, las sustituciones indicadas por las equivalencias del teorema 4.8.

**Ejemplo 4.5**

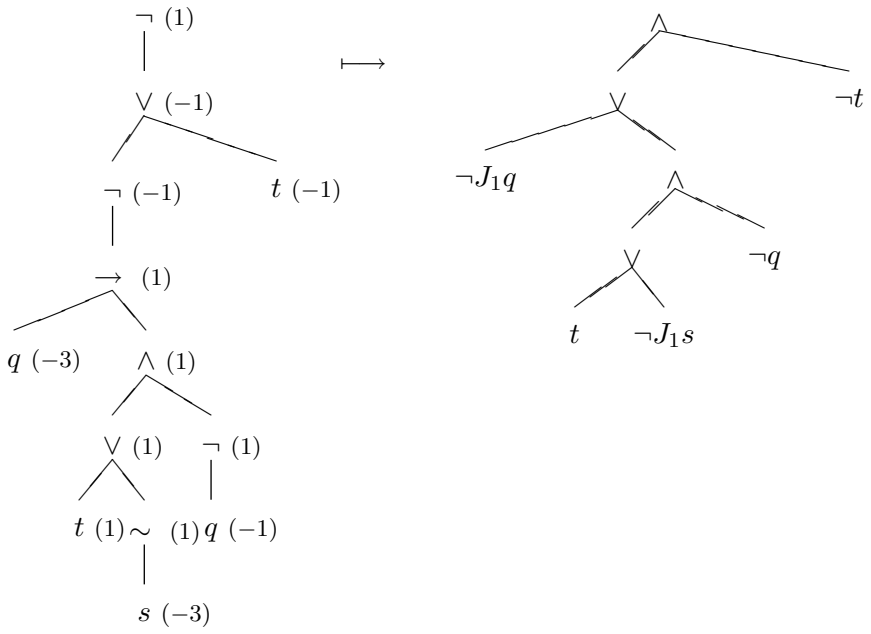
Volviendo al ejemplo 4.2, habíamos obtenido el árbol sintáctico etiquetado $T_B$:



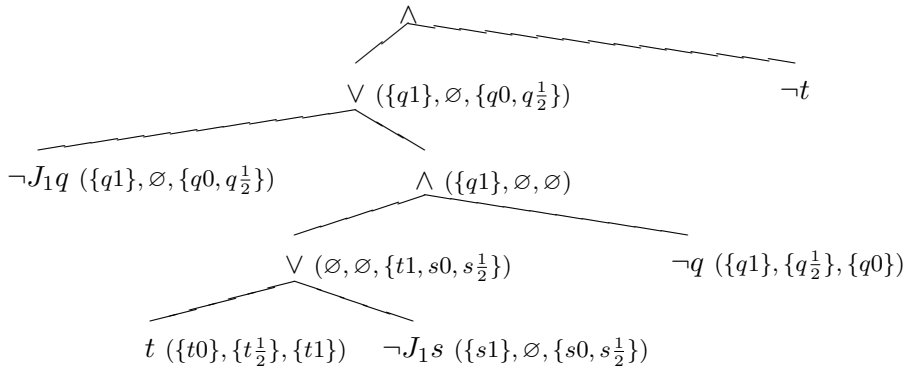

Este árbol $T_B$ es *0-concluyente* ya que $P_{\mathbf{3}} \subseteq \Delta_0(B)$ por tanto al *Simplificar* se obtiene $\bot$, que es finalizable. Por lo tanto, el método TAS-M**3** termina con la salida "VÁLIDA".

Obsérvese que los procesos *Signar* y *Etiquetar* pueden implementarse de una sola vez mediante un proceso recursivo, en la práctica, para determinar que la fórmula $\neg p \rightarrow (\sim p \wedge \neg p)$ es válida basta un solo recorrido del árbol sintáctico correspondiente.

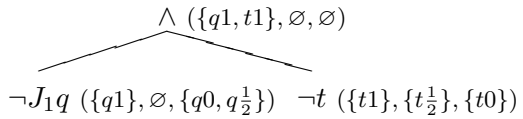
**Ejemplo 4.6** Sea la fórmula $A = \neg(q \rightarrow ((t \vee \sim s) \wedge \neg q)) \vee t$, la traza del proceso *Signar*, aplicada a $T_{\neg A}$ es la siguiente:




Sobre este árbol (única tarea generada al ser su raíz $\wedge$), la transformación $\mathcal{F}$ ejecutará el proceso *Etiquetar* proporcionando el árbol:




En el momento de ejecución del proceso recursivo *Etiquetar* que muestra esta figura, se detecta que la etiqueta $(\Delta_0, \Delta_{\frac{1}{2}}, \Delta_1)$ del nodo $\vee$, hijo izquierdo de la raíz (correspondiente a la fórmula $E = \neg J_1 q \vee ((t \vee \neg J_1 s) \wedge \neg q)$) verifica que $\Delta_0(E) \cup \Delta_{\frac{1}{2}}(E) \cup \Delta_1(E) = \{q1\} \cup \{q0, q\frac{1}{2}\} = Q_{\mathbf{3}}$, $\Delta_0(E) \neq \varnothing$ y $\Delta_1(E) \neq \varnothing$, por tanto $E$ es *q-simple* y su árbol es sustituido, en el proceso *Simplificar*, por el literal trivaluado $\neg J_1 q$ obteniéndose el árbol




El árbol anterior será la entrada del siguiente proceso de la transformación $\mathcal{F}$.

## 4.9. El proceso *Actualizar*

Una vez dada la definición de forma normal unaria *finalizable* y descrito el subproceso *Simplificar*, pasamos a describir el proceso *Actualizar* que, como podemos ver en la Figura **??**, puede intervenir de dos modos diferentes en la traza de ejecución de $\mathcal{F}$: para tratar la entrada a $\mathcal{F}$ o bien para tratar la salida de los procesos $\mathcal{S}$, *Reducir* y *Sintetizar* (que introduciremos en las secciones 4.10 y 4.11).

Las acciones ejecutadas por este proceso son de dos tipos:

- Etiquetar: o bien el árbol completo, si es la primera vez que actúa, o bien etiquetar los nodos modificados por algún otro proceso, si actúa sobre la salida de los procesos $\mathcal{S}$, *Reducir* y *Sintetizar*. Ahora bien, en aras de la eficiencia, puesto que tanto el test de finalizabilidad como el proceso *Simplificar* tan sólo requieren para su ejecución la lectura de la etiqueta, tan pronto como se calcula una etiqueta, *Actualizar* procede a la verificación de si ésta conduce a la finalizabilidad o bien es simplificable. De esta forma, podemos o bien terminar el método o bien reducir el tamaño de la fórmula analizada, incluso antes de terminar el proceso de etiquetado.

- Eliminar constantes: Concretamente, eliminar las constantes $\top$ y/o $\bot$ que pueden ser introducidas por los procesos *Simplificar*, $\mathcal{S}$, *Reducir* y *Sintetizar*y eliminar la constante $\oslash$ que (como veremos) puede ser introducida por los procesos $\mathcal{S}$, *Reducir* y *Sintetizar*. No obstante, conviene señalar que, al igual que advertimos en el ítem anterior, en aras de la eficiencia, toda constante será eliminada inmediatamente después de ser introducida.

Antes de describir formalmente el proceso *actualizar*, detallaremos las leyes utilizadas para realizar la eliminación de las constantes.

**Teorema 4.9** *En M***3** *se verifican las siguientes equivalencias:*

(1) $\neg\bot \equiv \top \quad \neg\oslash \equiv \oslash \quad \neg\top \equiv \bot$

(2) $J_0\bot \equiv \top \quad J_0\oslash \equiv \bot \quad J_0\top \equiv \bot$

(3) $\neg J_0\bot \equiv \bot \quad \neg J_0\oslash \equiv \top \quad \neg J_0\top \equiv \top$

(4) $J_{\frac{1}{2}}\bot \equiv \bot \quad J_{\frac{1}{2}}\oslash \equiv \top \quad J_{\frac{1}{2}}\top \equiv \bot$

(5) $\neg J_{\frac{1}{2}}\bot \equiv \top \quad \neg J_{\frac{1}{2}}\oslash \equiv \bot \quad \neg J_{\frac{1}{2}}\top \equiv \top$

(6) $J_1\bot \equiv \bot \quad J_1\oslash \equiv \bot \quad J_1\top \equiv \top$

(7) $\neg J_1\bot \equiv \top \quad \neg J_1\oslash \equiv \top \quad \neg J_1\top \equiv \bot$

*(8) Si $A$ y $B$ son fnus y $B \wedge \oslash$ es una subfórmula de $A$, entonces*

$$A \text{ y } A[(B \wedge \oslash)/B] \text{ son equicasisatisfacibles}$$

*(9) Si $A$ y $B$ son fnus y $B \vee \oslash$ es una subfórmula de $A$, entonces*

$$A \text{ y } A[(B \vee \oslash)/\top] \text{ son equicasisatisfacibles}$$

DEMOSTRACIÓN: Los siete primeros apartados son consecuencia inmediata de la definición de la semántica de las constantes y de las conectivas monarias.

(8) Sean $A$ y $B$ fnus tales que $B \wedge \oslash$ es una subfórmula de $A$, entonces

*a*) Si $A = B \wedge \oslash$ el resultado es obvio.

*b*) En otro caso, puesto que disponemos de la ley distributiva de $\wedge$ sobre $\vee$, podemos suponer sin pérdida de generalidad que $A$ es de la forma $\bigvee_{i=1}^{n} D_i$ donde

- cada $D_i$ es de la forma $\bigwedge_{j=1}^{n_i} D_i^j$.
- cada $D_i^j$ es un literal trivaluado, una constante o $B \wedge \oslash$

Entonces, podemos distinguir para cada $D_i$ las dos posibilidades siguientes:

- Todo $D_i^j$ con $1 \leq j \leq n_i$, $D_i^j \neq (B \wedge \oslash)$
- Existe al menos un $D_i^{j_0}$ con $1 \leq j_0 \leq n_i$, tal que $D_i^{j_0} = (B \wedge \oslash)$

Por tanto, el resultado es consecuencia de los siguientes hechos:

- $A$ es casisatisfacible si y sólo si algún $D_i$ es casisatisfacibles.

- Si $D_i = (\bigwedge^{n_i} j = 1 j \neq j_0 D_i^j) \wedge B \wedge \oslash$, entonces $D_i$ y $D_i = (\bigwedge^{n_i} j = 1 j \neq j_0 D_i^j) \wedge B$ son equicasisatisfacibles.

(9) Sean $A$ y $B$ fnus tales que $B \vee \oslash$ es una subfórmula de $A$, entonces

*a*) Si $A = B \vee \oslash$ el resultado es obvio.

*b*) En otro caso, como en el ítem anterior, podemos suponer sin pérdida de generalidad que $A$ es de la forma $\bigvee_{i=i}^{n} D_i$ donde

- cada $D_i$ es de la forma $\bigwedge_{j=1}^{n_i} D_i^j$.
- cada $D_i^j$ es un literal trivaluado, una constante o $B \vee \oslash$

Entonces, podemos distinguir para cada $D_i$ las dos posibilidades siguientes:

- Todo $D_i^j$ con $1 \leq j \leq n_i$, $D_i^j \neq (B \vee \oslash)$
- Existe al menos un $D_i^{j_0}$ con $1 \leq j_0 \leq n_i$, tal que $D_i^{j_0} = (B \vee \oslash)$

Por tanto, el resultado es consecuencia de los siguientes hechos:

- $A$ es casisatisfacible si y sólo si algún $D_i$ es casisatisfacible.
- Si $D_i = (\bigwedge^{n_i} j = 1 j \neq j_0 D_i^j) \wedge (B \vee \oslash)$, entonces $D_i$ y $D_i = \bigwedge^{n_i} j = 1 j \neq j_0 D_i^j$ son equicasisatisfacibles.

*q.e.d.*

Podemos ya dar la descripción completa del proceso *Actualizar*:

**Como ejecuta el proceso *Actualizar***

(I) Sobre la entrada de $\mathcal{F}$ el proceso *Actualizar* ejecuta como sigue: calcula las etiquetas de todos los nodos mediante el proceso recursivo *Etiquetar* y, cada vez que una etiqueta $(\Delta_0(N), \Delta_{\frac{1}{2}}(N), \Delta_1(N))$ es incorporada a un nodo $N$:

- Si $N$ es el nodo raíz entonces examina si es *finalizable*. En caso afirmativo el método termina.
- Para un nodo arbitrario $N$, examina si es *simplificable* y, si lo es, simplifica y, en su caso, elimina las constantes $\bot$ y $\top$ haciendo uso de las leyes $A \vee \top \equiv \top$, $A \wedge \top \equiv A$, $A \vee \bot \equiv A$ y $A \wedge \bot \equiv \bot$

(II) Sobre la salida de los procesos $\mathcal{S}$, *Reduce* y *Sintetiza*, si al ser ejecutados han modificado el árbol sobre el que actúan, *Actualizar* ejecuta como sigue:

(i) Elimina las constantes $\bot$, $\oslash$ y $\top$ haciendo uso del teorema 4.9.

(ii) Reetiqueta cada nodo modificado y sus ascendientes y, cada vez que una etiqueta $(\Delta_0(N), \Delta_{\frac{1}{2}}(N), \Delta_1(N))$ es incorporada a un nodo $N$:

- Si $N$ es el nodo raíz entonces examina si es *finalizable*. En caso afirmativo el método termina.
- Para un nodo arbitrario $N$, examina si es *simplificable* y, si lo es, simplifica y, en su caso, elimina las constantes $\bot$ y $\top$ haciendo uso de las leyes
  $A \vee \top \equiv \top, \quad A \wedge \top \equiv A, \quad A \vee \bot \equiv A \quad$ y $\quad A \wedge \bot \equiv \bot$

### 4.9.1. La salida de Actualizar

Si la ejecución de *Actualizar* se realiza sobre un árbol $T_B$ *finalizable*, entonces TAS-M**3** termina con una de las siguientes salidas:

(i) VÁLIDA, si $T_B$ es $\bot$.

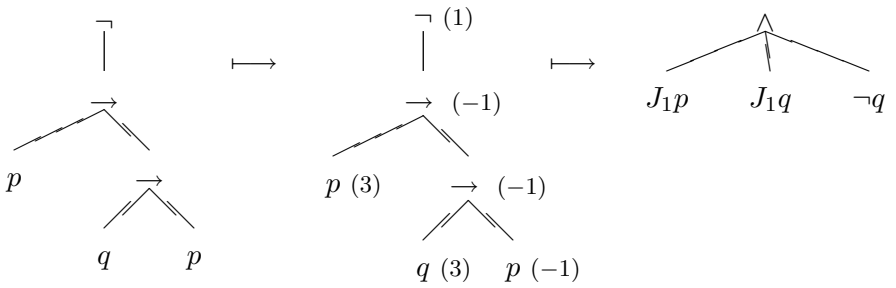
(ii) NO VÁLIDA y un contramodelo (descrito más adelante en la sección 4.14), en otro caso.

- Si $T_B$ es $\top$ o $\oslash$, el contramodelo generado no necesita información adicional a la guardada hasta este instante de la ejecución de TAS-M**3**.
- Si $\Delta_1(B) \neq \varnothing$ entonces, para generar el contramodelo de la fórmula $A$ analizada necesitamos almacenar el primer elemento de $\Delta_1(B)$.
- Si $\Delta_{\frac{1}{2}}(B) \neq \varnothing$ entonces, para generar el contramodelo de la fórmula $A$ analizada necesitamos almacenar el primer elemento de $\Delta_{\frac{1}{2}}(B)$.

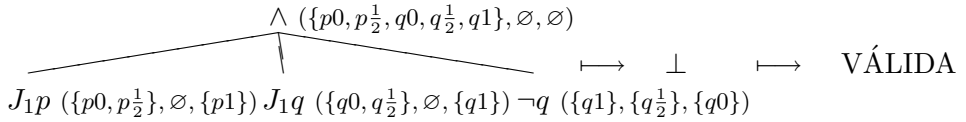
El teorema 4.8 asegura que todas las transformaciones realizadas al *Simplificar* un árbol proporcionan una fórmula lógicamente equivalente a la fórmula a la que se aplican. Por su parte, los Teoremas 4.9 y 4.7 aseguran, respectivamente, que las acciones realizadas tras la eliminación de las constantes y tras el test de finalizabilidad proporcionan una fórmula que es casisatisfacible si y sólo si lo es la fórmula sobre la que actúan. Por tanto, tenemos el siguiente resultado.

**Teorema 4.10** *Sean $A$ y $B$ dos fnus, si Actualizar $(T_A) = T_B$ entonces $A$ y $B$ son equicasisatisfacibles.*

**Ejemplo 4.7** Dada la fórmula $A = p \rightarrow (q \rightarrow p)$. La entrada a TAS-M**3** es el árbol sintáctico de $\neg A$. Tras *Signar*, se obtiene el último árbol de la secuencia mostrada en la siguiente figura:

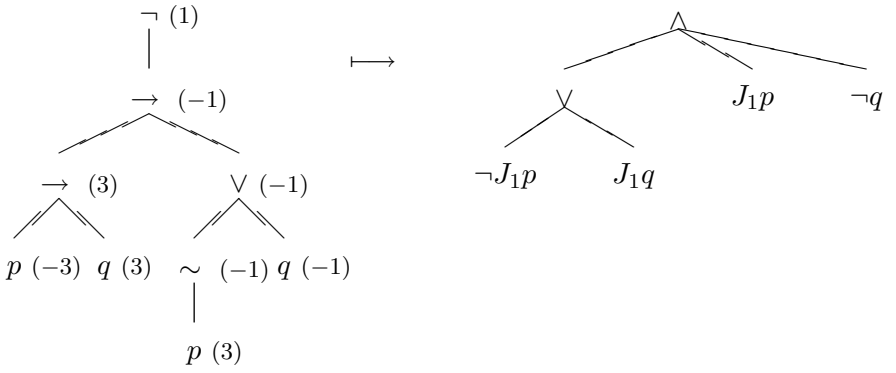



Este árbol es la entrada a la transformación $\mathcal{F}$(ya que al ser su raíz $\wedge$ la tarea generada es todo el árbol), concretamente la entrada al proceso *Actualizar* y, puesto que es la primera vez que interviene, etiqueta todos los nodos y obtenemos el árbol $T_B$ siguiente:

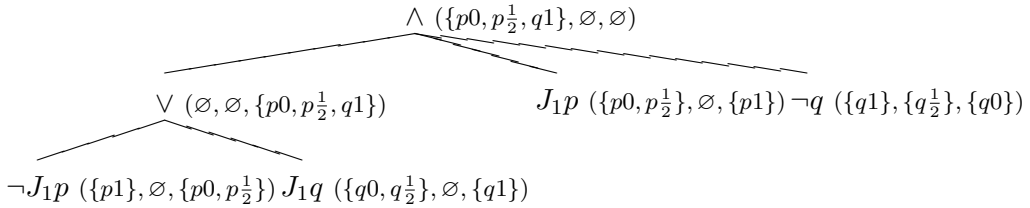



El árbol $T_B$ obtenido es *0-concluyente* ya que $Q\mathbf{3} \subseteq \Delta_0(T_B)$, y por tanto *Simplificar* sustituye el árbol por $\bot$, con lo que es *finalizable* y TAS-M**3** termina con salida "VÁLIDA".

**Ejemplo 4.8** Consideramos la fórmula $A = (p \to q) \to (\sim p \vee q)$, el árbol de la negación fuerte de $A$, $T_{\neg A}$, es la entrada del proceso *Signar* que proporciona el segundo árbol de la figura siguiente:




Se inicia ahora la transformación de árboles $\mathcal{F}$, sobre esta única tarea generada, con el proceso *Actualizar* que *etiqueta* el árbol de la siguiente forma:




Este árbol $T_B$ es *0-concluyente* puesto que la cláusula trivaluada restringida $\neg J_1 p \vee J_1 q$ es tal que

$$\Delta_1(\neg J_1 p \vee J_1 q) = \{p0, p\frac{1}{2}, q1\}, \quad \overline{\Delta_1(\neg J_1 p \vee J_1 q)} = \{p1, q0, q\frac{1}{2}\}$$

y

$$(P_{\mathbf{3}} \cup Q_{\mathbf{3}}) - \overline{\Delta_1(\neg J_1 p \vee J_1 q)} = \{p0, p\frac{1}{2}, q1\} \subseteq \Delta_0(B)$$

Por lo tanto *Simplificar* sustituye el árbol por $\perp$ que es *finalizable* y TAS-M**3** termina con salida "VÁLIDA".

## 4.10. El proceso $\mathcal{S}$

El proceso que describimos en esta sección, denotado por $\mathcal{S}$, si bien no hace uso de la información en los conjuntos $\Delta$, adquiere su importancia precisamente por su interacción con los procesos que hacen uso de tal información, es decir, su interacción con los procesos *Actualizar*, *Reducir* y *Sintetizar*.

El proceso $\mathcal{S}$ tiene como objetivo, una vez más, disminuir el tamaño de la fórmula antes de distribuir. El correspondiente proceso en el caso proposicional clásico es una generalización, para formas normales negativas, de la regla del literal puro para el análisis de la satisfacibilidad de formas clausales. Es decir, el proceso $\mathcal{S}$ en el caso proposicional clásico (como puede verse en el Apéndice A) consiste en la eliminación, en una forma normal negativa $A$, de los símbolos proposicionales $p$ que ocurren sólo positivamente en $A$ ($\neg p$ no ocurre como hoja de $T_A$) y de los símbolos proposicionales $p$ que ocurren sólo negativamente en $A$ ($p$ no ocurre como hoja de $T_A$).

El interés de este proceso en el marco de los métodos TAS, se debe a que su intervención tras cada aplicación de *Actualizar* asegura que, no sólo elimina símbolos positivos y símbolos negativos en la fórmula analizada sino en cada una de las fórmulas obtenidas a lo largo de la ejecución del método. Por otra parte, podríamos decir que $\mathcal{S}$ es el complemento adecuado al modo de reducción de tamaño aportado por los conjuntos $\Delta$: la existencia de muchas ocurrencias de literales con el mismo sufijo aumenta la probabilidad de aplicar *reducciones* y

si existen pocas (o han llegado a ser pocas por reducciones previas) aumenta la probabilidad de actuación de $\mathcal{S}$. Abordamos ya la descripción de $\mathcal{S}$ para M**3**.

Recordemos que nuestro objetivo es detectar, para algún símbolo proposicional $p$, subconjuntos $\Gamma$ de

$$\{\epsilon, p, \neg p, J_0 p, J_{\frac{1}{2}} p, J_1 p, \neg J_0 p, \neg J_{\frac{1}{2}} p, \neg J_1 p\}$$

tales que la conjunción de dos elementos cualesquiera de $\Gamma$ sea casisatisfacible y, consecuentemente, no culpables de la no casisatisfacibilidad de la fórmula analizada.

Veamos que, si en el caso proposicional clásico, este objetivo conducía a dos tipos de literales (positivos y negativos), para el caso trivaluado que nos ocupa la total consecución de nuestro objetivo nos lleva a considerar tres grupos de literales (uno por cada valor de verdad).

Por otra parte, destaquemos que estos tres conjuntos no son disjuntos, es decir, hay algunos literales que están en más de un grupo. Esto implica a una mayor aplicabilidad de $\mathcal{S}$ y por lo tanto a una mayor eficiencia.

La siguiente definición nos muestra cuáles son estos subconjuntos de $\Gamma$.

**Definición 4.15**

1. Un literal trivaluado se dice que es de *tipo 0* si su prefijo pertenece al conjunto de literales trivaluados $\{\neg, J_0, \neg J_{\frac{1}{2}}, \neg J_1\}$.

2. Un literal trivaluado se dice que es de *tipo* $\frac{1}{2}$ si su prefijo pertenece al conjunto de literales trivaluados $\{\epsilon, \neg, \neg J_0, J_{\frac{1}{2}}, \neg J_1\}$.

3. Un literal trivaluado se dice que es de *tipo 1* si su prefijo pertenece al conjunto de literales trivaluados $\{\epsilon, \neg J_0, \neg J_{\frac{1}{2}}, J_1\}$.

**Definición 4.16** Dada una fnu $A$, se dice que un símbolo proposicional $p$ es de *tipo* $b$ (con $b \in \mathbf{3}$) para $A$ si todas las ocurrencias de literales trivaluados en $A$ con sufijo $p$ son de tipo $b$.

El siguiente teorema determina cuándo y qué sustituciones podemos realizar manteniendo la casisatisfacibilidad.

**Teorema 4.11** *Dada una fnu $A$ y un símbolo proposicional $p$ de tipo $b$ en $A$ se tiene:*

1. *Si $p$ es de tipo 0 en $A$ entonces $A$ y $A[p/\bot]$ son equicasisatisfacibles.*

2. *Si $p$ es de tipo $\frac{1}{2}$ en $A$ entonces $A$ y $A[p/\oslash]$ son equicasisatisfacibles.*

*3. Si $p$ es de tipo 1 en $A$ entonces $A$ y $A[p/\top]$ son equicasisatisfacibles.*

DEMOSTRACIÓN: Lo demostramos para el primer caso, el resto de los casos se demuestran de forma similar.

Por el Lema 4.6 podemos suponer, sin pérdida de generalidad, que $A$ es de la forma $A = \bigvee_{i=1}^{n} D_i$ donde, para todo $i$ con $1 \leq i \leq n$, $D_i = \bigwedge_{j=1}^{n_i} l_i^j$ es un cubo.

Supongamos en primer lugar que $A[p/\bot]$ es casisatisfacible y sea $I$ una interpretación tal que $I(A[p/\bot]) \neq 0$. Entonces, obviamente, toda extensión $I'$ de $I$ tal que $I'(p) = 0$ satisface $I(A) \neq 0$ y por lo tanto $A$ es casisatisfacible.

Recíprocamente, si $A$ es casisatisfacible entonces, alguno de los cubos $D_{i_0}$ con $1 \leq i_0 \leq n$ es casisatisfacible. Ahora, podemos distinguir dos casos:

- Ningún literal trivaluado $l_i^j$ con sufijo $p$ interviene en $D_{i_0}$. En cuyo caso, $D_{i_0}[p/\bot] = D_{i_0}$ y, puesto que

$$A[p/\bot] = \bigvee_{i=1}^{n} D_i[p/\bot]$$

  se tiene que $A[p/\bot]$ es casisatisfacible.

- En otro caso, si en $D_{i_0}$ intervienen literales trivaluados con sufijo $p$, por asociatividad y conmutatividad se tiene que $D_{i_0} \equiv D'_{i_0} \wedge {D'_{i_0}}'$ donde se han agrupado todos los literales que tienen como sufijo a $p$ en el cubo $D'_{i_0}$ y los literales que no tienen como sufijo a $p$ en el cubo ${D'_{i_0}}'$. Puesto que $D_{i_0}$ es casisatisfacible, lo es ${D'_{i_0}}'$. Por lo tanto, existe una interpretación $I$ tal que $I({D'_{i_0}}') \neq 0$. Consideremos cualquier extensión $I'$ de $I$ tal que $I'(p) = 0$. Puesto que $p$ no ocurre en ${D'_{i_0}}'$, se tiene que $I'({D'_{i_0}}') = I({D'_{i_0}}') \neq 0$. Por otra parte, por hipótesis, en $D'_{i_0}$ sólo ocurren literales de tipo 0, es decir, todo literal trivaluado $l$ en $D'_{i_0}$ es tal que $l \in \{\neg p, J_0 p, \neg J_{\frac{1}{2}} p, \neg J_1 p\}$ y para todos ellos se tiene que $I'(l) \neq 0$ y, consecuentemente, $I'(D'_{i_0}) \neq 0$. Por lo tanto, $I'(D_{i_0}[p/\bot]) = I'(D'_{i_0}) \neq 0$. Así pues

$$I'(D_{i_0}[p/\bot]) = I'(D'_{i_0}[p/\bot] \wedge {D'_{i_0}}'[p/\bot]) \neq 0$$

  y $A[p/\bot]$ es casisatisfacible.

*q.e.d.*

Tenemos ya las herramientas necesarias para establecer el resultado que nos aporte un modo eficiente de utilizar en nuestro método la información del teorema anterior. Este resultado se recoge en el Teorema 4.12, para cuyo enunciado

necesitamos introducir la siguiente notación, que será ampliamente utilizada posteriormente en la descripción de los procesos *Reducir* y *Sintetizar*:

**Notación:** Además de la notación $A[B/C]$, ya utilizada, que indica que en $A$ se han sustituido todas las ocurrencias de la subfórmula $B$ de $A$ por la fórmula $C$, usaremos la siguiente:

- $A[\![l/C]\!]$ indica que en $A$ se han sustituido todas las ocurrencias del literal trivaluado $l$ en $A$ por la fórmula $C$.

**Teorema 4.12** *Sea $A$ una fnu, $p$ un símbolo proposicional $p$ de tipo $b$ en $A$ y $l$ un literal con sufijo $p$. Entonces,*

$$A \text{ y } A[\![l/\top]\!] \text{ son equicasisatisfacibles.}$$

Demostración:

Por el teorema anterior, $A$ es casisatisfacible si y sólo si lo es $A[p/\odot]$ donde $\odot$ es $\bot$ si $b = 0$, $\oslash$ si $b = \frac{1}{2}$ y $\top$ si $b = 1$. Por otra parte, el Teorema 4.9 asegura que para todo $b \in \mathbf{3}$, si $p$ es un símbolo proposicional de tipo $b$ se tiene que $\alpha p \equiv \top$ o bien $\alpha p \equiv \oslash$. Finalmente, de nuevo por el Teorema 4.9, se tiene que

- Si $A$ y $B$ son fnus y $B \wedge \oslash$ es una subfórmula de $A$, entonces

$$A \text{ y } A[(B \wedge \oslash)/B] \text{ son equicasisatisfacibles}$$

- Si $A$ y $B$ son fnus y $B \vee \oslash$ es una subfórmula de $A$, entonces

$$A \text{ y } A[(B \vee \oslash)/\top] \text{ son equicasisatisfacibles}$$

Por lo tanto, $A$ y $A[\![l/\top]\!]$ son equicasisatisfacibles. *q.e.d.*

Los dos teoremas anteriores son la justificación teórica del proceso $\mathcal{S}$ que se define formalmente a continuación:

**Definición 4.17** Sea $A$ una fnu, y $T_A$ la entrada del proceso $\mathcal{S}$. Entonces mientras exista un símbolo proposicional $p$ de tipo $b$ para $A$ para algún $b \in \mathbf{3}$, $\mathcal{S}$ realiza en $T_A$ las sustituciones de todos los literales con sufijo $p$ por $\top$

### 4.10.1. Información para la generación de modelos

Si $\mathcal{S}$ actúa, entonces, para generar el contramodelo de la fórmula $A$ analizada necesitamos almacenar los elementos de la forma $pb$ en los que $p$ es de tipo $b$ que han sido eliminados en la ejecución de $\mathcal{S}$.

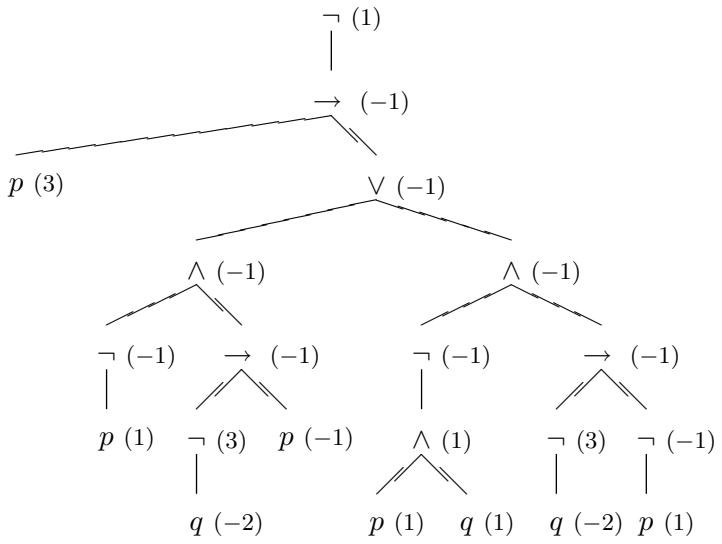
**Teorema 4.13** *Si $\mathcal{S}(T_A) = T_B$ entonces $A$ y $B$ son equicasisatisfacibles.*

DEMOSTRACIÓN: Es consecuencia inmediata del Teorema 4.12. *q.e.d.*

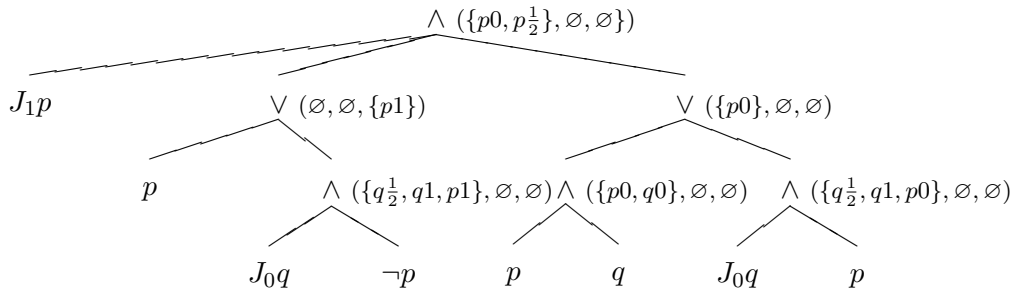
**Ejemplo 4.9** Consideremos la fórmula

$$A = p \rightarrow ((\neg p \wedge (\neg q \rightarrow p)) \vee (\neg(p \wedge q) \wedge (\neg q \rightarrow \neg p)))$$

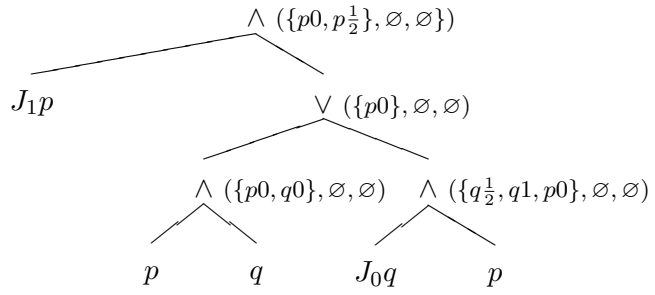
La traza de *Signar* y *Etiquetar* sobre $T_{\neg A}$ se muestra a continuación:



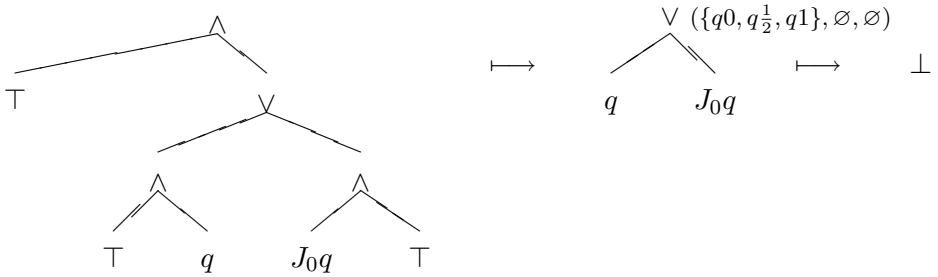

Tras *Signar*, se genera una única tarea que corresponde al siguiente árbol, que será la entrada del proceso $\mathcal{F}$:



Este árbol $T_B$ es podable, ya que el subárbol $T_C$ con $C = p \vee (J_0 q \wedge \neg p)$ verifica que $\Delta_0(C) = \varnothing$ y $P_{\mathbf{3}} - \Delta_1(C) = P_{\mathbf{3}} - \{p1\} = \{p0, p\frac{1}{2}\} \subseteq \Delta_0(B) = \{p0, p\frac{1}{2}\}$, por lo tanto *Simplificar* elimina (*poda*) el subárbol $T_C$, y queda el árbol:




En este árbol $T_D$ el símbolo $p$ es de tipo 1. Por lo tanto el proceso $\mathcal{S}$ sustituye todos los literales que lo tienen como sufijo por $\top$, y *Actualizar* elimina $\top$ tal como se muestra a continuación:




El último paso en la traza ha obtenido $\bot$ puesto que el árbol de la fórmula $q \wedge J_0 q$ es *0-concluyente* ya que $Q_{\mathbf{3}} \subseteq \Delta_0(E) = \{q0, q\frac{1}{2}, q1\}$. Por lo tanto, la salida de TAS-M**3** es "VÁLIDA".

Obsérvese que $\mathcal{S}$ no era aplicable a la fórmula inicial, pero sí lo es después de haber ejecutado *Actualizar*.

## 4.11. Los procesos *Reducir* y *Sintetizar*

El objetivo de estos dos procesos es dar condiciones más generales que permitan usar la información en los conjuntos $\Delta$ que no ha podido ser usada por el proceso *Actualizar*. El proceso *Simplificar* utiliza la información de los conjuntos $\Delta$ en *sentido fuerte*, es decir, para realizar sustituciones globales de una fórmula por una constante o un literal.

Ahora bien, como era de esperar, esta sustitución global no siempre es posible y nuestra siguiente opción será intentar utilizar la información en los conjuntos $\Delta$ en *sentido débil*. Específicamente, el nuevo uso de esta información permitirá disminuir el tamaño de la fórmula dejando *a lo sumo una* ocurrencia de cada símbolo proposicional $p$ que interviene en la etiqueta $(\Delta_0, \Delta_{\frac{1}{2}}, \Delta_1)$. Esta es la misión del proceso *Reducir*.

En particular, en los casos de cubos y cláusulas el proceso *Reducir* permitirá obtener su forma restringida sin ningún proceso adicional.

Por su parte, el proceso *Sintetizar* es, como veremos, una "reducción parcial".

### 4.11.1. El proceso *Reducir*

La descripción de este proceso requiere extender los Teoremas 4.4, 4.5 y 4.6.

Comenzamos extendiendo el Teorema 4.4, ampliando la información contenida en el conjunto $\Delta_0$.

**Teorema 4.14** *Sea $A$ una fnu y $p \in Q$ entonces:*

| | | | |
|---|---|---|---|
| (1) | $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$ | *si y sólo si* | $A \equiv J_1 p \wedge A[p/\top]$ |
| (2) | $\{p0, p1\} \subseteq \Delta_0(A)$ | *si y sólo si* | $A \equiv J_{\frac{1}{2}} p \wedge A[p/\oslash]$ |
| (3) | $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$ | *si y sólo si* | $A \equiv J_0 p \wedge A[p/\bot]$ |
| (4) | *Si* $\{p0\} = \Delta_0(A) \cap P_{\mathbf{3}}$ | *entonces* | $A \equiv \neg J_0 p \wedge A[\![\neg J_0 p/\top, J_0 p/\bot]\!]$ |
| (5) | *Si* $\{p\frac{1}{2}\} = \Delta_0(A) \cap P_{\mathbf{3}}$ | *entonces* | $A \equiv \neg J_{\frac{1}{2}} p \wedge A[\![\neg J_{\frac{1}{2}} p/\top, J_{\frac{1}{2}} p/\bot]\!]$ |
| (6) | *Si* $\{p1\} = \Delta_0(A) \cap P_{\mathbf{3}}$ | *entonces* | $A \equiv \neg J_1 p \wedge A[\![\neg J_1 p/\top, J_1 p/\bot]\!]$ |

DEMOSTRACIÓN: En los tres primeros apartados , que las equivalencias afirmadas a la derecha aseguran las afirmaciones sobre los conjuntos $\Delta$ establecidas a la izquierda, es consecuencia inmediata de la definición de los conjuntos $\Delta$ para los literales trivaluados y del lema 4.3.

Demostramos ahora, para cada item, que las afirmaciones sobre los conjuntos $\Delta$ establecidas a la izquierda aseguran las equivalencias afirmadas a la derecha.

1. Supongamos que $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$.

   Sea $I$ una interpretación arbitraria. Podemos distinguir dos casos:

   - Si $I(p) \in \{0, \frac{1}{2}\}$, por el lema 4.3, se tiene que $I(A) = 0$. Por lo tanto, $I(A) = I(J_1 p) = 0 = I(A[p/\top] \wedge J_1 p)$.
   - Si $I(p) = 1$ se tiene que $I(J_1 p) = 1$ y el comportamiento de $p$ en $A$ respecto a $I$ es el mismo que el de $\top$. Por lo tanto, $I(A) = I(A[p/\top]) = I(A[p/\top] \wedge J_1 p)$.

2. Supongamos que $\{p0, p1\} \subseteq \Delta_0(A)$.

   Sea $I$ una interpretación arbitraria. Podemos distinguir dos casos:

   - Si $I(p) \in \{0, 1\}$, por el lema 4.3, se tiene que $I(A) = 0$. Por lo tanto, $I(A) = I(J_{\frac{1}{2}}p) = 0 = I(A[p/\oslash] \wedge J_{\frac{1}{2}}p)$.
   - Si $I(p) = \frac{1}{2}$ se tiene que $I(J_{\frac{1}{2}}p) = 1$ y el comportamiento de $p$ en $A$ respecto a $I$ es el mismo que el de $\oslash$. Por lo tanto, $I(A) = I(A[p/\oslash]) = I(A[p/\oslash] \wedge J_{\frac{1}{2}}p)$.

3. Supongamos que $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$.

   Sea $I$ una interpretación arbitraria. Podemos distinguir dos casos:

   - Si $I(p) \in \{\frac{1}{2}, 1\}$, por el lema 4.3, se tiene que $I(A) = 0$. Por lo tanto, $I(A) = I(J_0p) = 0 = I(A[p/\bot] \wedge J_0p)$.
   - Si $I(p) = 0$ se tiene que $I(J_0p) = 1$ y el comportamiento de $p$ en $A$ respecto a $I$ es el mismo que el de $\bot$. Por lo tanto, $I(A) = I(A[p/\bot]) = I(A[p/\bot] \wedge J_0p)$.

4. Supongamos que $\{p0\} = \Delta_0(A) \cap P_{\mathbf{3}}$, entonces por el teorema 4.4 se tiene que $A \equiv A \wedge \neg J_0p$.

   Sea $I$ una interpretación arbitraria. Podemos distinguir dos casos:

   - si $I(p) = 0$, por el lema 4.3, se tiene que $I(A) = 0$. Por lo tanto $I(A) = I(\neg J_0p) = 0 = I(\neg J_0p \wedge A[\![\neg J_0p/\top, J_0p/\bot]\!])$.
   - si $I(p) \in \{\frac{1}{2}, 1\}$se tiene que $I(\neg J_0p) = 1$, es decir, el comportamiento de $\neg J_0p$ en $A$ respecto a $I$ es el mismo que el de $\top$ y el comportamiento de $J_0p$ en $A$ respecto a $I$ es el mismo que el de $\bot$. Por lo tanto $I(A) = I(A[\![\neg J_0p/\top, J_0p/\bot]\!]) = I(\neg J_0p \wedge A[\![\neg J_0p/\top, J_0p/\bot]\!])$.

5. Supongamos que $\{p\frac{1}{2}\} = \Delta_0(A) \cap P_{\mathbf{3}}$, entonces por el teorema 4.4 se tiene que $A \equiv A \wedge \neg J_{\frac{1}{2}}p$.

   Sea $I$ una interpretación arbitraria. Podemos distinguir dos casos:

   - si $I(p) = \frac{1}{2}$, por el lema 4.3, se tiene que $I(A) = 0$. Por lo tanto $I(A) = I(\neg J_{\frac{1}{2}}p) = 0 = I(\neg J_{\frac{1}{2}}p \wedge A[\![\neg J_{\frac{1}{2}}p/\top, J_{\frac{1}{2}}p/\bot]\!])$.
   - si $I(p) \in \{0, 1\}$ se tiene que $I(\neg J_{\frac{1}{2}}p) = 1$, es decir, el comportamiento de $\neg J_{\frac{1}{2}}p$ en $A$ respecto a $I$ es el mismo que el de $\top$ y el comportamiento de $J_{\frac{1}{2}}p$ en $A$ respecto a $I$ es el mismo que el de $\bot$. Por lo tanto $I(A) = I(A[\![\neg J_{\frac{1}{2}}p/\top, J_{\frac{1}{2}}p/\bot]\!]) = I(\neg J_{\frac{1}{2}}p \wedge A[\![\neg J_{\frac{1}{2}}p/\top, J_{\frac{1}{2}}p/\bot]\!])$.

6. Supongamos que $\{p1\} = \Delta_0(A) \cap P_{\mathbf{3}}$, entonces por el teorema 4.4 se tiene que $A \equiv A \wedge \neg J_1 p$.

   Sea $I$ una interpretación arbitraria. Podemos distinguir dos casos:

   - si $I(p) = 1$, por el lema 4.3, se tiene que $I(A) = 0$. Por lo tanto $I(A) = I(\neg J_1 p) = 0 = I(\neg J_1 p \wedge A[\![\neg J_1 p/\top, J_1 p/\bot]\!])$.
   - si $I(p) \in \{0, \frac{1}{2}\}$ se tiene que $I(\neg J_1 p) = 1$, es decir, el comportamiento de $\neg J_1 p$ en $A$ respecto a $I$ es el mismo que el de $\top$ y el comportamiento de $J_1 p$ en $A$ respecto a $I$ es el mismo que el de $\bot$. Por lo tanto $I(A) = I(A[\![\neg J_1 p/\top, J_1 p/\bot]\!]) = I(\neg J_1 p \wedge A[\![\neg J_1 p/\top, J_1 p/\bot]\!])$.

*q.e.d.*

Dualmente se obtiene el teorema 4.15 que extiende el teorema 4.5 y amplia la información contenida en el conjunto $\Delta_1$:

**Teorema 4.15** *Sea $A$ una fnu y $p \in Q$ entonces:*

| | | | |
|---|---|---|---|
| (1) | $\{p0, p\frac{1}{2}\} \subseteq \Delta_1(A)$ | *si y sólo si* | $A \equiv \neg J_1 p \vee A[p/\top]$ |
| (2) | $\{p0, p1\} \subseteq \Delta_1(A)$ | *si y sólo si* | $A \equiv \neg J_{\frac{1}{2}} p \vee A[p/\oslash]$ |
| (3) | $\{p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$ | *si y sólo si* | $A \equiv \neg J_0 p \vee A[p/\bot]$ |
| (4) | *Si* $\{p0\} = \Delta_1(A) \cap P_{\mathbf{3}}$ | *entonces* | $A \equiv J_0 p \vee A[\![J_0 p/\bot, \neg J_0 p/\top]\!]$ |
| (5) | *Si* $\{p\frac{1}{2}\} = \Delta_1(A) \cap P_{\mathbf{3}}$ | *entonces* | $A \equiv J_{\frac{1}{2}} p \vee A[\![J_{\frac{1}{2}} p/\bot, \neg J_{\frac{1}{2}} p/\top]\!]$ |
| (6) | *Si* $\{p1\} = \Delta_1(A) \cap P_{\mathbf{3}}$ | *entonces* | $A \equiv J_1 p \vee A[\![J_1 p/\bot, \neg J_1 p/\top]\!]$ |

Para analizar la utilidad de los teoremas anteriores, recordemos que, como indicamos en la sección 4.3, TAS-M**3** utiliza (en caso de no evitar todas las distribuciones) la distribución de $\wedge$ sobre $\vee$ y en consecuencia, el resultado:

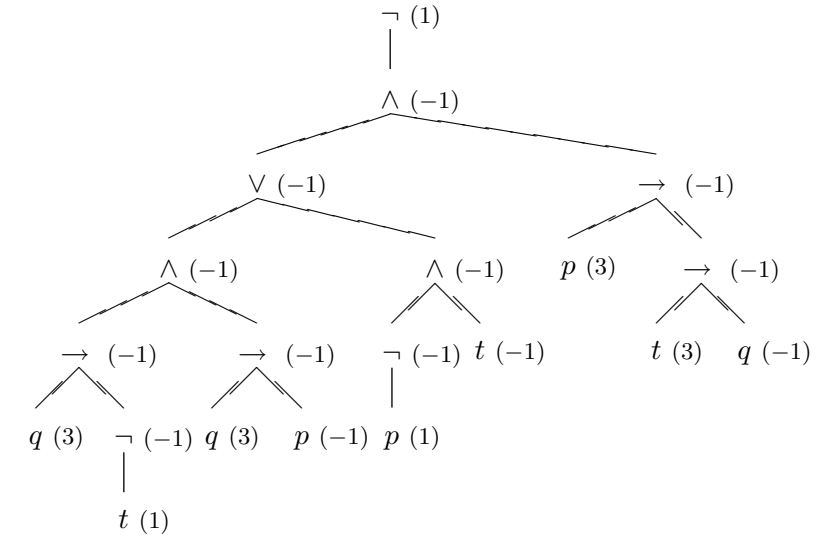
> Si $A = \bigvee_{i \in I} A_i$ entonces $A$ es casisatisfacible si y sólo si existe $i_0 \in I$ tal que $A_{i_0}$ es casisatisfacible.

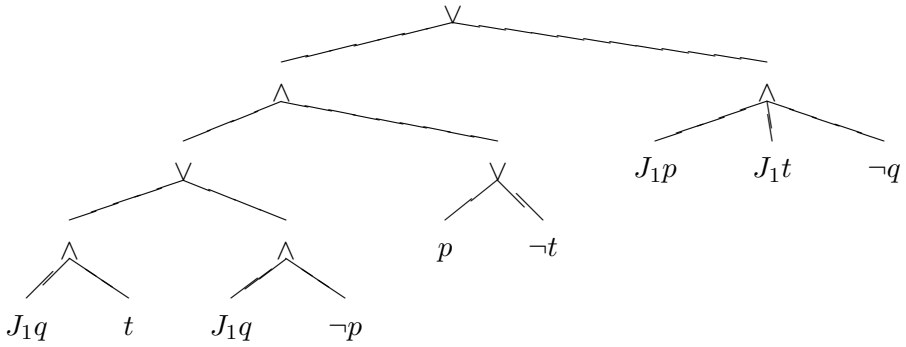
Por lo tanto, la utilidad del teorema 4.15 es clara, ya que nos permite realizar sustituciones que, además de disminuir el tamaño de la fórmula, suben una conectiva $\vee$. Sin embargo, el teorema 4.14 disminuye el tamaño de la fórmula pero es posible que, al realizar las sustituciones permitidas por el teorema en una subfórmula, suba una conectiva $\wedge$ motivando así una posible distribución de $\wedge$ sobre $\vee$ en el futuro. Este último caso es obviamente indeseable como mostramos a continuación.

**Ejemplo 4.10** Veamos que aplicar los resultados del teorema 4.14 no siempre es conveniente.

Consideremos la fórmula $A = \Big(((q \to \neg t) \wedge (q \to p)) \vee (\neg p \wedge t)\Big) \wedge \big(p \to (t \to q)\big)$.

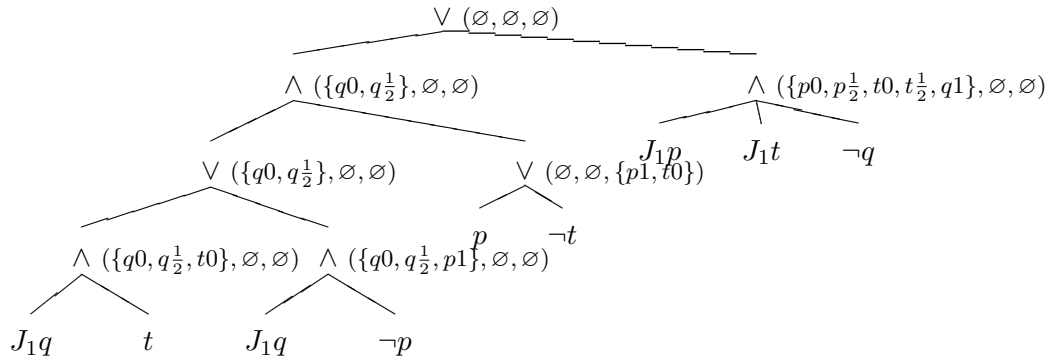
La traza del proceso *Signar* sobre $T_{\neg A}$ es:




La transformación $\mathcal{F}$, empieza *etiquetando* el único árbol generado[1] se tiene entonces el árbol:

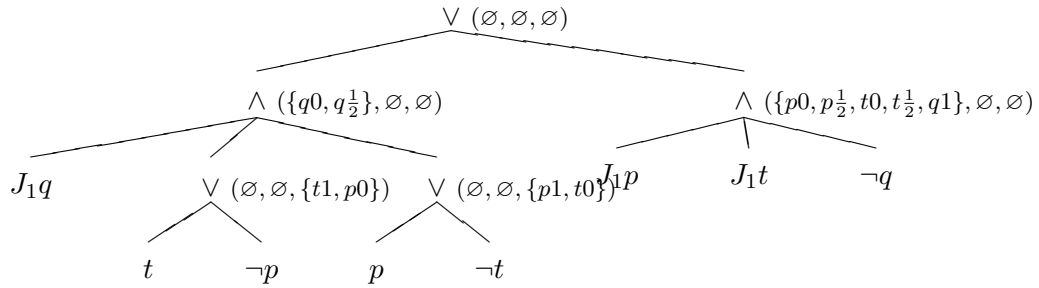
[1]Las etiquetas de los nodos hoja no se escriben para hacer más legible el árbol resultante




Este árbol no es *simplificable*, por lo que será la salida del proceso *Actualizar* y, puesto que $\mathcal{S}$ no puede actuar, será la entrada del proceso *Reducir*, detectando que para la subfórmula

$$B = ((J_1 q \wedge t) \vee (J_1 q \wedge \neg p)) \wedge (p \vee \neg t)$$

se tiene que $\Delta_0(B) = \{q0, q\frac{1}{2}\} \neq \varnothing$. Si no se pusiesen restricciones a las sustituciones indicadas por el teorema 4.14, se realizaría el cambio de $B$ por $J_1 q \wedge B[q/\top]$, y quedaría el árbol:




Este árbol es de menor tamaño y con menos niveles que el anterior, sin embargo, ha ocurrido algo indeseable, ha subido una $\wedge$ por encima de un $\vee$, lo que en un futuro nos llevaría a realizar una distribución de $\wedge$ sobre $\vee$. La filosofía del método es que se eviten tantas distribuciones como sea posible y en este caso, sin embargo, habríamos incrementado su número. Por tanto es necesario restringir el uso del teorema 4.14.

El análisis anterior nos muestra que, como en el método TAS para la lógica clásica ( (Ojeda, 1996) y (Aguilera y de Guzmán, 1993)), debemos restringir en algunos casos el uso del teorema 4.14. Para ello, introducimos el siguiente conjunto:

**Definición 4.18** Sea $A$ una fnu. Si la raíz de $T_A$ es $\wedge$, definimos

1. $\Delta_0^l(A) = \{\alpha p \mid \alpha \in \Lambda, p \in Q$ y $\alpha p$ es un hijo de la raíz de $T_A\}$.
2. $\Delta_0^*(A) = \{pb \mid pb \in \Delta_0(l)$ con $l \in \Delta_0^l(A)\}$

Si la raíz de $T_A$ no es $\wedge$, definimos $\Delta_0^l(A) = \varnothing$ y, por tanto, $\Delta_0^*(A) = \varnothing$.

Tenemos ahora el siguiente resultado, consecuencia inmediata del teorema 4.14:

**Teorema 4.16** *Sea* $A = \bigwedge_{i=1}^n A_i$ *una fnu y* $p \in Q$ *entonces:*

(1) *Si* $\{p0, p\frac{1}{2}\} \subseteq \Delta_0^*(A)$ *entonces* $A \equiv J_1 p \wedge A[p/\top]$

(2) *Si* $\{p0, p1\} \subseteq \Delta_0^*(A)$ *entonces* $A \equiv J_{\frac{1}{2}} p \wedge A[p/\oslash]$

(3) *Si* $\{p\frac{1}{2}, p1\} \subseteq \Delta_0^*(A)$ *entonces* $A \equiv J_0 p \wedge A[p/\bot]$

*En otro caso, es decir, si existe pb tal que* $\{pb\} = P_{\mathbf{3}} \cap \Delta_0^*(A)$*, entonces*

(4) *Si* $p \in \Delta_0^l(A)$ *entonces* $A \equiv p \wedge A[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!]$

(5) *Si* $\neg p \in \Delta_0^l(A)$ *entonces* $A \equiv \neg p \wedge A[\![\neg p/\top, \neg J_1 p/\top, J_1 p/\bot]\!]$

(6) *Si* $\neg J_0 p \in \Delta_0^l(A)$ *entonces* $A \equiv \neg J_0 p \wedge A[\![\neg J_0 p/\top, J_0 p/\bot]\!]$

(7) *Si* $\neg J_{\frac{1}{2}} p \in \Delta_0^l(A)$ *entonces* $A \equiv \neg J_{\frac{1}{2}} p \wedge A[\![\neg J_{\frac{1}{2}} p/\top, J_{\frac{1}{2}} p/\bot]\!]$

(8) *Si* $\neg J_1 p \in \Delta_0^l(A)$ *entonces* $A \equiv \neg J_1 p \wedge A[\![\neg J_1 p/\top, J_1 p/\bot]\!]$

Demostración:

1. Los tres primeros apartados son casos particulares de los tres primeros apartados del Teorema 4.14

4 Supongamos que $p \in \Delta_0^l(A)$. Entonces, por las propiedades conmutativa y asociativa de $\wedge$ se tiene que $A \equiv p \wedge B$.

Por el Lema 4.6, podemos suponer sin pérdida de generalidad que la fnu $B$ está en forma normal disyuntiva, es decir, que $B = \bigvee_{i \in I} D_i$ donde cada $D_i$ es un cubo trivaluado.

Tenemos pues que $A \equiv p \wedge B \equiv p \wedge p \wedge \bigvee_{i \in I} D_i \equiv \bigvee_{i \in I} (p \wedge D_i)$.

Ahora, para cada $i \in I$, podemos distinguir dos casos

- alguno de los literales trivaluados en $D_i$ tiene como prefijo $\epsilon$, $J_0$ ó $\neg J_0$. En este caso, teniendo en cuenta las equivalencias

$$p \wedge p \equiv p; \qquad p \wedge J_0 p \equiv \bot; \qquad p \wedge \neg J_0 p \equiv p$$

se tiene que $p \wedge D_i \equiv p \wedge D_i[\![p/\top, J_0 p/\bot, \neg J_0 p/\top]\!]$.

- ninguno de los literales trivaluados en $D_i$ tiene como prefijo $\epsilon$, ni $J_0$ ni $\neg J_0$. En este caso,se tiene que $D_i[\![p/\top, J_0p/\bot, \neg J_0p/\top]\!] = D_i$ y, en consecuencia, $p \wedge D_i \equiv p \wedge D_i[\![p/\top, J_0p/\bot, \neg J_0p/\top]\!]$.

Por tanto, $A \equiv p \wedge B \equiv p \wedge p \wedge B \equiv p \wedge A[\![p/\top, J_0p/\bot, \neg J_0p/\top]\!]$.

5 Si $\neg p \in \Delta_0^l(A)$, la demostración es la misma paso a paso al caso ($p \in \Delta_0^l(A)$) sin más que sustituir en la demostración las apariciones del literal $p$ por el literal $\neg p$ y tener en cuenta las equivalencias

$$\neg p \wedge \neg p \equiv \neg p; \qquad \neg p \wedge J_1 p \equiv \bot; \qquad p \wedge \neg J_1 p \equiv \neg p$$

6 a 8 Estos apartados son casos particulares, respectivamente, de los apartados 4, 5 y 6 del teorema 4.14.

*q.e.d.*

Este teorema, nos indica qué sustituciones son siempre beneficiosas y corresponderán al concepto de *0-reducción.*

El dual del teorema anterior nos permite mejorar las sustituciones del teorema 4.15 y, como consecuencia, mejorar el uso de $\Delta_1$. El interés de esta mejora radica en que, como hemos señalado, toda información de $\Delta_1$ es beneficiosa.

La extensión referida es el teorema 4.17 en el que se hace uso de la definición siguiente:

**Definición 4.19** Sea $A$ una fnu.

- Si la raíz de $T_A$ es $\vee$, definimos

$$\Delta_1^l(A) = \{\alpha p \mid \alpha \in \Lambda, p \in Q \text{ y } \alpha p \text{ es un hijo de la raíz de } T_A\}$$

- En otro caso, si la raíz de $T_A$ no es $\vee$, $\Delta_1^l(A) = \varnothing$.

**Teorema 4.17** *Sea $A$ una fnu y $p \in Q$ se tiene que:*

1. *Si* $\{p0, p\frac{1}{2}\} \subseteq \Delta_1(A)$ *entonces* $A \equiv A[p/\top] \vee \neg J_1 p$.
2. *Si* $\{p0, p1\} \subseteq \Delta_1(A)$ *entonces* $A \equiv A[p/\oslash] \vee \neg J_{\frac{1}{2}} p$.
3. *Si* $\{p\frac{1}{2}, p1\} \subseteq \Delta_1(A)$ *entonces* $A \equiv A[p/\bot] \vee \neg J_0 p$.

*Si existe $b \in$ **3** tal que $\{pb\} = \Delta_1(A) \cap P_{\mathbf{3}}$ se tiene que:*

4. *Si $b = 0$, se tiene que:*

*(i) Si* $\neg p \in \Delta_1^l(A)$*, entonces*

$$A \equiv \neg p \vee A[\![\neg p/\bot, J_0 p/\bot, \neg J_0 p/\top]\!]$$

*(ii) En otro caso,*

$$A \equiv J_0 p \vee A[\![J_0 p/\bot, \neg J_0 p/\top]\!]$$

5. *Si* $b = \frac{1}{2}$ *entonces*

$$A \equiv J_{\frac{1}{2}} p \vee A[\![J_{\frac{1}{2}} p/\bot, \neg J_{\frac{1}{2}} p/\top]\!]$$

6. *Si* $b = 1$*, se tiene que:*

*(i) Si* $p \in \Delta_1^l(A)$*, entonces*

$$A \equiv p \vee A[\![p/\bot, J_1 p/\bot, \neg J_1 p/\top]\!]$$

*(ii) En otro caso,*

$$A \equiv J_1 p \vee A[\![J_1 p/\bot, \neg J_1 p/\top]\!]$$

Hasta aquí, todas las transformaciones descritas para el proceso reducir permiten obtener una fórmula de menor tamaño en la que eliminamos ocurrencias de los literales trivaluados para cuyos sufijos existe información en los conjuntos $\Delta$. Todas estas transformaciones conservan el significado, es decir, transforman una fnu en otra equivalente.

Ahora bien, puesto que estamos diseñando un método por refutación, al igual que en el proceso *Actualizar*, podemos mejorar el uso *débil* de la información de los conjuntos $\Delta$ en transformaciones que , si bien no conservan el significado, sí conservan la casisatisfacibilidad. Esta mejora se recoge mediante el subproceso *0-reducción completa* que pasamos a describir y cuya justificación semántica viene dada por el siguiente teorema:

**Teorema 4.18** *Sea A una fnu y* $p \in Q$ *entonces:*

(1) *Si* $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$ *entonces* *A y* $A[p/\top]$ *son equicasisatisfacibles*

(2) *Si* $\{p0, p1\} \subseteq \Delta_0(A)$ *entonces* *A y* $A[p/\oslash]$ *son equicasisatisfacibles*

(3) *Si* $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$ *entonces* *A y* $A[p/\bot]$ *son equicasisatisfacibles*

Demostración:

1. Supongamos que $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(A)$.

   Sea $A$ casisatisfacible y sea $I$ una interpretación tal que $I(A) \in \{\frac{1}{2}, 1\}$. Puesto que, por el teorema 4.14, $A \equiv A[p/\top] \wedge J_1 p$, se tiene que $I(A[p/\top]) \in \{\frac{1}{2}, 1\}$, es decir, $A[p/\top]$ es casisatisfacible. Recíprocamente, supongamos que $A[p/\top]$ es casisatisfacible y sea $I$ una interpretación tal que $I(A[p/\top]) \in \{\frac{1}{2}, 1\}$. Entonces, para toda interpretación $I'$ tal que $I'(p) = 1$ e $I'(q) = I(q)$ para todo símbolo proposicional $q \in Q - \{p\}$ se tiene que $I'(A) \in \{\frac{1}{2}, 1\}$, es decir, $A$ es casisatisfacible.

2. Supongamos que $\{p0, p1\} \subseteq \Delta_0(A)$.

   Sea $A$ casisatisfacible y sea $I$ una interpretación tal que $I(A) \in \{\frac{1}{2}, 1\}$. Puesto que, por el teorema 4.14, $A \equiv A[p/\oslash] \wedge J_{\frac{1}{2}} p$, se tiene que $I(A[p/\oslash]) \in \{\frac{1}{2}, 1\}$, es decir, $A[p/\top]$ es casisatisfacible. Recíprocamente, supongamos que $A[p/\oslash]$ es casisatisfacible y sea $I$ una interpretación tal que $I(A[p/\oslash]) \in \{\frac{1}{2}, 1\}$. Entonces, para toda interpretación $I'$ tal que $I'(p) = \frac{1}{2}$ e $I'(q) = I(q)$ para todo símbolo proposicional $q \in Q - \{p\}$ se tiene que $I'(A) \in \{\frac{1}{2}, 1\}$, es decir, $A$ es casisatisfacible.

3. Supongamos que $\{p\frac{1}{2}, p1\} \subseteq \Delta_0(A)$.

   Sea $A$ casisatisfacible y sea $I$ una interpretación tal que $I(A) \in \{\frac{1}{2}, 1\}$. Puesto que, por el teorema 4.14, $A \equiv A[p/\bot] \wedge J_{\frac{1}{2}} p$, se tiene que $I(A[p/\bot]) \in \{\frac{1}{2}, 1\}$, es decir, $A[p/\bot]$ es casisatisfacible. Recíprocamente, supongamos que $A[p/\bot]$ es casisatisfacible y sea $I$ una interpretación tal que $I(A[p/\bot]) \in \{\frac{1}{2}, 1\}$. Entonces, para toda interpretación $I'$ tal que $I'(p) = 0$ e $I'(q) = I(q)$ para todo símbolo proposicional $q \in Q - \{p\}$ se tiene que $I'(A) \in \{\frac{1}{2}, 1\}$, es decir, $A$ es casisatisfacible.

*q.e.d.*

Podemos ya introducir las definiciones que intervienen en la descripción del proceso *Reducir*:

**Definición 4.20** Sea $A$ una fnu que no es ni una cláusula trivaluada restringida ni un cubo trivaluado restringido, decimos que $A$ es:

- 0-*reducible* si $\Delta_0^l(A) \neq \varnothing$.
- *1-reducible* si existe una subfórmula propia $B$ de $A$ que no es una cláusula trivaluada restringida y tal que $\Delta_1(B) \neq \varnothing$.
- *completamente 0-reducible* si existe $p \in Q$ tal que $|P_{\mathbf{3}} \cap \Delta_0(A)| = 2$.

- *reducible* si y sólo si es *0-reducible*, *1-reducible* o *completamente 0-reducible*.

**Definición 4.21** Si $A$ es reducible, *Reducir* $T_A$ significará:

a) Si $T_A$ es completamente 0-reducible, realizar las sustituciones determinadas por el Teorema 4.18 en $T_A$ (es decir, sustituir $A$ por una fórmula equicasisatisfacible donde no intervienen los símbolos de $\Delta_0(A)$) y guardar, para generar el contramodelo de la fórmula analizada, la siguiente información:

   Si $A$ es completamente 0-reducible respecto a un símbolo proposicional $p$, la información guardada es $P_{\mathbf{3}} - \Delta_0(A)$.

b) En otro caso, recorrer $T_A$ primero en profundidad realizando las sustituciones determinadas por el Teorema 4.16 en el primer subárbol propio 0-reducible o las sustituciones determinadas por el Teorema 4.17 en el primer subárbol propio 1-reducible.

   Para evitar bucles en la ejecución del método, cada vez que este tipo de reducción es realizada respecto a un símbolo proposicional $p$, se ha de incorporar una marca en los literales $\alpha p$ reducidos.

**Teorema 4.19** *El proceso* Reducir *mantiene la casisatisfacibilidad.*

DEMOSTRACIÓN: Es consecuencia inmediata de los Teoremas 4.18, 4.16 y 4.17 *q.e.d.*

### 4.11.2. El proceso *Reducir* obtiene cubos y cláusulas restringidos

Hasta aquí hemos visto que el proceso *Reducir*, cuando se produce una *completa 0-reducción*, elimina todas las apariciones del símbolo proposicional $p$ respecto del cual se realiza. En el caso de una *1-reducción* respecto a dos interpretaciones unitarias sobre el mismo símbolo proposicional $p$ en el conjunto $\Delta_1$, se deja una única aparición de este símbolo. Análogamente, para el caso de una *0-reducción* respecto a los literales $J_0p, J_{\frac{1}{2}}p$ o $J_1p$ en $\Delta_0^l$, se deja una única aparición de este símbolo. En los demás casos, basta con que $\Delta_1$ o $\Delta_0^l$ sean no vacíos para que el proceso pueda actuar.

Para terminar la descripción del proceso *Reducir*, destacamos una característica de este proceso, de interés en sí misma y de interés específico para la eficiencia de TAS-M**3**. Esta característica es que *Reducir* obtiene cubos y cláusulas *restringidos*, es decir, en cada cláusula o cubo a la que se aplica deja una sola ocurrencia de cada sufijo de literal trivaluado.

Una primera consecuencia de este hecho, que escapa al foco de interés de este trabajo, es que permite obtener especificaciones más simples. Concretamente, si ampliamos los conceptos de cubo y cláusula a una conjunción (respectivamente, disyunción) de literales y, posiblemente, la constante $\oslash$, y limitándonos a las transformaciones que intervienen en TAS-M**3** y que conservan el significado, podemos obtener una forma normal disyuntiva restringida (o, dualmente, conjuntiva restringida) equivalente a una fórmula dada.

Centrándonos en nuestro objetivo de la descripción de TAS-M**3**, el hecho de que el proceso *Reducir* proporcione cubos y cláusulas restringidas, tiene varias consecuencias de interés: en primer lugar disminuye el tamaño de la fórmula, y además, si hubiese posteriormente que distribuir, tenemos garantizado que si distribuimos respecto de una cláusula, en cada una de las tareas generadas, la cláusula sólo aportaría un símbolo proposicional, distinto para cada una de las tareas. De otro lado, el hecho de reducir el número de literales trivaluados con un mismo sufijo aumenta la probabilidad de que el proceso $\mathcal{S}$ actúe sobre él.

Los teoremas siguientes demuestran que, tras reiterar el proceso *Reducir* hasta que no pueda actuar más, se consigue que todos los cubos y cláusulas presentes en la fórmula en estudio sean restringidos.

**Teorema 4.20** *Dado un cubo trivaluado $C$, existe otro cubo trivaluado $C'$ que es restringido y tal que verifica las dos condiciones siguientes:*

1. *$C \equiv C'$ o bien $C \equiv C' \wedge \oslash$.*

2. *$C$ y $C'$ son equicasisatisfacibles.*

Demostración: Por la propiedad conmutativa de $\wedge$, podemos suponer que los literales con el mismo sufijo de literal en $C$ ocurren consecutivos. Para completar la demostración, basta destacar que, como se muestra en la siguiente tabla (que se completa por conmutatividad), la conjunción de dos literales con el mismo sufijo es equivalente a $\bot$, equivalente a un tercer literal con ese sufijo, o equivalente a $\oslash \wedge J_{\frac{1}{2}}p$. El proceso *Actualizar* eliminará $\oslash$ del cubo y por el teorema 4.9 su eliminación mantiene la casisatisfacibilidad.

| $\wedge$ | $\epsilon$ | $J_0$ | $J_{\frac{1}{2}}$ | $J_1$ | $\neg$ | $\neg J_0$ | $\neg J_{\frac{1}{2}}$ | $\neg J_1$ |
|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | $\epsilon$ | $\bot$ | $\oslash \wedge J_{\frac{1}{2}}$ | $J_1$ | $\oslash \wedge J_{\frac{1}{2}}$ | $\epsilon$ | $J_1$ | $\oslash \wedge J_{\frac{1}{2}}$ |
| $J_0$ | | $J_0$ | $\bot$ | $\bot$ | $J_0$ | $\bot$ | $J_0$ | $J_0$ |
| $J_{\frac{1}{2}}$ | | | $J_{\frac{1}{2}}$ | $\bot$ | $\oslash \wedge J_{\frac{1}{2}}$ | $J_{\frac{1}{2}}$ | $\bot$ | $J_{\frac{1}{2}}$ |
| $J_1$ | | | | $J_1$ | $\bot$ | $J_1$ | $J_1$ | $\bot$ |
| $\neg$ | | | | | $\neg$ | $\oslash \wedge J_{\frac{1}{2}}$ | $J_0$ | $\neg$ |
| $\neg J_0$ | | | | | | $\neg J_0$ | $J_1$ | $J_{\frac{1}{2}}$ |
| $\neg J_{\frac{1}{2}}$ | | | | | | | $\neg J_{\frac{1}{2}}$ | $J_0$ |
| $\neg J_1$ | | | | | | | | $\neg J_1$ |

*q.e.d.*

Dualmente se tiene el siguiente teorema:

**Teorema 4.21** *Dada una cláusula trivaluada $D$, existe otra cláusula trivaluada $D'$ que es restringida y tal que verifica las dos condiciones siguientes:*

1. *$D \equiv D'$ o bien $D \equiv D' \vee \oslash$..*
2. *$D$ y $D'$ son equicasisatisfacibles.*

Demostración: La demostración es dual de la anterior, y se utiliza la tabla que se muestra a continuación:

| $\vee$ | $\epsilon$ | $J_0$ | $J_{\frac{1}{2}}$ | $J_1$ | $\neg$ | $\neg J_0$ | $\neg J_{\frac{1}{2}}$ | $\neg J_1$ |
|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | $\epsilon$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_0$ | $\epsilon$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_0$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\top$ |
| $J_0$ | | $J_0$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_{\frac{1}{2}}$ | $\neg$ | $\top$ | $\neg J_{\frac{1}{2}}$ | $\neg J_1$ |
| $J_{\frac{1}{2}}$ | | | $J_{\frac{1}{2}}$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_0$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_1$ |
| $J_1$ | | | | $J_1$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_0$ | $\neg J_{\frac{1}{2}}$ | $\top$ |
| $\neg$ | | | | | $\neg$ | $\top$ | $\oslash \vee \neg J_{\frac{1}{2}}$ | $\neg J_1$ |
| $\neg J_0$ | | | | | | $\neg J_0$ | $\top$ | $\top$ |
| $\neg J_{\frac{1}{2}}$ | | | | | | | $\neg J_{\frac{1}{2}}$ | $\top$ |
| $\neg J_1$ | | | | | | | | $\neg J_1$ |

*q.e.d.*

El siguiente teorema nos muestra que el proceso *Reducir* consigue que los cubos y las cláusulas queden en forma restringida.

**Teorema 4.22**

1. *Si* $Reducir(T_A) = T_A$ *y* $C$ *es un cubo trivaluado que es subfórmula de* $A$*, entonces* $C$ *es restringido.*

2. *Si* $Reducir(T_A) = T_A$ *y* $C$ *es una cláusula trivaluada que es subfórmula de* $A$*, entonces* $C$ *es restringida.*

DEMOSTRACIÓN: Realizaremos la demostración considerando en primer lugar un cubo $A = \bigwedge_{i=1}^{n} l_i$. Razonamos por reducción al absurdo, es decir, suponemos que este cubo $A$ no es restringido y veamos que *Reducir* actúa en contradicción con la hipótesis de que $Reducir(T_A) = T_A$. En este caso, existe un símbolo proposicional $p$ que aparece como sufijo de al menos dos literales trivaluados de $A$. Teniendo en cuenta la definición de *Reducir*, podemos distinguir las siguientes posibilidades:

(I) Para cualquiera de los casos

   *a)* $J_0 p \in \Delta_0^l(A)$
   *b)* $J_{\frac{1}{2}} p \in \Delta_0^l(A)$
   *c)* $J_1 p \in \Delta_0^l(A)$

Llegaríamos a contradicción, ya que entonces se tienen, respectivamente, las inclusiones

$$\{p\frac{1}{2}, p1\} \subseteq \Delta_0^*(A), \quad \{p0, p1\} \subseteq \Delta_0^*(A) \text{ y } \{p0, p\frac{1}{2}\} \subseteq \Delta_0^*(A)$$

Por lo tanto, *Reducir* realizaría, respectivamente, las sustituciones

1. $A$ por $J_0 p \wedge A[p/\bot]$
2. $A$ por $J_{\frac{1}{2}} p \wedge A[p/\oslash]$
3. $A$ por $J_1 p \wedge A[p/\top]$

Ahora, puesto que las expresiones $[p/\bot], [p/\oslash], [p/\top]$ denotan sustituir todas las ocurrencias de $p$ por la constante indicada, el proceso *Reducir* habría sustituido cualquier otra aparición del sufijo $p$ y por tanto habría actuado.

(II) En otro caso, es decir, si los literales trivaluados con sufijo $p$ pertenecen al conjunto $\{\epsilon, \neg, \neg J_0, \neg J_{\frac{1}{2}}, \neg J_1\}$, podemos distinguir a su vez las siguientes posibilidades:

1. Para los casos $\neg J_0 p \in \Delta_0^l(A)$, $\neg J_{\frac{1}{2}} p \in \Delta_0^l(A)$ y $\neg J_1 p \in \Delta_0^l(A)$ tenemos asegurado por el teorema 4.16 que *Reducir* deja una sola ocurrencia de estos literales. Por lo tanto si *Reducir* no actúa es por que no existen literales repetidos de este tipo.

2. Para los casos, $p \in \Delta_0^l(A)$ y $\neg p \in \Delta_0^l(A)$ tenemos asegurado por el teorema 4.16 que *Reducir* deja una sola ocurrencia de estos literales y la desaparición del literal $\neg J_0 p$ para el primer caso y $\neg J_1 p$ en el segundo. Por lo tanto, si *Reducir* no actúa, es porque no existen literales repetidos $p$ ni $\neg p$ y, además, no existen ocurrencias simultáneas de $p$ y $\neg J_0 p$, ni ocurrencias simultáneas de $\neg p$ y $\neg J_1 p$.

3. En consecuencia, nos queda únicamente considerar el caso en el que en $A$ ocurren al menos dos literales de los conjuntos siguientes:

   *a*) $\{\neg J_0 p, \neg J_{\frac{1}{2}} p, \neg J_1\}$

   *b*) $\{p, \neg J_{\frac{1}{2}} p, \neg J_1\}$

   *c*) $\{\neg J_0, \neg J_{\frac{1}{2}}, \neg p\}$

   *d*) $\{p, \neg p\}$

   Por la asociatividad de $\wedge$, basta analizar las posibles combinaciones de dos ocurrencias cualesquiera de literales en estos conjuntos. Por otra parte, teniendo en cuenta el Teorema 4.16, agrupamos estas posibilidades como sigue:

   - Si se satisface una de las dos siguientes condiciones: siguientes
     1) $p, \neg p \in \Delta_0^l(A)$
     2) $\neg J_0 p, \neg J_1 p \in \Delta_0^l(A)$

     en ambos casos se tiene que $\{p0, p1\} \subseteq \Delta_0^*(A)$ y por tanto se realiza la sustitución $A$ por $J_{\frac{1}{2}} p \wedge A[p/\oslash]$ con lo que desaparecerían las demás ocurrencias del símbolo proposicional $p$.
   - Si se satisface una de las dos siguientes condiciones:
     1) $p, \neg J_{\frac{1}{2}} p \in \Delta_0^l(A)$
     2) $\neg J_0, \neg J_{\frac{1}{2}} p \in \Delta_0^l(A)$

     en ambos casos se tiene que $\{p0, p\frac{1}{2}\} \subseteq \Delta_0^*(A)$ y por tanto se realizaría la sustitución $A$ por $J_1 p \wedge A[p/\top]$ con lo que desaparecerían las demás ocurrencias del símbolo proposicional $p$.
   - Si se satisface una de las dos siguientes condiciones:

1) $\neg J_{\frac{1}{2}}p, \neg J_1 p \in \Delta_0^l(A)$
2) $\neg p, \neg J_{\frac{1}{2}}p \in \Delta_0^l(A)$

en ambos casos se tiene que $\{p\frac{1}{2}, p1\} \subseteq \Delta_0^*(A)$ y por tanto se realizaría la sustitución $A$ por $J_0 p \wedge A[p/\bot]$ con lo que desaparecerían las demás ocurrencias del símbolo proposicional $p$.

El razonamiento para una cláusula es dual, sin más que considerar en vez de el conjunto $\Delta_0^*(A)$ el conjunto $\Delta_1(A)$ y en vez de $\Delta_0^l(A)$ el conjunto $\Delta_1^l$.

*q.e.d.*

### 4.11.3. El proceso *Sintetizar*

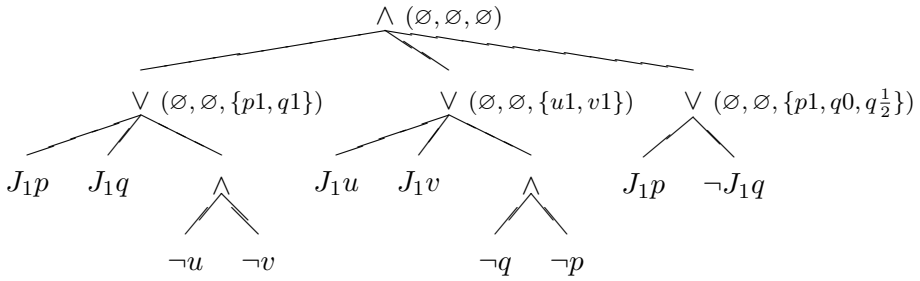
El proceso *Sintetizar* es una *reducción* parcial. Esta *reducción* parcial se realiza en el caso de que tengamos un nodo $\wedge$ con hijos $T_{A_i}, i \in J$ y varios de estos hijos verifican que $\bigcap_{j \in J'} \Delta_1^l(A_j) \neq \varnothing$. Para algún $J' \subsetneq J$ y $|J'| \geq 2$. El caso $J' = J$ provocaría una *reducción* total, que ya se habría realizado, por lo tanto no es el caso que nos ocupa.

En definitiva, el objetivo del proceso *Sintetizar* es rescatar *reducciones* que no pueden hacerse debido a la definición de los conjuntos $\Delta$ para la estructura de la fórmula $A$ analizada,

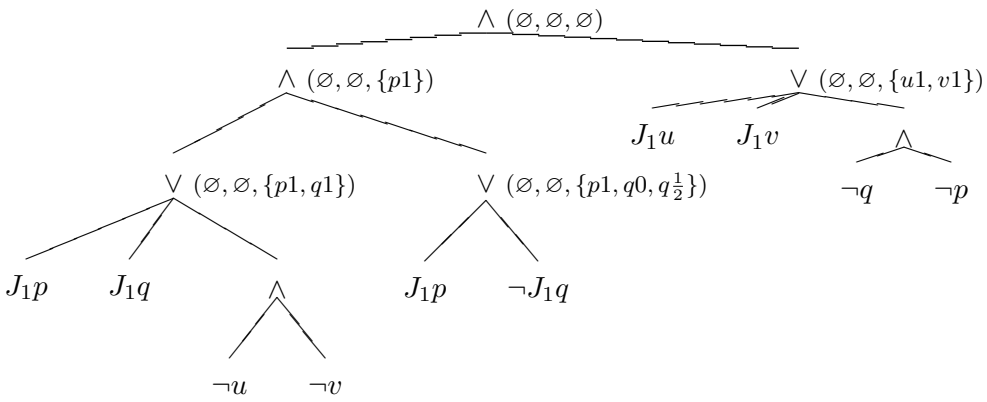
pero que sí son posibles sobre una fórmula $B$ equivalente a $A$ y sobre la que es fácil obtener la información que permite tales reducciones a partir de la información de las etiquetas de $T_A$.

Veamos un ejemplo que ilustra estos hechos.

**Ejemplo 4.11** Supongamos que tras *Signar* y *Etiquetar* un árbol $T_{\neg A}$ se ha obtenido el siguiente árbol $T_B$:




El árbol $T_B$ no es simplificable ni reducible. Ahora bien, podemos considerar el árbol $T_C$, con $C$ una fórmula equivalente a $B$, mostrado a continuación:




Este árbol sí es reducible, y el proceso *Reducir* podría ahora actuar sobre el hijo izquierdo de la raíz.

La detección de casos como el del ejemplo anterior puede ser incorporada al método analizando ocurrencias comunes de elementos *pb* en los conjuntos $\Delta_1$ de los hijos del nodo $\wedge$ (en el ejemplo anterior, $p1$). Ésta sería la extensión más general del correspondiente proceso *Sintetizar* para el caso proposicional clásico. Sin embargo, en aras de la eficiencia, nos limitaremos a extender el caso proposicional clásico contemplando únicamente las ocurrencias comunes de elementos $l$ en los conjuntos $\Delta_1^l$ de los hijos del nodo $\wedge$ (en el ejemplo anterior, $J_1p$). Específicamente, si $T_A$ es un árbol de raíz $\wedge$ con hijos $T_{A_i}, i \in I$ para mayor eficiencia buscamos el literal $l$ que más se repita en los conjuntos $\Delta_1^l(A_i)$ y respecto a este literal actuará el proceso *Sintetizar*. En otras palabras, *Sintetizar* no es más que la aplicación del proceso *Reducir* aplicado a la fórmula $\bigwedge_{j\in J} A_j \wedge \bigwedge_{i\in I-J} A_i$ donde $J$ es el subconjunto de $I$ tal que los subárboles $T_{A_j}$ satisfacen que $l \in \Delta_1^l(A_j))$.

Para definir formalmente el proceso *Actualizar* introducimos una definición auxiliar:

**Definición 4.22** Sea una fnu $A$ de la forma $A = \bigwedge_{i\in I} A_i$, a cada literal $l$ que ocurre en $A$ se le asocia un número natural $m(l, A) = |\{i \in I \mid l \in \Delta_1^l(A_i)\}|$.

Un literal $l$ se dice *destacado* para $A$ si se verifican las siguientes condiciones:

1. $A$ no es reducible.
2. $m(l, A) = \text{máx}\{m(l', A) \mid l' \text{ es un literal que ocurre en } A\}$.
3. $m(l, A) \geq 2$.

4. $l$ ocurre en la rama más izquierda entre los que verifican las tres propiedades anteriores.

Si $l$ es un literal destacado en $A$, se dice que $T_A$ es *sintetizable*.

**Teorema 4.23** *Sea una fnu $A$ de la forma $A = \bigwedge_{i\in I} A_i$, con $l$ un literal destacado en $A$ y sea $J = \{i \in I \mid l \in \Delta_1^l(A_i)\}$. Entonces:*

1. *Si* $l = \neg J_1 p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[p/\top]) \wedge (\bigwedge_{i\in I-J} A_i)$.
2. *Si* $l = \neg J_{\frac{1}{2}} p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[p/\oslash]) \wedge (\bigwedge_{i\in I-J} A_i)$.
3. *Si* $l = \neg J_0 p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[p/\bot]) \wedge (\bigwedge_{i\in I-J} A_i)$.
4. *Si* $l = \neg p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[\![\neg p/\bot, J_0 p/\bot, \neg J_0 p/\top]\!]) \wedge (\bigwedge_{i\in I-J} A_i)$.
5. *Si* $l = J_0 p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[\![J_0 p/\bot, \neg J_0 p/\top]\!]) \wedge (\bigwedge_{i\in I-J} A_i)$.
6. *Si* $l = J_{\frac{1}{2}} p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[\![J_{\frac{1}{2}} p/\bot, \neg J_{\frac{1}{2}} p/\top]\!]) \wedge (\bigwedge_{i\in I-J} A_i)$.
7. *Si* $l = p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[\![p/\bot, J_1 p/\bot, \neg J_1 p/\top]\!]) \wedge (\bigwedge_{i\in I-J} A_i)$.
8. *Si* $l = J_1 p$ *entonces* $A \equiv (l \vee (\bigwedge_{j\in J} A_j[\![J_1 p/\bot, \neg J_1 p/\top]\!]) \wedge (\bigwedge_{i\in I-J} A_i)$.

DEMOSTRACIÓN: Es inmediato a partir del Teorema 4.17 y de la definición de literal destacado. *q.e.d.*

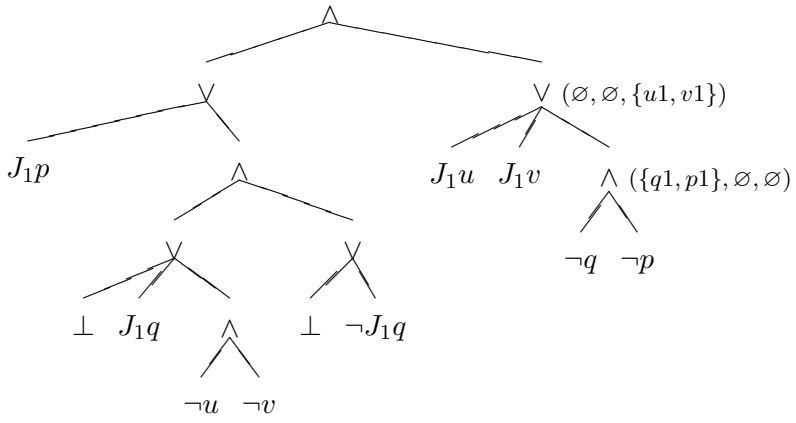
**Definición 4.23** *Sintetizar* $T_A$ es recorrer $T_A$ primero en profundidad hasta encontrar un subárbol $T_B$ con un literal destacado $l$ y realizar en $T_B$ las sustituciones determinadas por el Teorema 4.23.

**Teorema 4.24** *Si* Sintetizar $(T_A) = T_B$ *entonces $A$ y $B$ son equicasisatisfacibles.*

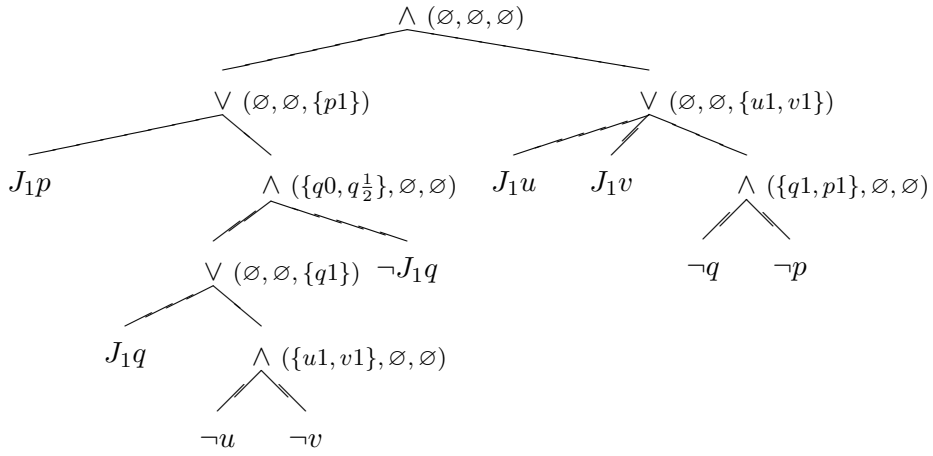
DEMOSTRACIÓN: Es consecuencia inmediata de la definición de *Sintetizar* y del Teorema 4.23. *q.e.d.*

**Ejemplo 4.12** Volviendo al ejemplo anterior, considerábamos el árbol $T_B$ con $B = C_1 \wedge C_2 \wedge C_3$ donde $C_1 = J_1 p \vee J_1 q \vee (\neg u \wedge \neg v), C_2 = J_1 u \vee J_1 v \vee (\neg q \wedge \neg p), C_3 =$

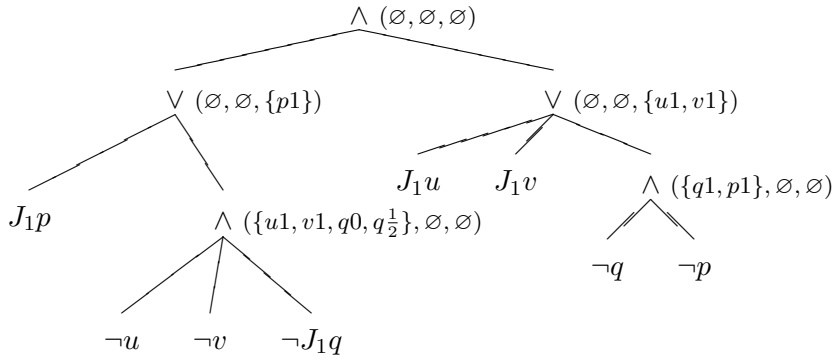
$J_1p \vee \neg J_1q$, que no era simplificable ni reducible. Sin embargo sí es sintetizable puesto que como $J_1p \in \Delta_1^l(C_1)$ y $J_1p \in \Delta_1^l(C_3)$, se tiene que $m(J_1p, B) = 2$ y es el máximo, ya que para los otros literales vale 1 ó 0, y como $B$ no es reducible, tenemos entonces que $J_1p$ es un literal destacado en $B$. Aplicando a $T_B$ el proceso *Sintetizar*, se obtiene el siguiente árbol $T_D$ que se muestra a continuación.




El proceso *Sintetizar* ha generado varias constantes en el árbol, el encargado de eliminarlas así como de calcular las nuevas etiquetas y realizar las correspondientes simplificaciones si las hubiese es el proceso *Actualizar*. Tras *Actualizar*se obtiene el árbol $T_E$ que se muestra a continuación:



Este árbol es reducible, ya que tenemos $\{q0, q\frac{1}{2}\} \subseteq \Delta_0((J_1 q \vee (\neg u \wedge \neg v)) \wedge \neg J_1 q)$, aplicando el proceso *Reducir* y posteriormente *Actualizar* se obtiene el árbol siguiente $T_F$:




El proceso $\mathcal{F}$ no realiza más acciones sobre este árbol, y tal como era de esperar, algunos casos necesitan distribuir. Veremos que esta distribución se realizará de forma controlada con el proceso ($\wedge$-$\vee$) que se verá en la siguiente sección, donde retomaremos este ejemplo.

## 4.12. El proceso ($\wedge$-$\vee$)

La labor del proceso ($\wedge$-$\vee$) es realizar las distribuciones de $\wedge$ sobre $\vee$, cuando éstas no pueden evitarse. En principio se realiza una sola distribución, y tras aplicar la generación de tareas, se aplican otra vez las reducciones determinadas por $\mathcal{F}$ a los árboles hijos de la raíz $\vee$ (tareas). De este modo, puede terminar el proceso, o en caso contrario reducir el tamaño de las tareas obtenidas y, en consecuencia, evitar futuras distribuciones.

Hasta la entrada en ($\wedge$-$\vee$), únicamente hemos utilizado procesos "tratables", sin realizar ninguna distribución. El peso de la complejidad exponencial recae sobre este proceso. Sin embargo, la filosofía del método es evitar cuantas distribuciones sea posible, por eso no sólo realizamos reducciones mediante el proceso $\mathcal{F}$ antes de entrar en ($\wedge$-$\vee$), sino que a cada una de las tareas que se obtienen al realizar una única distribución y generar tareas, se les vuelve a aplicar $\mathcal{F}$.

La tarea de distribución mantiene el significado y es la traducción en términos de árboles sintácticos de la ley distributiva de $\wedge$ sobre $\vee$ (realizada, a efectos de la explicación del método, sobre la rama más a la izquierda). Esta transformación se denomina ($\wedge$-$\vee$) y su forma de actuar se representa en la figura 4.5 donde el árbol de la izquierda es la entrada de ($\wedge$-$\vee$) y el árbol de la derecha es la salida.

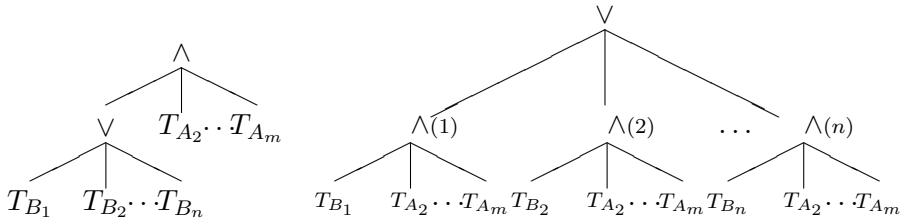



Figura 4.5: La transformación (∧-∨).

En la implementación del método, y a falta de un análisis más fino de eficiencia, se distribuye respecto de la rama determinada por la cardinalidad de los conjuntos $\Delta$ (eligiendo la de mayor cardinal) y el número de ocurrencias de los elementos de estos conjuntos $\Delta$ en el árbol total sobre el que se va a realizar la distribución.

El siguiente resultado elemental y de demostración inmediata (incluido tan sólo a efectos de posteriores referencias) establece que el proceso (∧-∨) mantiene el significado:

**Teorema 4.25** *Si* $(\wedge - \vee)(T_B) = T_C$ *entonces* $B$ *es lógicamente equivalente a* $C$.

La descripción del proceso (∧-∨) requiere algunos comentarios. Los métodos TAS introducidos hasta ahora por nuestro grupo de investigación, utilizaban la interacción de (∧-∨) y $\mathcal{F}$ como únicas herramientas para evitar distribuciones. Sin embargo, el análisis realizado en el presente trabajo nos ha permitido introducir una nueva estrategia de eficiencia, aplicable a todos los métodos TAS (su incorporación en el caso proposicional clásico se muestra en el Apéndice A de este trabajo) y que, en nuestra opinión, aporta una sustancial mejora a nuestra metodología. No es de extrañar que esto ocurra; abordar el diseño de un demostrador para una lógica de mayor complejidad da lugar a una mayor motivación de cara a que ésta no de lugar a una desproporcionada mayor complejidad para el demostrador y en consecuencia a un mayor grado de perspicacia.

Esta nueva estrategia se incorpora mediante el subproceso que denominaremos *Depurar* y cuya incorporación responde al siguiente análisis:

Supongamos que un árbol $T_A$ es la salida del proceso $\mathcal{F}$ y tiene ∧ como raíz. Este árbol será la entrada del proceso (∧-∨), en el que se realizará una distribución (ya que su raíz es ∧), esta distribución, como ya hemos indicado, se hace únicamente respecto de una rama para posteriormente aplicar sobre cada subtarea generada las reducciones que realiza $\mathcal{F}$. Sin embargo, cabe preguntarse si podemos extraer aún más información de los conjuntos $\Delta$ para que esta distribución se realice de forma óptima, es decir, intentando que el número total

de distribuciones sea el menor posible. La respuesta a la pregunta anterior es afirmativa. Más aún, como veremos, además de evitar distribuciones, podemos utilizar esta información para eliminar totalmente un símbolo proposicional $p$.

El modo de incorporar la información citada es, en síntesis, una potente extensión del teorema de Davis y Putnam (P. Gochet y et al., 1988) para el estudio de la satisfacibilidad de un conjunto de cláusulas. Para una mayor comprensión del interés de esta estrategia, es conveniente el conocimiento previo del caso proposicional clásico (incluido en el Apéndice A).

### 4.12.1. El proceso *Depurar*

Este proceso no mantiene la equivalencia lógica pero sí la casisatisfacibilidad. Para su aplicabilidad es preciso exigir ciertas condiciones sobre los conjuntos $\Delta_1$ de los hijos de la raíz $\wedge$ del árbol considerado. El siguiente teorema nos muestra las condiciones requeridas así como la transformación que debe realizarse en el árbol.

**Notación:**

En adelante usaremos $[n]$ para denotar $\{1, \ldots, n\}$

**Teorema 4.26** *Sean* $A = \bigwedge_{i=1}^{n} A_i$ *una fnu,* $p \in Q$ *un símbolo proposicional y sean* $i_1, i_2 \in [n]$ *índices tales que*

$$|P_{\mathbf{3}} \cap \Delta_1(A_{i_1})| = 2, \quad |P_{\mathbf{3}} \cap \Delta_1(A_{i_2})| = 2 \text{ y } P_{\mathbf{3}} \subseteq \Delta_1(A_{i_1}) \cup \Delta_1(A_{i_2})$$

*Entonces se tiene que*

$$A \quad y \quad \Big(\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]\Big) \vee \Big(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]\Big) \vee \Big(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3]\Big)$$

*son equicasisatisfacibles, donde*

$$\odot_j = \begin{cases} \top & si \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p\tfrac{1}{2}\} \\ \oslash & si \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p1\} \\ \bot & si \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p\tfrac{1}{2}, p1\} \end{cases} \qquad \text{para } j \in \{1, 2\}$$

*y*

$$\odot_3 = \begin{cases} \top & si \quad P_{\mathbf{3}} \cap (\Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})) = \{p1\} \\ \oslash & si \quad P_{\mathbf{3}} \cap (\Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})) = \{p\tfrac{1}{2}\} \\ \bot & si \quad P_{\mathbf{3}} \cap (\Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})) = \{p0\} \end{cases}$$

DEMOSTRACIÓN: Por el Teorema 4.17 se tiene que $A_{i_1} \equiv l_1 \vee A_{i_1}[p/\odot_1]$ y $A_{i_2} \equiv l_2 \vee A_{i_2}[p/\odot_2]$, donde

$$l_j = \begin{cases} \neg J_1 p & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p\frac{1}{2}\} \\ \neg J_{\frac{1}{2}} p & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p1\} \\ \neg J_0 p & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p\frac{1}{2}, p1\} \end{cases}$$

y

$$\odot_j = \begin{cases} \top & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p\frac{1}{2}\} \\ \oslash & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p1\} \\ \bot & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p\frac{1}{2}, p1\} \end{cases}$$

con $j \in \{1, 2\}$. Por las leyes asociativa y conmutativa y el teorema de equivalencia se tiene que

$$A \equiv (l_1 \vee A_{i_1}[p/\odot_1]) \wedge (l_2 \vee A_{i_2}[p/\odot_2]) \wedge \bigwedge_{i \in [n]-\{i_1,i_2\}} A_i$$

Podemos demostrar ya la afirmación del teorema.

1. Si $A$ es casisatisfacible, entonces existe una interpretación $I$ tal que

   $$I(l_1 \vee A_{i_1}[p/\odot_1]) = b_1, \quad I(l_2 \vee A_{i_2}[p/\odot_2]) = b_2 \text{ y } I(\bigwedge_{i \in [n]-\{i_1,i_2\}} A_i) = b_3$$

   con $b_1, b_2, b_3 \in \{\frac{1}{2}, 1\}$.

   Por otra parte, por las hipótesis del teorema, a fortiori tiene que existir un $b \in \mathbf{3}$ tal que $pb \in \Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})$

   Ahora tenemos que probar que

   $$(\bigwedge_{i \in [n]-\{i_2\}} A_i[p/\odot_1]) \vee (\bigwedge_{i \in [n]-\{i_1\}} A_i[p/\odot_2]) \vee (\bigwedge_{i \in [n]-\{i_1,i_2\}} A_i[p/\odot_3])$$

   es casisatisfacible.

   Distinguimos dos casos:

   (i) Si $I(p) = b$ entonces $I(\bigwedge_{i \in [n]-\{i_1,i_2\}} A_i[p/\odot_3]) = b_3$, donde

   $$\odot_3 = \begin{cases} \top & \text{si} \quad b = 1 \\ \oslash & \text{si} \quad b = \frac{1}{2} \\ \bot & \text{si} \quad b = 0 \end{cases}$$

(ii) Supongamos $I(p) = b' \neq b$. En este caso $pb'$ pertenece o bien a $\Delta_1(A_{i_1})$ o bien a $\Delta_1(A_{i_2})$ pero no a los dos.

Supongamos que $pb' \in \Delta_1(A_{i_1})$ y que $pb' \notin \Delta_1(A_{i_2})$, por tanto $I(l_2) \neq 1$, pero observando la semántica de los literales $\neg J_0$, $\neg J_1$ y $\neg J_{\frac{1}{2}}$ se observa que $I(l_2) \neq \frac{1}{2}$ y por tanto $I(l_2) = 0$, de esta igualdad y de $I(l_2 \vee A_{i_2}[p/\odot_2]) = b_2$ se tiene que $I(A_{i_2}[p/\odot_2]) = b_2$.

Por otra parte, puesto que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = b_3$ y $I(p) = b'$ y $pb' \notin \Delta_1(A_{i_2})$, se tiene que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_2]) = b_3$ y uniendo los dos resultados anteriores se tiene que

$$I(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \neq 0$$

Para el caso en el que $pb' \in \Delta_1(A_{i_2})$ y que $pb' \notin \Delta_1(A_{i_1})$ se obtiene de forma análoga que

$$I(\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]) \neq 0$$

y por tanto se tiene asimismo que

$$I((\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1])) \neq 0$$

De los dos casos anteriores se concluye que

$$I((\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]) \vee (\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3])) \neq 0$$

y por lo tanto que es casisatisfacible la fórmula

$$(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]) \vee (\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3])$$

2. Recíprocamente, si la fórmula

$$(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]) \vee (\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3])$$

es casisatisfacible, se tiene que al menos una de las tres fórmulas siguientes es casisatisfacible:

*a)* $\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]$

*b)* $\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]$

*c)* $\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3]$

(a) Supongamos que $\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]$ es casisatisfacible y sea $I$ una interpretación tal que $I(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \neq 0$; por lo tanto

$$I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_2]) \neq 0 \qquad \text{y} \qquad I(A_{i_2}[p/\odot_2]) \neq 0$$

Consideremos la extensión de $I$ definida por

$$I'(q) = \begin{cases} I(q) & \text{si} \quad q \neq p \\ v & \text{si} \quad q = p \text{ y } pv \in \Delta_1(A_{i_1}), \text{ con } v \neq 0 \end{cases}$$

Veamos que $I'(A) \neq 0$.

- Puesto que $p$ no ocurre en $\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_2]$, se tiene que

$$I'(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_2]) \neq 0$$

- Puesto que $p$ no ocurre en $A_{i_2}[p/\odot_2]$, se tiene que

$$I'(A_{i_2}) = I(A_{i_2}[p/\odot_2]) \neq 0$$

- Por último, puesto que $pv \in \Delta_1(A_{i_1})$ y $I'(p) = v$, se tiene que $I'(A_{i_1}) = v \neq 0$

Por lo tanto $I'(A) \neq 0$ y $A$ es casisatisfacible.

(b) El caso en el que $\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]$ es casisatisfacible se razona de forma análoga al razonamiento anterior.

(c) Si $\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3]$ es casisatisfacible, existe una interpretación $I$ tal que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3]) \neq 0$. Sea $b \in \mathbf{3}$ tal que $pb \in \Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})$. Consideremos la extensión de $I$ definida por

$$I'(q) = \begin{cases} I(q) & \text{si} \quad q \neq p \\ b & \text{si} \quad q = p \end{cases}$$

Veamos que $I'(A) \neq 0$.

- Puesto que $I'(p) = b$, se tiene que

$$I'(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3]) \neq 0$$

- Puesto que $I'(p) = b$ y $pb \in \Delta_1(A_{i_2})$, se tiene que $I'(A_{i_2}) = 1$.
- Por último, de $I'(p) = b$ y $pb \in \Delta_1(A_{i_1})$, se tiene que $I'(A_{i_1}) = 1$.

Por lo tanto $I'(A) \neq 0$ y $A$ es casisatisfacible.

*q.e.d.*

Para referirnos a las condiciones que establece el teorema para poder aplicar las transformaciones indicadas, introducimos las siguientes definiciones:

**Definición 4.24** Un árbol sintáctico $T_A$, con $A = \bigwedge_{i=1}^{n} A_i$ una fnu se dice *depurable* si existe $p \in Q$ un símbolo proposicional e índices $i_1, i_2 \in [n]$ tales que $|P_{\mathbf{3}} \cap \Delta_1(A_{i_1})| = 2$, $|P_{\mathbf{3}} \cap \Delta_1(A_{i_2})| = 2$ y $P_{\mathbf{3}} \subseteq \Delta_1(A_{i_1}) \cup \Delta_1(A_{i_2})$.

**Definición 4.25** Sea $T_A$ un árbol *depurable* con $A = \bigwedge_{i=1}^{n} A_i$ fnu, $p \in Q$, $i_1, i_2 \in [n]$ tales que $|P_{\mathbf{3}} \cap \Delta_1(A_{i_1})| = 2$, $|P_{\mathbf{3}} \cap \Delta_1(A_{i_2})| = 2$ y $P_{\mathbf{3}} \subseteq \Delta_1(A_{i_1}) \cup \Delta_1(A_{i_2})$. Por *Depurar* entenderemos sustituir el árbol $T_A$ por el árbol $T_B$, donde:

$$B = (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\odot_1]) \vee (\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\odot_2]) \vee (\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\odot_3])$$

con

$$\odot_j = \begin{cases} \top & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p\tfrac{1}{2}\} \\ \oslash & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p0, p1\} \\ \bot & \text{si} \quad P_{\mathbf{3}} \cap \Delta_1(A_{i_j}) = \{p\tfrac{1}{2}, p1\} \end{cases} \qquad \text{para } j \in \{1,2\}$$

*y*

$$\odot_3 = \begin{cases} \top & \text{si} \quad P_{\mathbf{3}} \cap (\Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})) = \{p1\} \\ \oslash & \text{si} \quad P_{\mathbf{3}} \cap (\Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})) = \{p\tfrac{1}{2}\} \\ \bot & \text{si} \quad P_{\mathbf{3}} \cap (\Delta_1(A_{i_1}) \cap \Delta_1(A_{i_2})) = \{p0\} \end{cases}$$

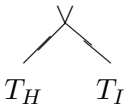
Con esta terminología el teorema 4.26 puede expresarse diciendo que si $T_A$ es depurable y $Depurar(T_A) = T_B$ entonces $A$ y $B$ son casisatisfacibles.

Una vez introducidas la transformación (∧-∨) y el proceso *Depurar*, queda completo el cuadro de procesos que intervienen en TAS-M**3**.

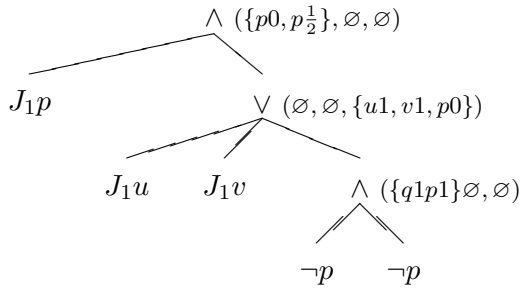
Según se observa en el diagrama de flujo de TAS-M**3** en la figura **??** las posibles salidas pueden ser:

- "NO VÁLIDA" si alguna de las tareas generadas tiene como salida "NO VÁLIDA".
- "VÁLIDA" si todas las subtareas tienen como salida "VÁLIDA".

**Ejemplo 4.13** Continuando con el ejemplo 4.12, habíamos obtenido el árbol $T_G$ con $G = (J_1p \vee (\neg u \wedge \neg v \wedge J_1q)) \wedge (J_1u \vee J_1v \vee (\neg q \wedge \neg p))$, que es la entrada del proceso (∧-∨). Este proceso realiza una distribución de ∧ sobre ∨ y se obtiene el árbol:




donde $H = J_1p \wedge (J_1u \vee J_1v \vee (\neg q \wedge \neg p))$ e $I = (\neg u \wedge \neg v \wedge J_1q) \wedge (J_1u \vee J_1v \vee (\neg q \wedge \neg p))$. La raíz de este árbol es ∨ por lo que se genera una tarea para el subárbol $T_H$ y otra para el $T_I$. Veamos la traza de ejecución de la primera tarea (correspondiente a $T_H$) es:




Este árbol $T_H$ tiene dos símbolos de tipo 1: $u$ y $v$; por lo tanto la transformación $\mathcal{S}$ sustituye las ocurrencias de $u$ y de $v$ por $\top$, y tras *Actualizar* se obtiene

$$J_1p$$

que es finalizable ya que el conjunto $\Delta_1$ de la raíz es $\{p1\} \neq \varnothing$. La salida parcial de esta tarea es "NO VÁLIDA" lo que hace que no se ejecuten más tareas y se dé la salida general "NO VÁLIDA".

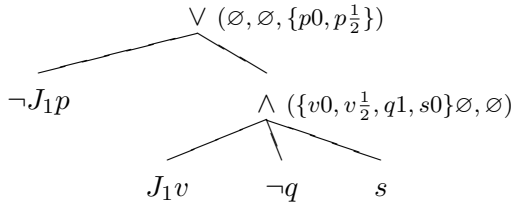
Veamos ahora otro ejemplo que muestra la potencia de *Depurar*.

**Ejemplo 4.14** Supongamos que la salida de $\mathcal{F}$ es el árbol $T_A$ donde $A = B \wedge C \wedge D$ con

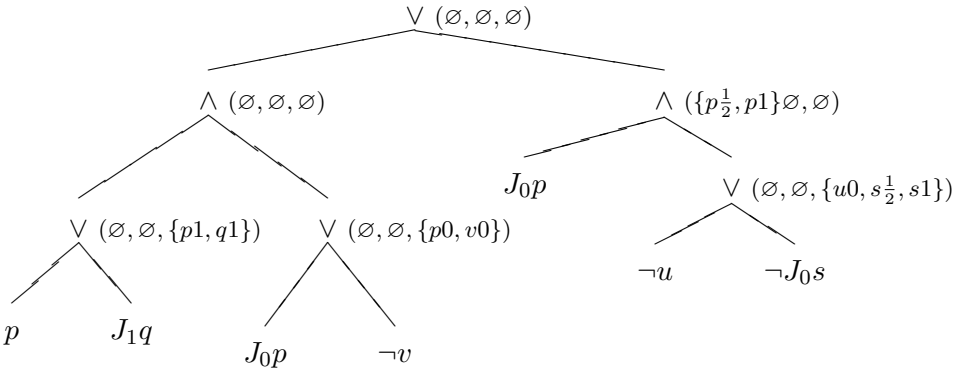
$B = (\neg J_1 p \vee (J_1 v \wedge \neg q \wedge s))$,
$C = (((p \vee J_1 q) \wedge (J_0 p \vee \neg v)) \vee (J_0 p \wedge (\neg u \vee \neg J_0 s)))$
$D = (\neg J_0 p \vee (J_1 q \wedge J_0 s \wedge J_1 u))$

A continuación se muestran los árboles $T_B$, $T_C$ y $T_D$ etiquetados:

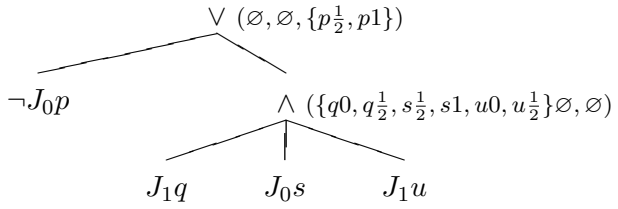
$T_B =$



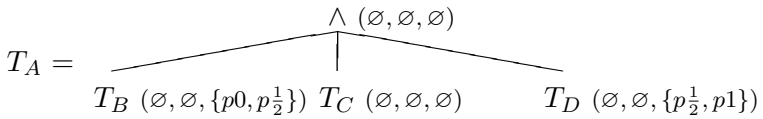

$T_C =$




$T_D =$

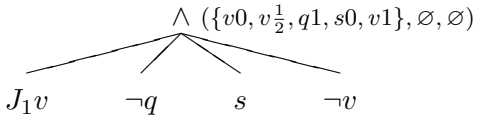



Por lo tanto, el árbol etiquetado de $T_A$ es de la forma:

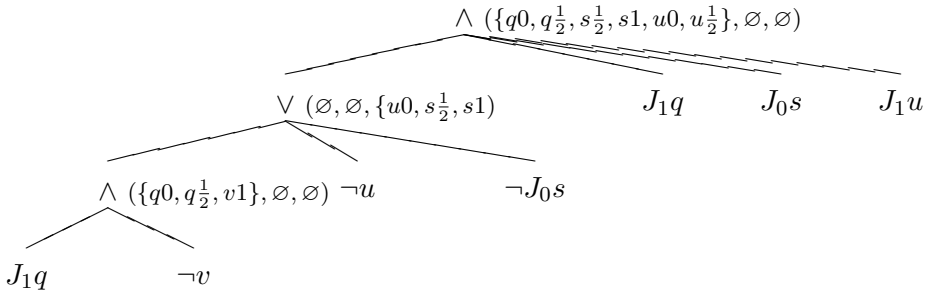



$T_A$ es depurable ya que $\{p0, p1\} \subseteq \Delta_1(B)$ y $\{p\frac{1}{2}, p1\} \subseteq \Delta_1(D)$, por lo tanto se ejecuta *Depurar* y como la raíz del árbol resultante es $\vee$ se generan tres tareas, la primera corresponde a $T_{(B\wedge C)[p/\top]}$, la segunda a $T_{(C\wedge D)[p/\bot]}$ y la tercera a $T_{C[p/\varnothing]}$.

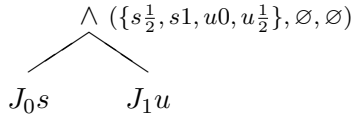
La primera tarea corresponde al siguiente árbol ya etiquetado:




Este árbol $T_{B_1}$ es *0-concluyente* ya que $V_{\mathbf{3}} \subseteq \Delta_0(B_1)$. Por tanto este árbol se sustituye por $\bot$. Y pasa a ser *finalizable*, obteniéndose la salida parcial "VÁLIDA", con lo que debemos seguir con las demás tareas: Para la segunda tarea obtenemos el árbol:




En este árbol $T_{B_2}$ el símbolo proposicional $q$ es de tipo 1, y $v$ es de tipo 0, por tanto se sustituyen los literales donde aparecen por $\top$ y se obtiene el árbol:



En este árbol, el símbolo proposicional $s$ es de tipo 0, y $u$ es de tipo 1, por tanto, se sustituyen por $\top$, los literales donde aparecen estos símbolos y se obtiene el árbol:

$$\top$$

que es finalizable. Se obtiene así la salida parcial "NO VÁLIDA" dando lugar a que termine TAS-M**3** con salida "NO VÁLIDA", sin necesidad de procesar la tercera tarea, que tendría como árbol $\neg v$.

## 4.13. Corrección y completitud de TAS-M3

Una vez introducidos todos los procesos que constituyen el método TAS-M**3**, y observado que cualquier transformación mantiene la casisatisfacibilidad, estamos en condiciones de afirmar la corrección y completitud del método mediante el siguiente teorema:

**Teorema 4.27** *Una fbf $A$ es válida si y sólo si TAS-M**3**$(T_{\neg A})$ = "VÁLIDA".*

Demostración: El método TAS-M**3** para tras la aplicación de un número finito de procesos, dando como salida "VÁLIDA" o bien "NO VÁLIDA", ya que los procesos que no son ($\wedge$-$\vee$) ni *Depurar* reducen el tamaño de la fórmula y el número de distribuciones posibles de $\wedge$ sobre $\vee$ es finito. Veamos que en la última disyunción se tiene que producir una de estas dos salidas:

1. Si en la disyunción final hubiese algún disyunto que es un cubo, por el Teorema 4.22 éste será restringido y se le podría aplicar el proceso $\mathcal{S}$, que generaría $\top$.

2. Si en la disyunción final hay un disyunto que es un literal trivaluado, el árbol correspondiente a su tarea es finalizable, ya que el conjunto $\Delta_1$ de su raíz es no vacío, y el método termina con la salida "NO VÁLIDA" por el proceso *Actualizar*.

3. Por último, si el árbol final es una constante, entonces es finalizable y se produce una de las dos salidas: "VÁLIDA" o "NO VÁLIDA" dependiendo de que la constante sea o no $\bot$.

Por lo tanto sólo hay que probar los dos puntos siguientes:

- Si TAS-D$(T_{\neg A})$ ="VÁLIDA" entonces $A$ es válida.
- Si TAS-D$(T_{\neg A})$ ="NO VÁLIDA" entonces $A$ no es válida.

Para ello es suficiente observar que:

1. Todos los procesos que componen el método TAS-M**3** conservan la casisatisfacibilidad según nos aseguran los teoremas: 4.3, 4.10, 4.13, 4.19, 4.24, 4.25 y 4.26.

2. Una fbf $C = \bigwedge_{i=1}^{n} B_i$ es casisatisfacible si y sólo si $B_i$ es casisatisfacible para todo $i \in [n]$

3. La salida "VÁLIDA" si todas las tareas que se generan tienen como salida "VÁLIDA", lo que lleva a que en último término el árbol correspondiente a esta tareas haya sido reducido a la constante $\perp$. Por lo tanto la fórmula $\neg A$ correspondiente al árbol de entrada es equicasisatisfacible con $\perp$ y en consecuencia no es casisatisfacible y $A$ es válida.

4. La salida "NO VÁLIDA" se produce si alguna de las tareas generadas finaliza con salida "NO VÁLIDA", es decir alguno de los árboles hijos de la raíz $\vee$ es $\top$ o tiene un conjunto $\Delta_1 \neq \varnothing$ o $\Delta_{\frac{1}{2}} \neq \varnothing$ asociado a su raíz. En cualquier caso, esto nos lleva a que la fórmula $\neg A$ asociada al árbol de entrada es casisatisfacible y por tanto que $A$ no es válida.

*q.e.d.*

## 4.14. TAS-M3 proporciona un contramodelo

En el caso de que TAS-M**3** pare con salida "NO VÁLIDA" se ofrece además un contramodelo para la fórmula $A$, cuya negación es la entrada de TAS-M**3**. La construcción de modelos se realiza de forma natural a través del árbol de modelos que se define de forma análoga al caso proposicional clásico (incluido en el Apéndice A). Una diferencia importante respecto al caso proposicional es que la valuación del contramodelo de $A$, aplicada a $\neg A$ es un elemento del conjunto $\{\frac{1}{2}, 1\}$ y no únicamente el valor 1 como en el caso clásico. Pasamos a definir ahora el procedimiento de construcción del árbol de modelos:

**Definición 4.26** Un *árbol de modelos* asociado a la ejecución de TAS-M**3** sobre un árbol $T_B$, es un árbol $T_{\mathcal{M}}$ cuyos nodos son conjuntos de elementos de la forma $pb$ con $p \in Q$ y $b \in \mathbf{3}$ y que es generado durante la ejecución de TAS-M**3** mediante la asociación de sus nodos a los árboles obtenidos en tal ejecución. La generación de $T_{\mathcal{M}}$ se realiza del modo descrito a continuación:

1. Al iniciarse la ejecución de TAS-D, $T_{\mathcal{M}}$ es un árbol cuyo único nodo es el conjunto vacío y está asociado al árbol de entrada a TAS-M**3**, $T_B$.

2. Para cada etapa de ejecución de TAS-M**3**:

- Si en la ejecución de TAS-M**3**, a partir de un árbol $T_C$ se generan las tareas $T_{C_1}, T_{C_2}, \ldots, T_{C_n}$, se añaden $n$ hijos en $T_{\mathcal{M}}$ como descendientes del nodo asociado al árbol $T_C$. Estos hijos serán:
  *a*) Todos ellos $\varnothing$, si no se ha aplicado el proceso *Depurar*.
  *b*) En caso contrario, es decir, si se ha aplicado *Depurar*, se tiene que $n = 3$ y los hijos serán:
    - $\{p0\}$ para el hijo asociado a la tarea en la que se sustituyó el símbolo $p$ por $\bot$.
    - $\{p\frac{1}{2}\}$ para el hijo asociado a la tarea en la que se sustituyó el símbolo $p$ por $\oslash$.
    - $\{p1\}$ para el hijo asociado a la tarea en la que se sustituyó el símbolo $p$ por $\top$.
- Si en la ejecución de TAS-M**3** se obtiene un árbol finalizable $T_F$ que da salida "NO VÁLIDA", se tiene que dar al menos uno de los siguientes casos:
  *a*) Si $T_F = \top$ entonces el nodo asociado queda tal como está.
  *b*) Si $T_F = \oslash$ entonces el nodo asociado queda tal como está.
  *c*) Si $\Delta_1(F) \neq \varnothing$, entonces se añade a su nodo asociado en $T_{\mathcal{M}}$ el primer elemento en orden lexicográfico de $\Delta_1(F)$.
  *d*) Si $\Delta_{\frac{1}{2}}(F) \neq \varnothing$, entonces se añade a su nodo asociado en $T_{\mathcal{M}}$ el primer elemento en orden lexicográfico de $\Delta_{\frac{1}{2}}(F)$.

  Si ocurren simultáneamente (c) y (d), se considera únicamente (c).
- El resto de los procesos de TAS-M**3**, son de dos tipos, los que mantienen la equivalencia lógica y los que sólo conservan la casisatisfacibilidad. Veamos qué sucede con cada uno de ellos:
  i) Los procesos que sólo mantienen la casisatisfacibilidad y no la equivalencia lógica, sustituyen todas las apariciones de un símbolo proposicional por las constantes $\top$ o $\bot$ en todo el árbol. Los procesos de este tipo son la *completa 0-reducción*, $\mathcal{S}$ y *Depurar*. Puesto que las acciones para el proceso *Depurar* ya han sido explicadas dentro del apartado de generación de tareas, en este apartado sólo nos queda considerar los procesos *completa 0-reducción* y $\mathcal{S}$. En estos casos, si $T_G$ es la entrada de uno de estos procesos y $T_H$ la salida, el nodo asociado a $T_G$ pasa a asociarse a $T_H$ y a este nodo se le añade para cada símbolo proposicional $p$ eliminado, el elemento $p0$ si $p$ ha sido sustituido en todas sus ocurrencias por $\bot$, el elemento $p\frac{1}{2}$ si $p$ ha sido sustituido por $\oslash$ y el elemento $p1$ si $p$ ha sido sustituido por $\top$.

ii) Si un proceso transforma un árbol $T_D$ en otro $T_E$, manteniendo la equivalencia lógica entre $D$ y $E$ el nodo asociado a $T_D$ pasa a ser asociado a $T_E$ sin modificación alguna en su contenido.

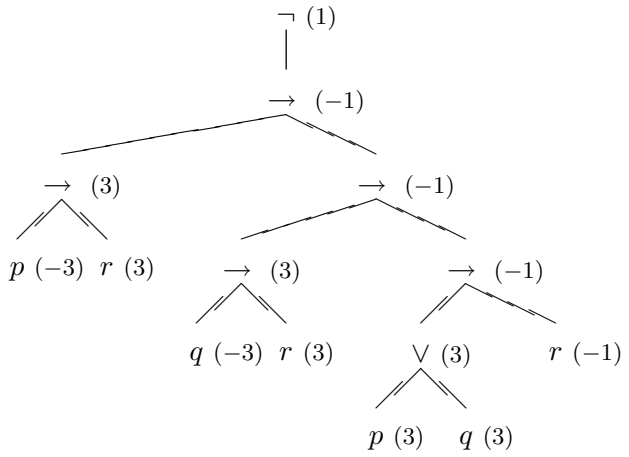
Si $T_B$ es el árbol de entrada en el método TAS-M**3** y éste para con salida "NO VÁLIDA" es porque el proceso *Actualizar* ha detectado que el conjunto $\Delta_1$ o el conjunto $\Delta_{\frac{1}{2}}$ de la raíz de un árbol completo es distinto del vacío y por tanto que es finalizable, o bien porque el árbol esté constituido por un único nodo que sea o bien la constante $\oslash$ o bien la constante $\top$. En este caso, asociado a este árbol tenemos un nodo hoja de $T_{\mathcal{M}}$. El conjunto $\mathcal{M}_B^\rho$ asociado a la rama $\rho$ de $T_{\mathcal{M}}$ determinada por este nodo hoja, se obtiene uniendo todos los nodos de $\rho$. Construido $\mathcal{M}_B^\rho$, un modelo para $B$ se obtiene tomando cualquier interpretación $I$ que verifique que $I(p) = b$ para cualquier $pb \in \mathcal{M}_B^\rho$.

Obviamente, al ser TAS-M**3** un método por refutación, para determinar la validez de una fórmula $A$ la entrada de TAS-M**3** será el árbol $T_{\neg A}$ y si la salida es "NO VÁLIDA" se obtendrá un conjunto $\mathcal{M}_{\neg A}^\rho$ que originará una interpretación $I$ tal que $I(\neg A) \in \{\frac{1}{2}, 1\}$, lo que hace que $\neg A$ sea casisatisfacible y por tanto que $A$ no sea VÁLIDA.

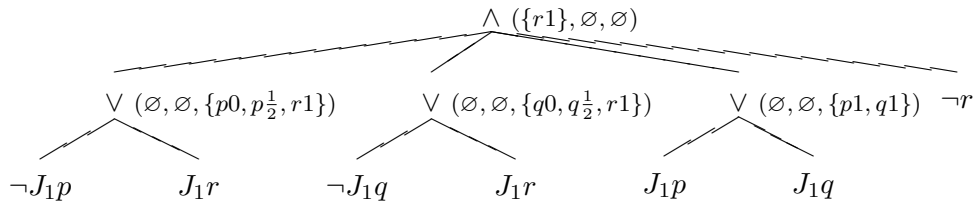
## 4.15. Ejemplos

A continuación demostramos algunos de los axiomas para la lógica trivaluada de Łukasiewicz presentados en (Iturrioz, 1977)

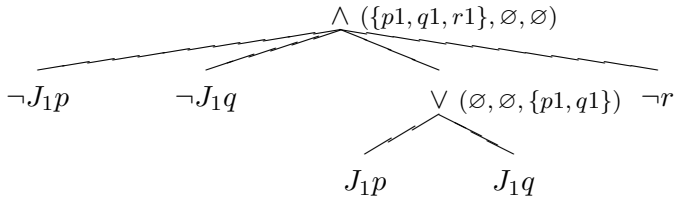
**Ejemplo 4.15** Sea $A = (p \to r) \to ((q \to r) \to ((p \vee q) \to r)$, el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de *Signar*:



Tras el signado y etiquetado de la única tarea que se genera se obtiene el siguiente árbol $T_B$:

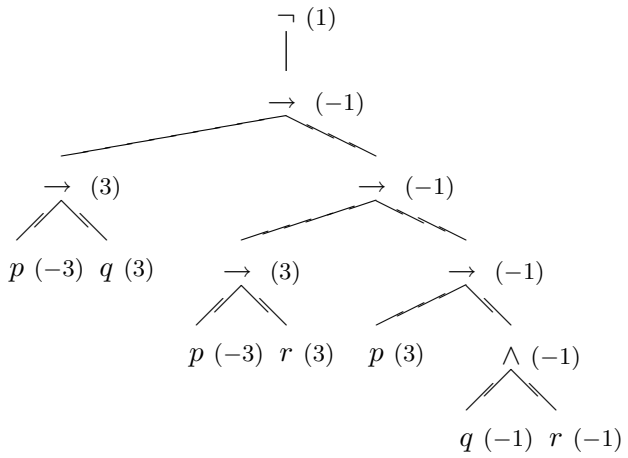



$T_B$ es 0-reducible y realizando las sustituciones $[\![J_1 r/\bot]\!]$ queda el árbol $T_C$ que se muestra a continuación:

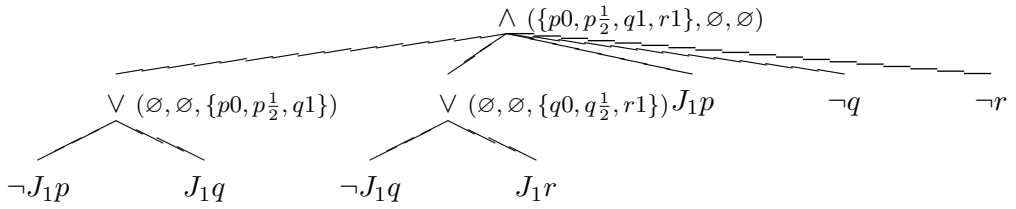



Este árbol es *0-concluyente* ya que $P_{\mathbf{3}} - \overline{\Delta_1(J_1 p \vee J_1 q)} \subseteq \Delta_0(C)$ y $Q_{\mathbf{3}} - \overline{\Delta_1(J_1 p \vee J_1 q)} \subseteq \Delta_0(C)$ por lo tanto se sustituye todo el árbol por $\bot$, que es finalizable y TAS-M**3** termina con salida: "VÁLIDA".

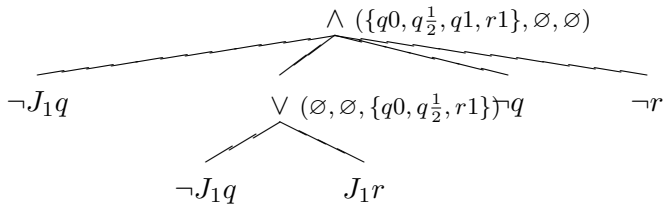
**Ejemplo 4.16** Sea $A = (p \to q) \to ((p \to r) \to (p \to (q \wedge r)))$, el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de signado:




Tras el signado y etiquetado de la única tarea generada, se obtiene el siguiente árbol $T_B$:

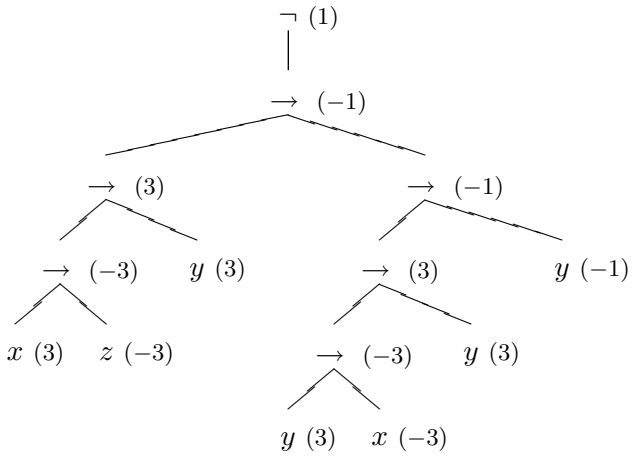



$T_B$ es *completamente 0-reducible* ya que $\{p0, p\frac{1}{2}\} \subseteq \Delta_0(B)$ y realizando las sustituciones $[p/\top]$ (el elemento $p1$ se guarda para un posible contramodelo) y queda el árbol $T_C$ que se muestra a continuación:

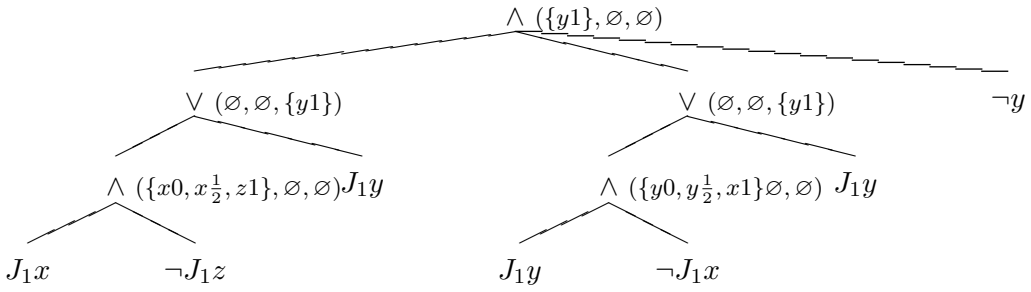



$T_C$ es *0-concluyente*, ya que $Q_{\mathbf{3}} \subseteq \Delta_0(C)$, por lo tanto se sustituye el árbol por $\bot$, que es finalizable y TAS-M**3** termina con salida: "VÁLIDA".

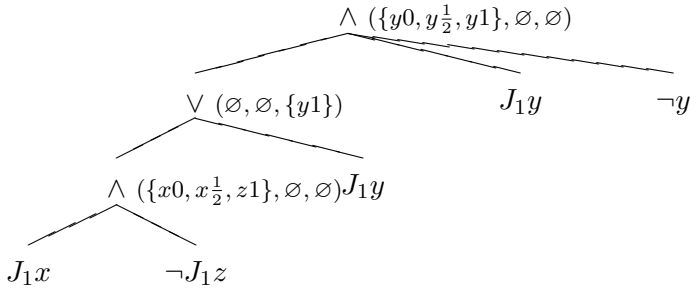
**Ejemplo 4.17** Sea $A = ((x \to z) \to y) \to (((y \to x) \to y) \to y)$, el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de signado:




En la única tarea generada, tras el signado y etiquetado se obtiene el siguiente árbol $T_B$:



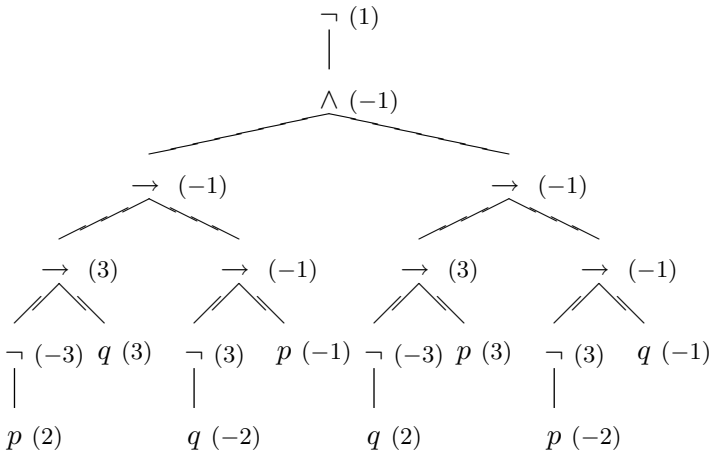

$T_B$ es un árbol podable, ya que el segundo subárbol $T_{(J_1y \wedge \neg J_1x) \vee J_1y}$ verifica que $Y_{\mathbf{3}} - \Delta_0(J_1y \wedge \neg J_1x) = Y_{\mathbf{3}} - \{y0, y\frac{1}{2}, x1\} = \{y1\} \subseteq \Delta_1(B_2)$, por tanto se "poda" el subárbol $T_{J_1y \wedge \neg J_1x}$, obteniéndose el árbol $T_C$ que se muestra a continuación:



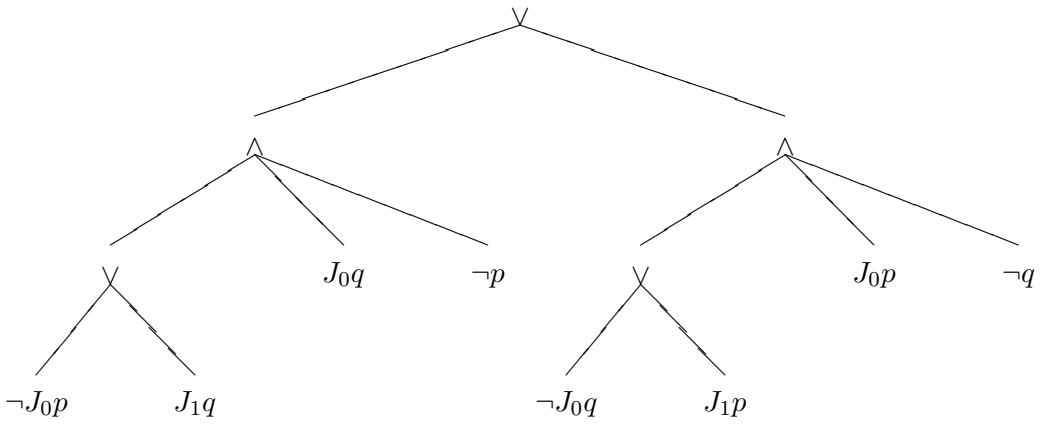

El árbol $T_C$ es *0-concluyente*, ya que $Q_{\mathbf{3}} \subseteq \Delta_0(C)$, por lo tanto se sustituye el árbol entero por $\perp$ que es finalizable y TAS-M**3** termina con salida: "VÁLIDA".

Los siguientes ejemplos están tomados de (Pelletier, 1986), en este artículo se proponen distintas fórmulas válidas para probar demostradores, algunas de estas fórmulas válidas en la Lógica Clásica Proposicional, no lo son en M**3**, éste es el caso del ejemplo siguiente:

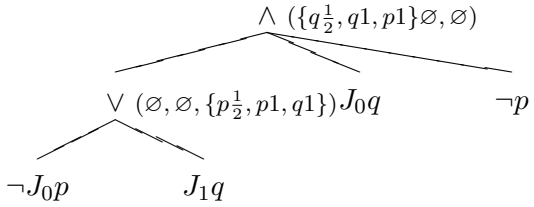
**Ejemplo 4.18** En el problema número 4 de (Pelletier, 1986) se propone: $A = ((\neg p \to q) \to ((\neg q \to p)) \wedge ((\neg q \to p) \to (\neg p \to q))$, el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de *Signar*:




Tras el signado se obtiene el siguiente árbol $T_B$:



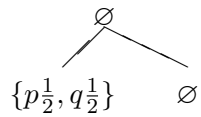

Este árbol tiene una raíz $\vee$, por lo tanto se generan dos tareas, una para cada subárbol hijo de esta raíz. La traza de ejecución para la primera tarea es la siguiente:




El símbolo $p$ es de tipo $\frac{1}{2}$ en el árbol $T_{B_1}$, el proceso $\mathcal{S}$ sustituye los literales que tienen a $p$ como sufijo por $\top$, para la generación de un contramodelo almacenamos en el hijo izquierdo del árbol de modelos el elemento $p\frac{1}{2}$. Al eliminar las constantes en el proceso *Actualizar* se obtiene el árbol:

$$J_0q\ (\{q0\}, \varnothing, \{q\tfrac{1}{2}, q1\})$$

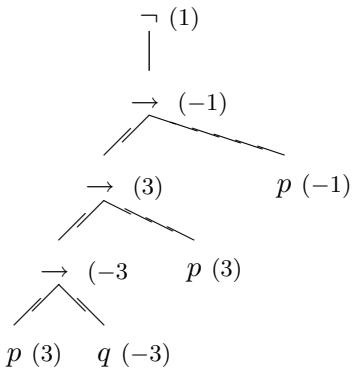
Este árbol es *finalizable* con $\Delta_1 \neq \varnothing$ y da como salida: "VÁLIDA". Para el contramodelo se guarda el primer elemento generado en $\Delta_1$, es decir $q\frac{1}{2}$, por lo que el árbol de modelos asociado a la ejecución de TAS-M**3** queda como:



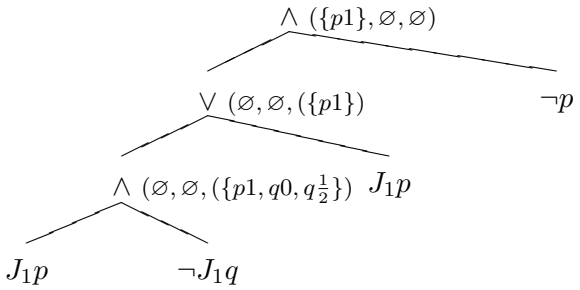
El hijo izquierdo de la raíz del árbol de modelos determina la rama respecto de la que se calcula el modelo, realizando su unión queda $\mathcal{M}^{\rho}_{\neg A} = \{p\frac{1}{2}, q\frac{1}{2}\}$, así cualquier interpretación $I$ que asigne a $p$ y a $q$ el valor $\frac{1}{2}$ es un contramodelo para $A$.

Sin embargo, el siguiente ejemplo, muestra una fórmula válida de la Lógica Clásica Proposicional que sigue siéndolo en M**3**.

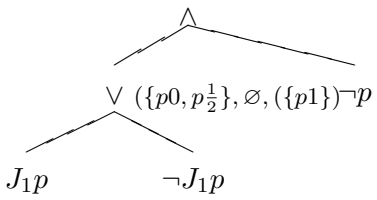
**Ejemplo 4.19** En el problema número 8 de (Pelletier, 1986) se tiene la fórmula $A = ((p \to q) \to p) \to p$, el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de *Signar*:



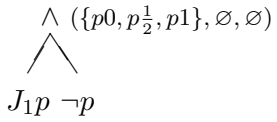

Tras el signado y etiquetado de la única tarea, se obtiene el siguiente árbol $T_B$:




El símbolo $q$ es de tipo 0 en $T_B$, por lo tanto $\mathcal{S}$ sustituye $\neg J_1 q$ por $\top$ (apuntando $q0$ para la generación de un posible contramodelo) y al eliminar la constante el proceso *Actualizar* comienza el etiquetado obteniéndose el árbol $T_C$:



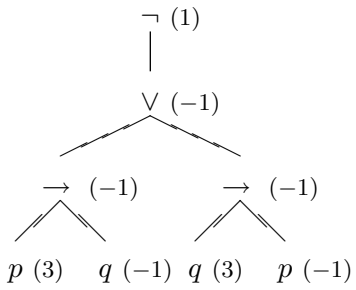

La etiqueta del hijo izquierdo de la raíz de $T_C$, verifica que $\Delta_0(J_1p \vee J_1) \cup \Delta_1(J_1p \vee J_1P) = P_{\mathbf{3}}$, siendo los dos no vacíos, por lo tanto tenemos un árbol *p-simple* y se sustituye $T_{J_1p \vee J_1p}$ por $J_1p$, quedando el árbol $T_D$:



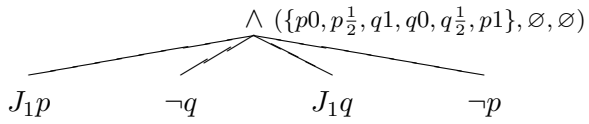

El árbol $T_D$ es *0-concluyente* ya que $P_{\mathbf{3}} \subseteq \Delta_0(D)$ y por tanto el proceso *Actualizar* lo sustituye por $\perp$ que es finalizable y TAS-M**3** termina con salida: "VÁLIDA".

El ejemplo siguiente, muestra cómo puede comprobarse la validez de una fórmula en un sólo recorrido del árbol sin más que *Etiquetar*.

**Ejemplo 4.20** En el problema número 16 de (Pelletier, 1986) se propone la fórmula $A = (p \rightarrow q) \vee (q \rightarrow p)$, el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de *Signar*:




Tras el signado y etiquetado se obtiene el siguiente árbol $T_B$:

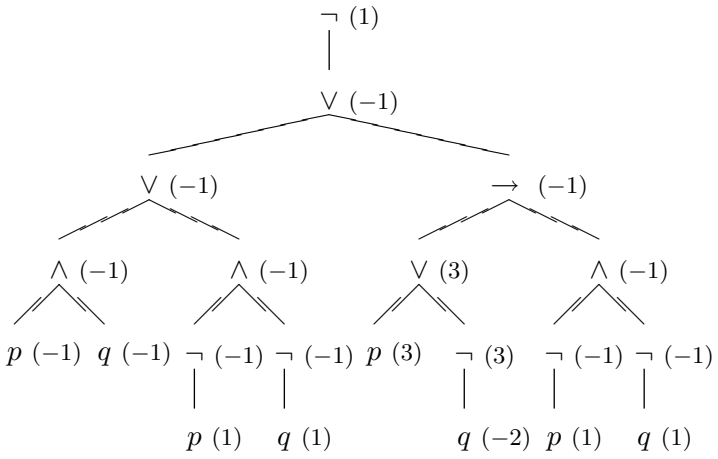



El árbol $T_B$ es *0-concluyente* ya que $P_{\mathbf{3}} \subseteq \Delta_0(B)$ y por tanto el proceso *Actualizar* lo sustituye por el árbol $\perp$ y TAS-M**3** termina con la salida: "VÁLIDA".

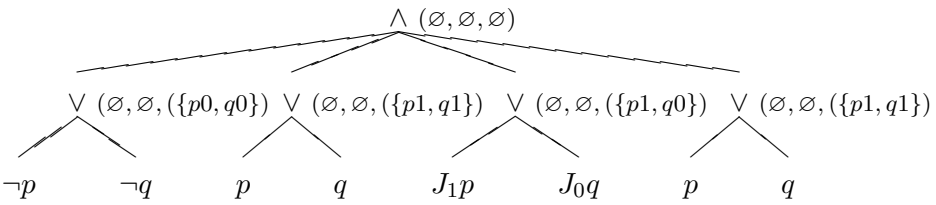
**Ejemplo 4.21** Consideramos ahora la fórmula

$$A = ((p \wedge q) \vee (\neg p \wedge \neg q)) \vee ((p \vee \neg q) \rightarrow (\neg p \wedge \neg q))$$

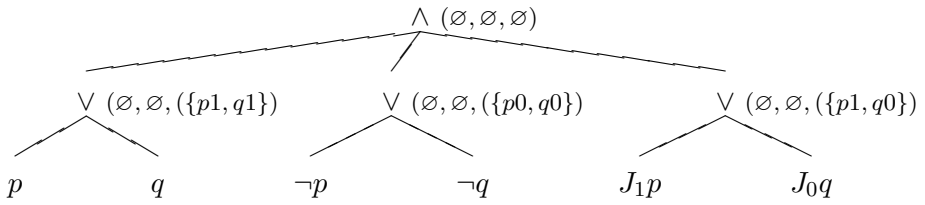
el árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de *Signar*:



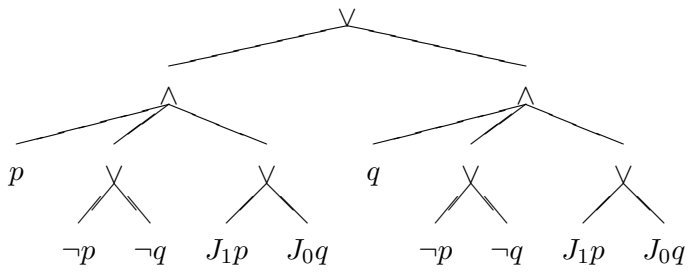

Tras el signado se genera una sola tarea con todo el árbol y tras el etiquetado se obtiene el siguiente árbol $T_B$:



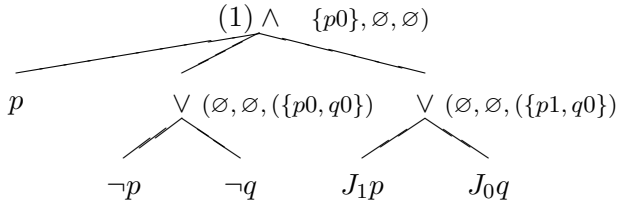

$T_B$ es *sintetizable* respecto al literal trivaluado $p$, aplicando *Sintetizar* y eliminando las constantes aparecidas mediante *Actualizar*, se obtiene el árbol $T_C$:




$T_C$ es la salida del proceso $\mathcal{F}$, al no ser *depurable*, se le aplica el proceso ($\wedge$-$\vee$) y se tiene el árbol $T_D$:



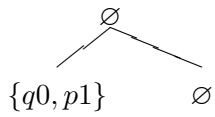

A partir del árbol $T_D$ se generan dos tareas, ya que su raíz es $\vee$, con dos hijos. La primera de ellas, corresponde al subárbol izquierdo $T_E$: par




El símbolo $q$ es de tipo 0 en $T_E$, por lo tanto el proceso $\mathcal{S}$ sustituye todos los literales en los que aparece por $\top$ (guardando $q0$ para el contramodelo y queda el árbol:

$$p\ (\{p0\}, \{p\tfrac{1}{2}\}, \{p1\})$$

Este árbol es finalizable con $\Delta_1(p) = \{p1\} \neq \varnothing$ por tanto la salida es "NO VÁLIDA" y el árbol de modelos es:

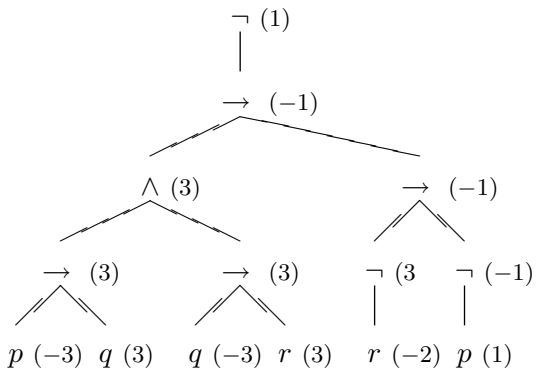



Uniendo los conjuntos de la rama determinada por la hoja izquierda se obtiene el conjunto $\mathcal{M}^{\rho_1}_{\neg A} = \{p1, q1\}$ y por lo tanto podemos afirmar que cualquier interpretación $I$ que verifique que $I(p) = 1$ y $I(q) = 0$ es un contramodelo para $A$.

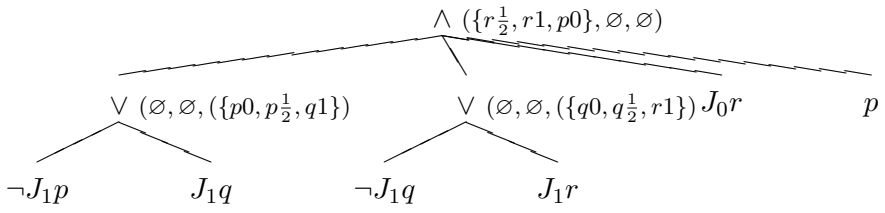
**Ejemplo 4.22** Veamos ahora que la fórmula

$$A = ((p \to q) \wedge (q \to r)) \to (\neg r \to \neg p)$$

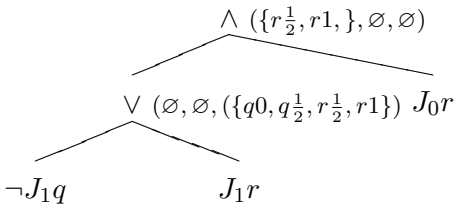
que es válida en la Lógica Clásica Proposicional no lo es en M**3**. El árbol sintáctico de $\neg A$ se muestra a continuación, con las marcas de *Signar*:




Tras el signado se genera una sola tarea con todo el árbol y tras el etiquetado se obtiene el siguiente árbol $T_B$:



El símbolo $p$ es de tipo $\frac{1}{2}$ en $T_B$ por lo tanto el proceso $\mathcal{S}$ sustituye todos los literales que lo tienen como sufijo por $\top$ (y se guarda $p\frac{1}{2}$ para el posible contramodelo). Tras eliminar las constantes introducidas mediante el proceso *Actualizar*, se obtiene el árbol $T_C$:




El símbolo $q$ es de tipo 0 en $T_C$ por lo tanto el proceso $\mathcal{S}$ sustituye todos los literales que lo tienen como sufijo por $\top$ (y se guarda $q0$ para el posible contramodelo). Tras eliminar las constantes introducidas mediante el proceso *Actualizar*, se obtiene el árbol:

$$J_0r \ (\{r\tfrac{1}{2}, r1\}, \varnothing, \{r0\})$$

Este árbol es finalizable, ya que $\Delta_1(J_0r) = \{r0\} \neq \varnothing$ por tanto se apunta $r0$ para el contramodelo y se da la salida "NO VÁLIDA" y el árbol de modelos es:

$$\{p\frac{1}{2}, q0, r0\}$$

El conjunto del árbol hoja $\mathcal{M}_{\neg A} = \{p\frac{1}{2}, q0, r0\}$ determina un contramodelo para $A$, es decir, cualquier interpretación $I$ que verifique que $I(p) = \frac{1}{2}$ y $I(q) = I(r) = 0$ es un contramodelo para $A$.

# Apéndice A

# Resumen de TAS-D con la mejora de *Depurar*

Mostramos en este apéndice un resumen de TAS-D, el demostrador para la Lógica Clásica Proposicional introducido en la tesis de D. Francisco Sanz bajo la dirección de Inmaculada P. de Guzmán, al que el grupo de investigación GIMAC ha ido añadiendo algunas mejoras y extendiendo a la Lógica de Predicados de Primer Orden y a lógicas modales y temporales.

El núcleo principal del presente trabajo es precisamente la extensión de TAS-D a lógicas multivaluadas, concretamente a la lógica trivaluada completa M**3**. Llamamos a este demostrador TAS-M**3**. La razón de incluir en él un resumen del demostrador TAS-D, es múltiple: por una parte, es sensiblemente más fácil comprender el demostrador TAS-M**3** a partir del demostrador TAS-D y por otra parte, nos vemos en cierta medida obligados, porque presentamos aquí una mejora inédita para este demostrador, el proceso *Depurar* que, cómo veremos, es una mejora sustancial. Por esta última razón, incluimos las demostraciones de los resultados relativos al proceso *Depurar*, ya que éstas no han sido publicadas.

## A.1. El proceso *Signar* y los conjuntos $\Delta_0$ y $\Delta_1$

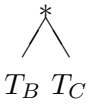
**Notación:**

En lo que sigue usaremos $L$ para denotar al lenguaje de la Lógica Proposicional Clásica, $Q$ para el conjunto de los símbolos proposicionales y $T_A$ para el árbol sintáctico generalizado de una fórmula bien formada (fbf) $A$.

**Definición A.1** El *árbol sintáctico* de una fbf $A$ de L es el árbol binario, recursivamente definido como sigue:

1. Si $A \in Q \cup \{\bot, \top\}$, entonces $T_A$ es $A$.

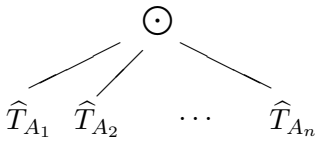
2. Si $A = \neg B$, entonces $T_A$ es 


3. Si $A = B * C$ con $* \in \{\vee, \wedge, \rightarrow\}$, entonces $T_A$ es 


**Definición A.2** Decimos que una fbf $A$ está en *forma normal negativa* (fnn) si satisface las siguientes condiciones:

- En $A$ no interviene $\rightarrow$ ni $\leftrightarrow$.
- El ámbito de las negaciones en $A$ se reduce a los símbolos proposicionales.

**Definición A.3** Sea $A$ una fnn, su *árbol sintáctico generalizado*, denotado por $\widehat{T}_A$, se define recursivamente como sigue:

1. Si $A = \bigodot_{i=1}^{n} A_i$, donde $\bigodot$ es $\bigwedge$ o $\bigvee$, entonces $\widehat{T}_A$ es 


2. $\widehat{T}_A = A$ en otro caso.

Por abuso del lenguaje, hablaremos sólo de árboles sintácticos y utilizaremos únicamente la representación $T_A$, siendo el contexto el que determinará si corresponden a árboles sintácticos generalizados o no.

La primera transformación que TAS-D realiza sobre el árbol sintáctico de una fbf $A$ es *Signar*. El objetivo de *Signar* es transformar el árbol de la fórmula de entrada $T_A$ en el árbol de una forma normal negativa equivalente a $A$.

**Definición A.4** Dada una fbf $A$ y $\kappa \in \{+, -\}$, se define recursivamente $A(\kappa)$ como sigue:

$$
\begin{array}{ll}
p\,(+) = p & p\,(-) = \neg p \\
(\neg A)\,(+) = A\,(-) & (\neg A)\,(-) = A\,(+) \\
(A \vee B)\,(+) = A\,(+) \vee B\,(+) & (A \vee B)\,(-) = A\,(-) \wedge B\,(-) \\
(A \wedge B)\,(+) = A\,(+) \wedge B\,(+) & (A \wedge B)\,(-) = A\,(-) \vee B\,(-) \\
(A \rightarrow B)\,(+) = A\,(-) \vee B\,(+) & (A \rightarrow B)\,(-) = A\,(+) \wedge B\,(-)
\end{array}
$$

Dada una fbf $A$, por *Signar* un árbol $T_A$ entenderemos "obtener $T_{A\,(+)}$". Es obvio que si $A$ es una fbb entonces $A\,(+)$ es una fórmula en forma normal negativa equivalente a $A$.

La idea de usar la información contenida en las interpretaciones parciales, exhaustivamente utilizada en el método de Quine, se utiliza en TAS-D, utilizando solamente interpretaciones parciales unitarias, pero como veremos con una potencia sorprendente. Esta información se usa ampliamente en TAS-D a través de los conjuntos $\Delta_0$ y $\Delta_1$, que constituyen las herramientas básicas del método. Estos conjuntos se definen a continuación:

**Definición A.5** Dada una fnn $A$, los conjuntos $\Delta_0(A)$ y $\Delta_1(A)$ se definen recursivamente como sigue:

$$\Delta_0(p) = \{p\,0\}; \qquad \Delta_1(p) = \{p\,1\}; \qquad \Delta_0(\neg p) = \{p\,1\}; \qquad \Delta_1(\neg p) = \{p\,0\}$$

$$\Delta_0\left(\bigvee_{i=1}^{n} A_i\right) = \bigcap_{i=1}^{n} \Delta_0(A_i); \qquad \Delta_1\left(\bigvee_{i=1}^{n} A_i\right) = \bigcup_{i=1}^{n} \Delta_1(A_i)$$

$$\Delta_0\left(\bigwedge_{i=1}^{n} A_i\right) = \bigcup_{i=1}^{n} \Delta_0(A_i); \qquad \Delta_1\left(\bigwedge_{i=1}^{n} A_i\right) = \bigcap_{i=1}^{n} \Delta_1(A_i)$$

Es obvio que $\Delta_1(A)$ puede verse como el conjunto de los modelos unitarios de $A$ y $\Delta_0(A)$ como el conjunto de los modelos unitarios de $\neg A$.

Si $A$ es una fnn, *Etiquetar* el árbol $T_A$ significa etiquetar cada nodo $N$ de $T_A$ con el par ordenado $\big(\Delta_0(B), \Delta_1(B)\big)$ donde $B$ es la subfórmula de $A$ tal que $N$ es la raíz de $T_B$. Dada una fbf $A$, la entrada de TAS-D es $T_{\neg A}$, el árbol sintáctico de $\neg A$. Tras *Signar*, se aplican los procesos $\mathcal{F}$ e $(\wedge\text{-}\vee)$-*par* según se muestra en el diagrama de la Figura A.1, en el que las salidas son o bien VÁLIDA o bien NO-VÁLIDA y, en este último caso, un modelo para $\neg A$ (contramodelo).

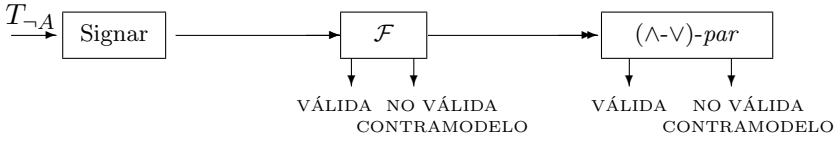



Figura A.1: El método TAS-D.

Para la mayoría de las fórmulas, si TAS-D no termina tras la ejecución del proceso $\mathcal{F}$, el tamaño de la fórmula de entrada al proceso$(\wedge\text{-}\vee)$-*par* ha decrecido drásticamente. El interés de esta estrategia se basa en el hecho de que podemos reducir el tamaño del problema antes de aplicar la transformación que soporta el peso de la complejidad ($(\wedge\text{-}\vee)$-*par* en nuestro caso).

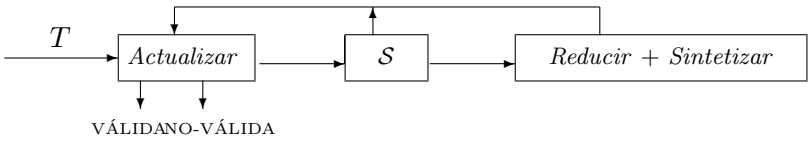



Figura A.2: El proceso $\mathcal{F}$.

## A.2. El proceso $\mathcal{F}$

La entrada de este proceso es un árbol sintáctico generalizado $T_C$. En esta transformación de árboles detectamos cuándo las etiquetas $(\Delta_0, \Delta_1)$ proporcionan, bien información completa sobre la insatisfacibilidad de $C$ o información útil para reducir su tamaño antes de distribuir.

El proceso $\mathcal{F}$ está compuesto a su vez por los procesos *Actualizar*, $\mathcal{S}$, *Reducir*, y *Sintetizar*. Estos procesos se realizan en secuencia, tantas veces como sea posible, teniendo en cuenta que cuando uno de estos proceso actúa se vuelve al proceso inicial de $\mathcal{F}$, es decir al proceso *Actualizar*, donde se eliminan las constantes aparecidas e incluso puede detectarse el fin del método. Esta secuencia se muestra en la Figura A.2.

Este proceso puede contemplarse como una secuencia de filtros realizados con la información de los conjuntos $\Delta$, de aquí el nombre de $\mathcal{F}$.

A continuación se muestra cómo se usa la información contenida en las etiquetas $(\Delta_0, \Delta_1)$:

1. El proceso *Actualizar* hace uso de los conceptos de *Simplificar* y *Finalizar*. Estos conceptos se detallan a continuación:

   *a*) El proceso *Simplificar* permite detectar que una subfórmula de una fórmula $B$, o incluso la fórmula completa son equivalentes a $\top$, $\bot$ o a un literal. Esto puede dar lugar a que el método termine o a reducir el tamaño de la fórmula. La corrección de este proceso la asegura el siguiente resultado:

**Teorema A.1** *Sea $p$ un símbolo proposicional que ocurre en una fbf $A$ entonces:*

a*) Si $\{p0, p1\} \in \Delta_0(A)$ si y sólo si $A \equiv \bot$.*

b*) Si $\{p0, p1\} \in \Delta_1(A)$ si y sólo si $A \equiv \top$.*

c*) Si $p0 \in \Delta_0(A)$ y $p1 \in \Delta_1(A)$ si y sólo si $A \equiv p$.*

d) *Si* $p1 \in \Delta_0(A)$ *y* $p0 \in \Delta_1(A)$ *si y sólo si* $A \equiv \neg p$.

(b) Si el árbol es $\top$ o $\bot$ o bien si el conjunto $\Delta_1$ de la raíz es distinto del vacío, decimos que el árbol es *finalizable* y por *Finalizar* entendemos terminar el método TAS-D con la salida "VÁLIDA" en el caso de que el árbol sea $\bot$ o la salida "NO VÁLIDA" y un contramodelo en el caso de que el árbol sea $\top$ o bien el conjunto $\Delta_1$ de la raíz sea distinto del vacío.

El proceso *Actualizar* interviene de dos formas diferentes:

(i) La primera vez que interviene el proceso *Actualizar*, etiqueta el árbol a la vez que comprueba si es simplificable o finalizable, en cuyo caso se simplifica o se finaliza respectivamente.

(ii) Cuando *Actualizar* se aplica después de que hayan realizado alguna acción los procesos *Simplificar*, $\mathcal{S}$, *Reducir* o *Sintetizar*, su acción consiste en: eliminar las constantes incluidas por estos procesos, calcular las etiquetas de los nodos ascendientes de los nodos modificados y comprobar la finalizabilidad. Una vez más, si la respuesta es afirmativa, se termina el método.

De esta forma el proceso *Actualizar* puede forzar al método a terminar o a reducir el tamaño de la fórmula bajo análisis.

2. El proceso $\mathcal{S}$ extiende a cualquier fnn la regla del literal puro ampliamente utilizada en el método de resolución. Para ello distinguimos entre las ocurrencias positivas y negativas de un símbolo proposicional. Llamamos una ocurrencia negativa de un símbolo $p$ a aquella ocurrencia en la que aparece en un literal precedido de una negación, y ocurrencia positiva en caso contrario. Decimos que un determinado símbolo $p$ es positivo en $A$ si sólo tiene ocurrencias positivas, y negativo si sólo tiene ocurrencias negativas.

   El proceso $\mathcal{S}$ sustituye todos los símbolos positivos por $\top$ y los negativos por $\bot$. Este proceso no mantiene el significado, pero sí la satisfacibilidad.

3. El proceso *Reducir* usa de nuevo la información contenida en los conjuntos $(\Delta_0, \Delta_1)$ para reducir el tamaño del árbol antes de distribuir. Concretamente, el proceso sustituye el árbol sintáctico de una fnn $A$ por el de otra que es equisatisfacible con $A$ y en el que los símbolos en $\Delta_0(A)$ o $\Delta_1(A)$ ocurren a lo sumo una vez. Tras *Reducir* un árbol no simplificable, se puede obtener uno que sí lo sea y por tanto evitar distribuciones o incluso terminar el método TAS-D. La corrección de este proceso se asegura por el siguiente resultado:

**Teorema A.2**

a) *Si* $p0 \in \Delta_0(A)$ *si y sólo si* $A \equiv p \wedge A[p/\top]$. *Por tanto,* $A$ *y* $A[p/\top]$ *son equisatisfacibles.*

b) *Si* $p1 \in \Delta_0(A)$ *si y sólo si* $A \equiv \neg p \wedge A[p/\bot]$. *Por tanto,* $A$ *y* $A[p/\bot]$ *son equisatisfacibles.*

c) *Si* $p0 \in \Delta_1(A)$ *si y sólo si* $A \equiv \neg p \vee A[p/\top]$.

d) *Si* $p1 \in \Delta_1(A)$ *si y sólo si* $A \equiv p \vee A[p/\bot]$.

e) *Si* $A$ *es finalizable y* $\Delta_1(A) \neq \varnothing$ *entonces* $A$ *es satisfacible.*

Los dos primeros puntos del teorema anterior nos permiten sustituir un árbol $T_A$ por otro $T_B$ equisatisfacible con él en el que se han sustituido todas las apariciones del símbolo $p$ por $\top$ o por $\bot$ según sea $p0$ el elemento que está en la etiqueta $\Delta_0$ de la raíz del árbol o sea $p1$. A esta clase especial de reducción que no conserva la equivalencia lógica, pero sí la satisfacibilidad y que elimina completamente un determinado símbolo proposicional se le llama *completa 0-reducción*.

Obsérvese que para poder aplicar este proceso el conjunto $\Delta_0$ asociado a la raíz del árbol completo en estudio debe ser distinto del vacío; por supuesto esto no puede generalizarse a subárboles.

4. El proceso *Sintetizar* no es más que la restricción del proceso *Reducir* cuando este proceso podría aplicarse si elmináseмos algunos hijos de un determinado nodo. La forma de actuar de este proceso es la de realizar una reducción agrupando por un lado a los hijos del nodo a los que se le podría aplicar el proceso *Reducir* y aplicarlo sólo a este grupo, y por otro lado los demás hijos.

## A.3. El proceso ($\wedge$-$\vee$)-*par*

Sobre el proceso ($\wedge$-$\vee$)-*par* recae el peso de la complejidad exponencial. ($\wedge$-$\vee$)-*par* es el encargado de realizar las distribuciones, pero estas distribuciones se realizan en principio sólo una vez y se obtienen una serie de tareas paralelizables. Los métodos de reducción vistos en el proceso $\mathcal{F}$ se aplican a cada una de las tareas resultantes.

La distribución se realiza únicamente sobre el hijo más a la izquierda de la raíz $\wedge$ mediante la transformación ($\wedge$-$\vee$) que es la traducción en términos de árboles sintácticos de la ley distributiva de $\wedge$ sobre $\vee$ para el hijo más a la izquierda. Así el árbol $T_{(B_1 \wedge B_2 \wedge \cdots \wedge B_n) \vee A_2 \vee \cdots \vee A_m}$ proporciona un árbol de raíz $\vee$ con $n$ hijos: $T_{B_1 \wedge A_2 \wedge \cdots \wedge A_m}$, $T_{B_2 \wedge A_2 \wedge \cdots \wedge A_m}$, $\ldots$, $T_{B_n \wedge A_2 \wedge \cdots \wedge A_m}$. Cada uno de estos

árboles constituye una tarea paralelizable a la que se le vuelven a aplicar todos los procesos descritos. Esta filosofía permite por una parte la posible realización en paralelo de las tareas y en cualquier caso la reducción de las fórmulas asociadas a cada subtarea con las técnicas vistas. En el momento en el que una tarea detecta que la fórmula es "NO VÁLIDA" se para todo el proceso y se da la salida general "NO VÁLIDA". Si todas las tareas responden "VÁLIDA" entonces la salida general es "VÁLIDA".

## A.4. Una mejora de TAS-D: El proceso *Depurar*

Supongamos que un árbol $T_A$ es la salida del proceso $\mathcal{F}$ y tiene $\wedge$ como raíz. Este árbol será la entrada del proceso ($\wedge$-$\vee$)-*par*, en el que se realizará una distribución (ya que su raíz es $\wedge$), esta distribución se hace únicamente respecto de una rama (a efectos de la explicación del método, supondremos que distribuimos respecto de la rama más izquierda) y se vuelven a aplicar las reducciones que realiza $\mathcal{F}$ a cada uno de las nuevas tareas que se generan. Sin embargo cabe preguntarse si podemos aún extraer más información de los conjuntos $\Delta$ para que esta distribución se realice de forma óptima, es decir, intentando que el número total de distribuciones sea el menor posible.

Concretamente, sea una fnn $A = \bigwedge_{i=1}^{n} A_i$ tal que existe $i_1 \in [n]$, $p \in Q$ y $b \in \{0,1\}$ con $pb \in \Delta_1(A_{i_1})$. Entonces, puesto que $T_A$ es la salida de $\mathcal{F}$, podemos garantizar que:

- La única ocurrencia de $p$ en $T_{A_{i_1}}$ es como hijo literal de su raíz (puesto que el árbol ya ha pasado por el proceso *Reducir*).
- No existe $i_2 \in [n]$, con $i_1 \neq i_2$ tal que $pb \in \Delta_1(A_{i_2})$ (puesto que el árbol ya ha pasado por el proceso *Sintetizar*).

Pero sí podría ocurrir que además de tener $pb \in \Delta_1(A_{i_1})$ se tenga que $p\bar{b} \in \Delta_1(A_{i_2})$ para algún $i_2 \in [n]$. ¿ Podemos utilizar esta información para conseguir el objetivo propuesto? La respuesta a la pregunta anterior es afirmativa. Más aún, además de evitar distribuciones, podemos utilizar esta información para eliminar totalmente el símbolo $p$.

El modo de incorporar la información citada en TAS-D queda reflejado en el siguiente teorema, que es una potente extensión del teorema de Davis y Putnam (**?**) para el estudio de la satisfacibilidad de un conjunto de cláusulas.

**Teorema A.3** *Sean* $A = \bigwedge_{i=1}^{n} A_i$ *una fnn,* $p \in Q$ *un símbolo proposicional y sean* $i_1, i_2 \in [n]$ *tales que* $p1 \in \Delta_1(A_{i_1})$ *y* $p0 \in \Delta_1(A_{i_2})$. *Entonces se tiene que las fbfs*

$$A \quad y \quad (\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot]) \textit{ son equisatisfacibles.}$$

DEMOSTRACIÓN: Por el Teorema A.2 se tiene que $A_{i_1} \equiv p \vee A_{i_1}[p/\bot]$, más aún, las reducciones realizadas por $\mathcal{F}$, nos permiten asegurar que $A_{i_1}$ es de la forma $A_{i_1} = p \vee B_{i_1}$ y que $p$ no ocurre en $B_{i_1}$. Análogamente, $A_{i_2} \equiv \neg p \vee A_{i_2}[p/\top]$ y podemos asegurar que $A_{i_2} = \neg p \vee B_{i_2}$ y $p$ no ocurre en $B_{i_2}$. Por las leyes asociativa y conmutativa y el teorema de equivalencia se tiene que

$$A \equiv (p \vee A_{i_1}[p/\bot]) \wedge (\neg p \vee A_{i_2}[p/\top]) \wedge \bigwedge_{i\in[n]-\{i_1,i_2\}} A_i$$

Podemos demostrar ya la afirmación del teorema.

1. Si $A$ es satisfacible entonces existe una interpretación $I$ tal que

$$I(p \vee A_{i_1}[p/\bot]) = 1, \quad I(\neg p \vee A_{i_2}[p/\top]) = 1 \text{ y } I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = 1$$

Tenemos que probar que $(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot])$ es satisfacible. Distinguimos dos casos:

   (a) $I(p) = 1$. En este caso, puesto que $I(\neg p \vee A_{i_2}[p/\top]) = 1$ se tiene que $I(A_{i_2}[p/\top]) = 1$. Por otra parte, puesto que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = 1$ y $I(p) = 1$ se tiene que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\top]) = 1$. Por lo tanto, tenemos asegurado que $I(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) = 1$

   (b) $I(p) = 0$. En este caso, puesto que $I(p \vee A_{i_1}[p/\bot]) = 1$ se tiene que $I(A_{i_1}[p/\bot]) = 1$. Por otra parte, puesto que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = 1$ y $I(p) = 0$ se tiene que $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\bot]) = 1$. Por lo tanto, tenemos asegurado que $I(\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot]) = 1$

De los dos casos anteriores se concluye que

$$I((\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot])) = 1$$

es decir,

$$(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot]) \quad \text{es satisfacible.}$$

2. Recíprocamente, supongamos que

$$(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot])$$

es satisfacible. Entonces, se tiene que al menos una de las dos fórmulas siguientes es satisfacible:

- $\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]$
- $\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot]$

Supongamos que lo es $\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]$ (el razonamiento para la segunda es análogo) y sea $I$ una interpretación tal que $I(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) = 1$; por lo tanto $I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\top]) = 1$ y $I(A_{i_2}[p/\top]) = 1$. Consideremos la extensión de $I$ definida por

$$I'(q) = \begin{cases} I(q) & \text{si} \quad q \neq p \\ 1 & \text{si} \quad q = p \end{cases}$$

Veamos que $I'(A) = 1$.

- Puesto que $p$ no ocurre en $\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\top]$, se tiene que

$$I'(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i) = I(\bigwedge_{i\in[n]-\{i_1,i_2\}} A_i[p/\top]) = 1$$

- Puesto que $p$ no ocurre en $A_{i_2}[p/\top]$, se tiene que $I'(A_{i_2}) = I(A_{i_2}[p/\top]) = 1$.

- Por último, puesto que $p_1 \in \Delta_1(A_{i_1})$ y $I'(p) = 1$, se tiene que $I'(A_{i_1}) = 1$

Por lo tanto $I'(A) = 1$ y $A$ es satisfacible.

*q.e.d.*

**Definición A.6** Sea $A$ una fnn. El árbol $T_A$ se dice *depurable* si $A = \bigwedge_{i=1}^{n} A_i$ y existe un símbolo proposicional $p \in Q$ y dos índices $i_1, i_2 \in [n]$ tales que $p1 \in \Delta_1(A_{i_1})$ y $p0 \in \Delta_1(A_{i_2})$. En este caso entendemos por *depurar* la sustitución de $T_A$ por el árbol de la fórmula: $(\bigwedge_{i\in[n]-\{i_1\}} A_i[p/\top]) \vee (\bigwedge_{i\in[n]-\{i_2\}} A_i[p/\bot])$.

El flujo del proceso ($\wedge$-$\vee$)-*par* con la mejora de *Depurar* puede observarse en la figura A.3.

La corrección y completitud del TAS-D viene dada por el siguiente resultado cuya demostración puede verse en (**?**)

**Teorema A.4** *Una fbf $A$ es válida si y sólo si TAS-D($T_{\neg A}$) = "VÁLIDA".*

## A.5. TAS-D es un método de construcción de modelos

En el caso de que TAS-D pare con salida "NO VÁLIDA" se ofrece además un modelo para la fórmula $\neg A$ cuyo árbol $T_{\neg A}$ es la entrada del método, es decir, un contramodelo para $A$. La construcción de modelos se realiza de forma natural a través del árbol de modelos que pasamos a definir.

**Definición A.7** Un *árbol de modelos* asociado a la ejecución de TAS-D sobre un árbol $T_B$, es un árbol $T_{\mathcal{M}}$ cuyos nodos son conjuntos de elementos de la forma $pb$ con $p \in Q$ y $b \in \{0, 1\}$ y que es generado durante la ejecución de TAS-D mediante la asociación de sus nodos a los árboles obtenidos en tal ejecución. La generación de $T_{\mathcal{M}}$ se realiza del modo descrito a continuación:

1. Al iniciarse la ejecución de TAS-D, $T_{\mathcal{M}}$ es un árbol cuyo único nodo es el conjunto vacío (asociado al árbol de entrada a TAS-D, $T_B$).

2. Para cada etapa de ejecución de TAS-D:

   - Si en la ejecución de TAS-D (dentro del proceso ($\wedge$-$\vee$)-*par*) a partir de un árbol $T_C$ se generan las tareas $T_{C_1}, T_{C_2}, \ldots, T_{C_n}$, se añaden $n$ hijos en $T_{\mathcal{M}}$ como descendientes del nodo asociado al árbol $T_C$. Estos hijos serán:
     *a*) Todos ellos $\varnothing$ si no se ha aplicado el proceso *Depurar*.
     *b*) En caso contrario, es decir, si se aplica *Depurar*, se tiene que $n = 2$ y los hijos serán:
       - $\{p0\}$ para el hijo asociado a la tarea en la que se ha sustituido el símbolo $p$ por $\bot$.
       - $\{p1\}$ para el hijo asociado a la tarea en la que se ha sustituido el símbolo $p$ por $\top$.

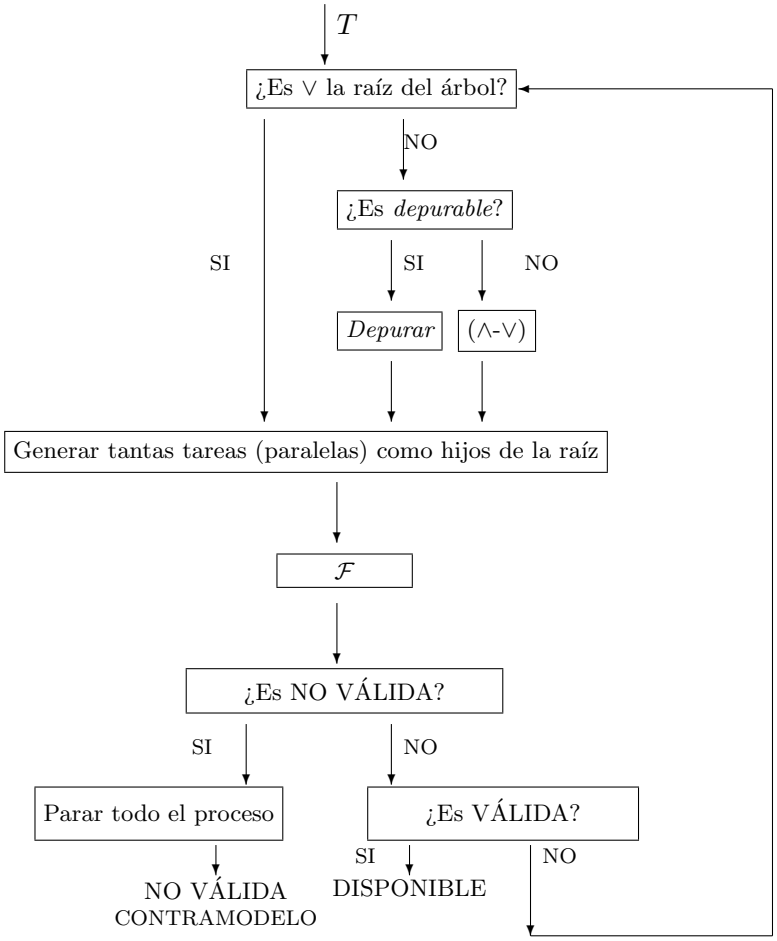



Figura A.3: El proceso (∧-∨)-*par*.

- Si en la ejecución de TAS-D se obtiene un árbol finalizable, $T_F$, con $\Delta_1(F) \neq \varnothing$, se añade a su nodo asociado en $T_{\mathcal{M}}$ el primer elemento en orden lexicográfico de $\Delta_1(F)$.
- El resto de los procesos de TAS-D, son de dos tipos, los que mantienen la equivalencia lógica y los que sólo conservan la satisfacibilidad. Veamos que sucede con cada uno de ellos:
  i) Los procesos que sólo mantienen la satisfacibilidad y no la equivalencia lógica sustituyen todas las apariciones de un símbolo proposicional por las constantes $\top$ o $\bot$ en todo el árbol. Los procesos de este tipo son *completa 0-reducción*, $\mathcal{S}$ y *Depurar*. Puesto que las acciones para el proceso *Depurar* ya han sido explicadas dentro del apartado ($\wedge$-$\vee$)-*par*, en este apartado sólo nos queda considerar los procesos *completa 0-reducción* y $\mathcal{S}$. En estos casos, si $T_G$ es la entrada de uno de estos procesos y $T_H$ la salida, el nodo asociado a $T_G$ pasa a asociarse a $T_H$ y a este nodo se le añade para cada símbolo proposicional $p$ eliminado, el elemento $p0$ si $p$ ha sido sustituido en todas sus ocurrencias por $\bot$ y el elemento $p1$ si $p$ ha sido sustituido por $\top$.
  ii) Si un proceso transforma un árbol $T_D$ en otro $T_E$, manteniendo la equivalencia lógica entre $D$ y $E$ el nodo asociado a $T_D$ pasa a ser asociado a $T_E$ si modificación alguna en su contenido.

Si $T_B$ es el árbol de entrada en el método TAS-D y éste para con salida "NO VÁLIDA" es porque el proceso *Actualizar* ha detectado que el conjunto $\Delta_1$ de la raíz de un árbol completo es distinto del vacío y por tanto que es finalizable. En este caso, asociado a este árbol tenemos un nodo hoja de $T_{\mathcal{M}}$. El conjunto $\mathcal{M}_B^\rho$ asociado a la rama $\rho$ de $T_{\mathcal{M}}$ determinada por este nodo hoja, se obtiene uniendo todos los nodos de $\rho$. Construido $\mathcal{M}_B^\rho$, un modelo para $B$ se obtiene tomando cualquier interpretación $I$ que verifique que $I(p) = b$ para cualquier $pb \in \mathcal{M}_B^\rho$.

Obviamente, como TAS-D es un método por refutación, para determinar la validez de un fórmula $A$ la entrada de TAS-D será el árbol $T_{\neg A}$ y si la salida es "NO VÁLIDA" se obtendrá un conjunto $\mathcal{M}_{\neg A}^\rho$ que originará un modelo para $\neg A$ y por tanto un contramodelo para $A$.

# Bibliografía


Aguilera, G., de Guzmán, I., Galán, J., y Ojeda, M. (1994a). TAS-D$^{++}$ vs tablas semánticas. En *GULP-PRODE '94 Joint Conference on Declarative Programming*, Valencia.

Aguilera, G., de Guzmán, I., Galán, J., y Ojeda, M. (1994b). Using TAS-D$^{++}$ for inferences. En *Proceedings of Iberamia'94*, pp. 44–59, Caracas (Venezuela).

Aguilera, G., de Guzmán, I., y Ojeda, M. (1994c). TAS-D$^{++}$ syntactic trees transformations for automated theorem proving. *Lect. Notes in Artif. Intelligence 838*, pp. 198–216.

Aguilera, G. y de Guzmán, I. P. (1993). *Lógica para la Computación (vol 1: Lógica Proposicional).* Editorial Librería Ágora.

Aguilera, G., de Guzmán, I. P., Galán, J. L., y Ojeda, M. (1995a). A new general approach to ATPs. En *Proceedings of KI-15, Workshop on Computational Propositional Logic*, pp. 4–10, Bielefeld (Germany).

Aguilera, G., de Guzmán, I. P., y Ojeda, M. (1993). Un algoritmo eficiente y paralelo para la transformación a forma normal conjuntiva. En *Programación Declarativa*, pp. 275–290, Blanes (Spain).

Aguilera, G., de Guzmán, I. P., y Ojeda, M. (1994d). Automated model building via syntactic trees transformations. En *CADE12. Workshop 2B: Automated model building*, pp. 4–10, Nancy (France).

Aguilera, G., de Guzmán, I. P., y Ojeda, M. (1995b). Increasing the efficiency of automated theorem proving. *Journal of Applied Non-Classical Logics*, 5(1):9–29.

Anderson, A. y N.D.Belnap (1975). *Entailment. Vol 1.* Princenton University Press.

Andrews, P. (1981). Theorem proving via general matings. *Journal of the ACM.*, 28(2):193–214.

Avron, A. (1991). Natural 3-valued logics. Characterization and proof theory. *The Journal of Symbolic Logic*, 56(1).

Baaz, M. y Zach, R. (1992). Note on calculi for three-valued logic for logic programming. *Bulletin of the EATCS*, pp. 157–164.

Beth, E. (1959). *The Foundations of Mathematics*. North-Holland.

Beth, E. (1967). *Aspects of modern logic*. Reidel Publishing Company.

Bibel, W. (1982). *Predicate Logic as a Programming Language*. Vieweg Verlag.

Birkhoff, G. y von Neumann, J. (1936). The logic of quantum mechanics. *Annals of Mathematics*, 37:823–843.

Bledsoe, W. y Loveland, D. (1984). *Automated Theorem Proving after 25 years*. Contemporary Mathematics.

Blikle, A. (1991). Three-valued predicate for software specification and validation. *Fundamenta Informaticae*, XIV:387–410.

Bochvar, D. (1939). On a 3-valued logical calculus and its applications to analysis of contradictions. *Matématičéskij sbornik*, 4.

Bolc, L. y Borowick, P. (1992). *Many-valued Logics. Theoretical Foundations*. Springer-Verlag.

Byrd, M. (1979). A formal interpretation of Łukasiewicz logics. *Notre Dame Journal of Formal Logic*, XX(4).

Carnielli, W. (1985). An algorithm for axiomatizing and theorem proving in finite many-valued propositional logics. *Journal of Symbolic Logic*, pp. 363–368.

Carnielli, W. (1987). Systematization of finite many-valued logics through the method of tableaux. *Journal of Symbolic Logic*, 52(2):473–493.

Church, A. (1936). A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41 y 101–102 (una corrección).

Coferra, R. y Zabel, N. (1991). *Extending Resolution for Model Construction*. LNCS 478.

Colmerauer, A. y Pique, J. (1981). *About natural logic*. Advances in data base theory, vol. 1 (H. Gallier et al. editors) Plenum press, New York.

de Bessonet, C. (1991). *A Many-Valued Approach to Deduction and Reasoning for Artificial Intelligence*. Kluwer Academic Publishers.

de Guzmán, I. P. y Enciso, M. (1995). A new and complete theorem prover for temporal logic. En *Proceedings of the IJCAI Workshop on Executable Temporal Logics*, Montreal (Canada).

de Guzmán, I. P., Enciso, M., y Rossi, C. (1995). Just one approach for several temporal logics in computing: the topological semantics. En *Proceedings of the TIME'95 Workshop, FLAIRS*, Melbourne, Florida (USA).

Delahaye, J. y Thibau, V. (1991). Programming in three-valued logic. *Theoretical Computer Science*, 78:189–216.

Dunn, J. (1986). *Relevance logic and entailment. Handbook of Philosophical Logic Vol. III: Alternatives in Classical Logic.* D. Gabbay and F. Guenthner, editors. Reidel, Dordrecht.

E. Trillas, C. A. y Terricabras, J. (1995). *Introducción a la lógica borrosa.* Ariel Barcelona.

Enciso, M. (1995). *Lógica temporal y demostración automática de teoremas. Eficiencia y paralelismo.* PhD thesis, Universidad de Málaga, España.

et al, A. C. (1973). *Un Systeme de Comunication Homme-Machine.* Groupe de Recherche en Intelligence Artificielle, Université d'Aix-Marseille.

Fitting, M. (1983). *Proof Methods for Modal and Intuitionistic Logics.* D. Reidel Publishing Company.

Fitting, M. (1986). Partial models and logic programming. *Theoretical Computer Science*, 48:229–255.

Fitting, M. (1990a). Bilattices in logic programming. *20th International Symposium on Multiple-Valued*, pp. 238–247.

Fitting, M. (1990b). *First Order Logic and Automated Theorem Proving.* Springer Verlag.

Fitting, M. (1994). Tableaux for logic programming. *Journal of Automated Reasoning*, 13(2):175–188.

Gabbay, D. M. (1993). *LDS. Labelled Deductive Systems.* Comunicación personal.

Gallier, J. (1987). *Logic for Computer Science: Foundations for Automatic Theorem Proving.* John Wiley & sons.

Gentzen, G. (1935). Investigations into logical deduction. *M. Szabo, ed. The Collected Papers of Gerhard Gentzen. North Holland.*

Goddar, L. y Routley, R. (1973). *The Logic of Significance and Context*. Academic Press.

H. Barringer, J. C. y Jones, B. (1984). A logic covering undefinedness in program proofs. *Acta Informatica*, 21:251–269.

Haak, S. (1978). *Philosophy of Logics*. Cambridge University Press.

Hähnle, R. (1993). *Automated Deduction in Multiple-valued Logics*. Oxford University Press.

Hähnle, R. y Schmitt, P. (1994). The liberalized $\delta$-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2):211–221.

Halldén, S. (1949). The logic of nonsense. *Uppsala Universiets arsskrift*, 9.

Herbrand, J. (1967). *Investigations in Proof Theory* en *From Frege to Gödel. A Source Book in Mathematical Logic*. J. van Heijenoort ed. Harvard University Press. Cambridge, Mass.

Hodes, H. (1989). Three-valued logics: an introduction and comparison of various logical lexica and some philosophical remarks. *Annals of Pure and Applied Logic*, 43.

Huet, G. y et al, E. S. (1986). *Fundamentals of Artificial Intelligence*. Springer Verlag.

Iturrioz, L. (1976). Les algèbres de Heyting-Brouwer et de łukasiewizc trivalentes. *Notre Dame Journal of Formal Logic*, XVII(1):119–126.

Iturrioz, L. (1977). An axiom system for three-valued łukasiewicz propositional calculus. *Notre Dame Journal of Formal Logic*, XVIII(2):616–620.

Jeffrey, R. (1967). *Formal Logic: its scope and limits*. McGraw-Hill Inc. Traducción al español por Ed. Univ. de Navarra, 1986.

Jones, C. (1986). *Systematic software development using VDM*. Prentice-Hall, Englewood Cliffs, New Jersey.

Kleene, S. (1938). On a notation for ordinal numbers. *J. Symbolic Logic*, 3:150–155.

Kleene, S. (1952). *Introduction to Metamathematics*. van Nostrand and Princeton.

Kowalski, R. (1974). *Automated Theorem Proving.* Information Processing 74, Stockholm, North Holland.

Kunen, K. (1987). Negation in logic programming. *Journal of Logic Programming*, pp. 289–308.

Łukasiewicz, J. y Tarski, A. (1930). Untersuchungen über den aussagenkalkül. *Comptes rendus des séances de la Societé des Sciences et des Lettres de Varsovie*, 23:1–21.

Manna, Z. y Waldinger, R. (1985). *The Logical basis for Computer Programming.* Addison Wiley.

Martin-Löf, P. (1987). Truth of a proposition, evidence of a judgement, validity of a proof. *Synthese*, 73.

Mendelson, E. (1987). *Introduction to Mathematical Logic.* Wadsworth & Brook/Cole, third edition.

Mundici, D. (1987a). Satisfiability in many-valued sentential logic is np-complete. *Theoretical Computer Science*, pp. 145–153.

Mundici, D. (1987b). The turing complexity of af c*-algebras with lattice-ordered ko. *Lecture Notes in Computer Science*, 270:256–264.

Mundici, D. (1988a). The derivative of truth in lukasiewicz sentential calculus. En *VII Latin American Symposium on Mathematical Logic, Contemporary Mathematics AMS*, volumen 69, pp. 209–227.

Mundici, D. (1988b). Farey stellar subdivisions, ultrasimplicial groups, and ko of af c*-algebras. *Advances in Mathematics*, 68:23–39.

Mundici, D. (1990). Algebras of ulam's game with lies. En *New Problems in Logic and Philosophy of Science*, pp. 8–13, Viareggio, Italy.

Mundici, D. (1991). The complexity of adaptive error-correcting codes. *Lecture Notes in Computer Science*, 533:300–307.

Mundici, D. (1992). The logic of ulam's game with lies. *Knowledge, Belief, and Strategic Interaction, M.L. Dalla Chiara, C. Bicchieri, Editors. Cambridge Studies in Probability, Induction and Decision Theory. Cambridge University Press*, pp. 275–284.

Mundici, D. (1993). Logic of infinite quantum systems. *International Journal of Theoretical Physics*, 32:1941–1955.

Mundici, D. (1995a). Averaging the truth-value in łukasiewicz logic. *Studia Logica*, pp. 113–127.

Mundici, D. (1995b). Averaging the truth value in lukasiewicz logic. *Studia Logica, (special issue in honor of Helena Rasiowa)*, 55:113–127.

Mundici, D. (1996). Ullam game, the logic of maxsat, and many-valued partitions. En *Conference on Nonclassical Logics and Fuzzy Sets*, Linz.

Ojeda, M. (1996). *Métodos formales para normalización en lógica de primer orden usando la metodología TAS*. PhD thesis, Universidad de Málaga, España.

Ojeda, M., de Guzmán, I. P., y Enciso, M. (1996). *Theorem proving for temporal logic using the TAS paradigm*. En *Proceedings of Iberamia'96*, pp. 48–57, Cholula-Puebla (Méjico).

Oppacher, F. y Suen, E. (1988). Harp: A tableau-based theorem prover. *Journal of Automated reasoning*, 4.

P. Gochet, E. G. y et al., P. G. (1988). *From Standard Logic to Logic Programming*. A. Thayse editor. John Wiley and Sons.

Paech, B. (1986). *Gentzen-systems for propositional temporal logic*. En *En 2nd Workshop on Computer Science Logic*, Duisburg.

Pelletier, F. (1986). Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2.

Post, E. (1920). Introduction to a general theory of elementary propositions. *Bulletin of the American Mathematical Society*, 26.

Przymusinski, T. (1991). Three-valued nonmonotonic formalisms and semantics of logic programs. *Artificial Intelligence*, 49:309–343.

Ramsay, A. (1988). *Formal Methods in Artificial Intelligence*. Cambridge University Press.

Rasiowa, H. (1974). *An algebraic approach to non-classical logics*. Studies in Logic and the Foundations of Mathematics, 78. North-Holland.

Reichenbach, H. (1962). On the connection of the first-order functional calculus with many-valued propositional calculi. *Notre Dame Journal of Formal Logic*, 3:102–107.

Reichenbach, H. (1964). A note about connection of the first-order functional calculus with many-valued propositional calculi. *Notre Dame Journal of Formal Logic*, 5:158–160.

Rich, E. (1983). *Artificial Intelligence*. McGraw-Hill.

Robinson, J. (1965). A machine-oriented logic based on the resolution principle. *Journal ACM*, 12(1).

Rosser, J. y Turquette, A. (1952). *Many-valued Logic*. North-Holland.

Sanz, F. (1992). *Hacia una alternativa a resolución*. PhD thesis, Universidad de Málaga, España.

Słupecki, J. (1972). Completeness criterion for systems of many-vakued propositional calculus. *Studia Logica*, 30:153–157.

Smullyan, R., editor (1968). *First-Order Logic*. Springer- Verlag. Vuelto a publicar por Dover en 1995.

Suchoń, W. (1974). La méthode de Smullyan de construire le calcul n-valent de łukasiewicz avec implication et negation. *Reports on Mathematical Logic, Universidades de Cracovia y Katowich*, 2:37–42.

Surma, S. J. (1984). *An algorithm for axiomatizing very finite logic*. D.C. Rine (ed.): Computer science and multiple-valued logic: theory and applications. North Holland.

Tokarz, M. (1974). A method of axiomatization of łukasiewicz logics. *Studia Logica*, XXXIII(4):333–338.

Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 42:230–265. Corrección en v43 pp 544–546, 1937.

Turner, R. (1985). *Logics for Artificial Intelligence*. Ellis Horwood Limited.

Urquhart, A. (1986). *Many Valued Logic. Chap 2 of: Handbook of Philosophical Logic Vol. III: Alternatives in Classical Logic*. Kluwer Academic Publishers. Dordrecht.

Winker, S. (1982). Generation and verification of finite models and counterexamples using an automated theorem prover answering two open question. *Journal Assoc. Comp. Math.*, 2:273–284.

Woodcock, J. y Loomes, M. (1991). *Software Engineering Mathematics*. Pitman.

Woss, L. y Winker, S. (1984). *Open questions solved with the assistance of AURA*. Contemporary Mathematics.

# Índice alfabético