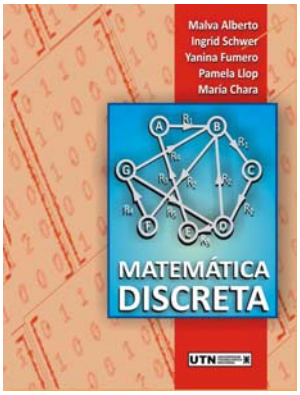
Editorial de la
Universidad Tecnológica Nacional



Matemática Discreta

Malva Alberto, Ingrid Schwer, Yanina Fumero,
Pamela Llop, María Chara

Facultad Regional Santa Fe

UTN UNIVERSIDAD TECNOLÓGICA NACIONAL

Editorial de la Universidad Tecnológica Nacional – edUTecNe
http://www.edutecne.utn.edu.ar
edutecne@utn.edu.ar

© [Copyright] La Editorial de la U.T.N. recuerda que las obras publicadas en su sitio web son *de libre acceso para fines académicos y como un medio de difundir el conocimiento generado por autores universitarios*, pero que los mismos y edUTecNe se reservan el derecho de autoría a todos los fines que correspondan.

Universidad Tecnológica Nacional – República Argentina

Rector: Ing. Héctor C. Brotto
Vicerrector: Ing. Carlos E. Fantini

edUTecNe – Editorial de la Universidad Tecnológica Nacional

Coordinador General: Ing. Ulises J. P. Cejas
Director de Ediciones: Ing. Eduardo Cosso
Coordinador del Comité Editorial: Ing. Juan Carlos Barberis
Área Comercialización: Ing. Hector H. Dabbadie
Áreas Pre-prensa y Producción: Téc. Bernardo H. Banega, Ing. Carlos Busqued

Prohibida la reproducción total o parcial de este material
sin permiso expreso de edUTecNe

Matemática Discreta
Malva Alberto, Ingrid Schwer, Yanina Fumero, Pamela Llop, María Chara

Diseño de tapa: *Ing. Carlos Busqued*
Pre-prensa interior: *Bernardo H. Banega*

La publicación de la presente obra a través de edUTecNe cuenta con el aval del Consejo Directivo de la Facultad Regional Santa Fe de la UTN, según Resolución N° 558 del 30/11/2009.

Impreso en Argentina – Printed in Argentina
ISBN 978-987-26665-1-4
Queda hecho el depósito que marca la ley 11.723
© edUTecNe, 2010
Sarmiento 440, Piso 6
(C1041AAJ) Buenos Aires, República Argentina

A nuestros alumnos, por habernos regalado la idea de escribir este libro.
A nuestras familias, por su paciencia ante tantas ausencias.

Las autoras.

Prólogo

Como docentes, realizamos permanente y sistemáticamente acciones que tienden a motivar, gestionar, asistir, implementar, contextualizar, redirigir y redimensionar las realizaciones didácticas en el aula, pensando en nuestros alumnos y atendiendo a la planificación, los contenidos, los materiales, los medios o recursos con los que contamos o los que podamos generar. Hacemos una revisión crítica y de fortalecimiento de nuestra propia gestión educativa. Miramos retrospectivamente el camino recorrido y detectamos errores, falencias, debilidades y vacancias. Pero también miramos prospectivamente hacia otras direcciones para poder mejorar las situaciones existentes. Integramos el diseño flexible del currículum con los materiales y recursos disponibles; la factibilidad del uso de los medios tecnológicos con el fortalecimiento del proceso de enseñanza y aprendizaje; la coherencia interna y manifiesta de la cátedra con la ordenanza de la carrera, la institución y la calidad educativa; la coherencia externa con los criterios y estándares de acreditación en los que estamos inmersos.

De este ejercicio continuo de revisión, búsqueda y solución, surge la necesidad de contar con materiales didácticos apropiados que puedan ser compartidos por docentes y alumnos. Creemos además, que estos materiales que son producidos y generados por los propios docentes de la cátedra y revisados por los estudiantes, pueden favorecer de manera especial la integración de conceptos y procedimientos disciplinares con la resolución de problemas de la especialidad.

La mirada crítica y permanente de los hechos que ocurren en el aula, el proceso de acreditación de la carrera Ingeniería en Sistemas de Información de la UTN Santa Fe, la finalización del proyecto de investigación “Secuencias Didácticas en Matemática y Tecnologías Básicas” (PID 25/O106) que forma parte del programa “Tecnología Educativa y Enseñanza de la Ingeniería” de la Universidad Tecnológica Nacional, la búsqueda permanente de estándares para mejorar la calidad educativa, la participación de docentes jóvenes que están doctorándose en el CONICET, son algunas de las motivaciones que incidieron más fuertemente para llevar a cabo esta nueva edición.

Pensamos en un texto dirigido a estudiantes de los primeros niveles de estudios superiores, tales como ingenierías, licenciaturas y tecnicaturas con orientaciones en las ciencias de la programación y la computación. Por ello, hemos acotado y recortado los contenidos disciplinares priorizando los aspectos pedagógicos y didácticos acerca del saber enseñando, por sobre el saber científico. El texto ofrece numerosos ejemplos con detalles aclaratorios y suficientes problemas para resolver, que ponen de manifiesto la integración y abundancia de aplicaciones de la lógica, los métodos del conteo, las variables discretas, las estructuras algebraicas finitas, la teoría de grafos y la teoría elemental de los números enteros en las ciencias de la computación. Hemos dedicado tiempo a las lecturas complementarias, a las

actividades para la comprensión y a las aplicaciones no rutinarias. En todo momento intentamos favorecer la resolución de situaciones problemáticas, la participación activa, la búsqueda de alternativas propias y la toma de decisiones razonadas. Los ejercicios y problemas seleccionados permiten afianzar el material teórico de cada sección, enlazar ideas con temas desarrollados anteriormente y favorecer la reflexión, la síntesis y la apertura a nuevos interrogantes.

Nuestra primera convocatoria es "trabajar activamente en el aula" y la segunda es alentar la participación para que "cada estudiante se comprometa con su propio aprendizaje".

Características de la organización del texto

El texto es una verdadera introducción a un curso de Matemática Discreta y se lo puede desarrollar a lo largo de un semestre con una carga horaria áulica de seis horas semanales. Para una propuesta cuatrimestral, de dieciséis semanas, es aconsejable alguna selección de contenidos.

En cada capítulo hemos introducido contenidos conceptuales, procedimentales y actitudinales y seleccionado una aplicación o una lectura complementaria.

Dividimos el texto en ocho capítulos. En el *capítulo 1* hacemos una introducción a la lógica proposicional clásica e incluimos una lectura complementaria relativa a las demostraciones en matemática. Los algoritmos están escritos en pseudo código y hemos unificado su notación y estructura dadas inicialmente, a lo largo de todo el texto. No pretendemos, que el lector diseñe un algoritmo; el objetivo es evaluar una proposición, seguir un algoritmo y decidir qué tarea resuelve. Su lectura es opcional y lo recomendamos para los estudiantes interesados en las áreas de programación y computación.

En el *capítulo 2* presentamos distintas opciones y estrategias para contar colecciones finitas de objetos y ofrecemos una aplicación para determinar cuántas comparaciones son necesarias realizar para ordenar una lista finita de números, utilizando distintos métodos: burbujas, selección e inserción. Es importante el trabajo planteado con los números combinatorios y sus propiedades.

En el *capítulo 3*, introducimos la teoría de números, hacemos un primer estudio sobre la divisibilidad, los números primos y compuestos, el teorema fundamental de la aritmética y el algoritmo de la división entera; utilizamos congruencias enteras para resolver ecuaciones y problemas. Usamos el principio de inducción matemática para probar propiedades referidas a los enteros positivos. Mediante este principio probamos que algunos algoritmos propuestos resuelven correctamente ciertas tareas para cualquier número natural dado.

En el *capítulo 4* estudiamos las relaciones de recurrencia y una aplicación a la generación de números aleatorios. En especial desarrollamos las relaciones de recurrencia lineales de primer y segundo orden, con coeficientes constantes, homogéneas y no homogéneas. En las lecturas complementarias proponemos dos temas: un desarrollo sobre las funciones recursivas y una relación muy interesante que se genera entre los números de Fibonacci y el algoritmo de Euclides.

En el *capítulo 5* revisamos una buena cantidad de ejemplos para introducir estructuras algebraicas: grupos, anillos, cuerpos y álgebras de Boole. Utilizamos los axiomas y propiedades de las álgebras booleanas para simplificar expresiones de la aritmética booleana. Las funciones booleanas escritas en su forma normal conjuntiva o disyuntiva, permitirán a los jóvenes continuar con otro curso más avanzado en arquitectura de las computadoras. Finalmente, en la lectura complementaria, nos referimos a la codificación y decodificación de mensajes.

En el *capítulo 6* hacemos una presentación de digrafos y grafos, con el diseño de algoritmos y referencias a temas previos, poniendo énfasis en árboles de refutación de fórmulas de la lógica proposicional, en árboles con raíz, binarios, completos, etc. Damos algunas definiciones en forma clásica y otras en forma recursiva. Es un tema con amplias aplicaciones en la investigación de operaciones, optimización de recursos y evaluación de la performance de sistemas informáticos.

En el *capítulo 7* presentamos una aplicación de la teoría de digrafos: el estudio de autómatas. Damos numerosos ejemplos de máquinas de estado finito y de aquellas que son reconocedoras de un lenguaje. Luego trabajamos con la simplificación de estas máquinas, para optimizar los estados internos que poseen.

En el *capítulo 8*, abordamos la lógica de primer orden, ampliando los contenidos dados en el capítulo 1. En este momento ya contamos con numerosos ejemplos desarrollados a lo largo del texto que aseguran una rápida familiarización con la presentación formal que el tema requiere.

En cada capítulo aparecen actividades, lecturas, preguntas y observaciones. Las actividades están intercaladas dentro de los desarrollos teóricos. La adopción de la palabra "problema" debe entenderse en un sentido muy amplio; en algunos casos representa un ejercicio rutinario; en otros, un ejercicio no rutinario o un "verdadero problema" y esta clasificación dependerá mucho del interés y de los conceptos matemáticos que posean nuestros lectores. Consideramos prioritario resolver todas las actividades.

En todos los capítulos incluimos problemas complementarios y ejercicios de opción múltiple. Para el abordaje de los problemas complementarios es aconsejable la consulta con el docente. A veces, éstos tienen como objetivo hacer una revisión global del contenido pero en otras oportunidades pretendemos dar nuevas propuestas para que el estudiante pueda avanzar en temas más complejos. Los ejercicios de opción múltiple son sintetizadores de contenidos desarrollados.

Sobre el final del texto proporcionamos respuestas y/o sugerencias para resolver la mayoría de los problemas que figuran dentro de las actividades. Nuestra propuesta finaliza con una adecuada cita de la bibliografía consultada, donde es posible encontrar las demostraciones de la totalidad de los teoremas y nuevos ejercicios y problemas para resolver.

Esperamos satisfacer las expectativas de nuestros lectores y en especial las de nuestros estudiantes. Es nuestro deseo poder mejorar este material y para ello necesitamos conocer las distintas opiniones de todos aquellos que quieran hacernos llegar sus observaciones.

Agradecimientos

La resignificación y reorganización de este libro de texto es el resultado del sostenido, fecundo y constante trabajo llevado a cabo por una comunidad educativa que cada año aportó una permanente revisión y actualización de contenidos, actividades y aplicaciones.

Pasó una década desde que expresamos nuestros primeros y sinceros reconocimientos a los auxiliares y jefes de trabajos prácticos de la cátedra Matemática Discreta, a las autoridades, colegas, amigos, estudiantes y graduados que nos acompañaron para que los otrora apuntes de clase, existan hoy, como un texto de cátedra. Viviana Cámara, Cristina Rogiano, Silvina Meinero, Juan Pablo Puppo y Liliana Fiorito hicieron que nuestra tarea sea más sencilla. Reconocemos la dedicación puesta, hace ya varios años atrás por Daniel González, Silvina Ariel, Ramiro Jorge y Luis Larrateguy. Durante el último año se sumaron las alumnas Mariana Herrera, Cecilia Gaspoz, Luciana Bertona y María Julia Blas.

Alumnos y colegas de la carrera Ingeniería en Sistemas de Información de la Facultad Regional Santa Fe de la Universidad Tecnológica Nacional, han permitido que nuestra labor se desarrolle en un clima de constante interacción. Nuestro profundo reconocimiento a los ingresantes 2008, 2009 y 2010 por el esfuerzo realizado al estudiar estos contenidos y resolver los ejercicios. Extendemos nuestro agradecimiento a todos los estudiantes que han hecho que nuestro proyecto se mantenga en permanente acción.

Las observaciones y sugerencias para la selección de los problemas de aplicación dadas por Marta Castellaro mejoraron nuestros escritos. Stella Agüeria, Hernán Melgratti, Aldo Vecchietti, Silvio Gonnet y Pablo Marchetti sugirieron lecturas y material bibliográfico. No podemos dejar de mencionar la valiosa colaboración prestada por Manuel Marina a través de las correcciones y sugerencias realizadas en los algoritmos propuestos.

Agradecemos los ejercicios propuestos por los alumnos que asistieron al seminario extracurricular sobre Teoría del Conteo desarrollado durante en año 2010.

Las grandes o pequeñas sugerencias de toda esta comunidad sirvieron para incluir más ejemplos, eliminar ambigüedades y mejorar la presentación del contenido.

A EdUTecNe, la editorial de la Universidad Tecnológica Nacional por la posibilidad de la difusión del conocimiento.

Las Autoras

Capítulo 1
Lógica Proposicional

1.1 Introducción

En este capítulo haremos inicialmente un acercamiento intuitivo a la Lógica Proposicional y continuaremos más formalmente con su sintaxis y semántica. Avanzaremos con la identificación de reglas y procedimientos que nos permitirán decidir si los argumentos utilizados son válidos o no, y finalmente mostraremos aplicaciones y métodos para obtener demostraciones. Usaremos estos contenidos muy frecuentemente para justificar parte del trabajo matemático que desarrollaremos a lo largo del texto.

La palabra lógica tiene sus raíces en Grecia, derivando de *logos*, que significa *palabra, tratado, idea, principio, pensamiento o razón*; y del término *ica* que significa, *relacionado a* o *relacionado con*; entonces la lógica puede ser asociada con *lo relacionado al pensamiento o a la razón*.

La inclusión de la Lógica Proposicional no es arbitraria. Varios y buenos motivos justifican su inserción como contenido de un texto de Matemática Discreta. El primero, porque constituye un buen ejemplo al cual nos referiremos cuando aprendamos Álgebras de Boole. El segundo, porque facilitará la introducción al aprendizaje de la Lógica de Primer Orden, que desarrollaremos más adelante y el tercero porque tiene muy interesantes aplicaciones en las ciencias de la computación (esta enumeración no es jerárquica). Usaremos la Lógica Proposicional para modelar y simplificar circuitos lógicos y para comprender procesos de decisión en la lectura y seguimiento de algoritmos.

La Lógica Proposicional y la Lógica de Primer Orden o Lógica de Predicados forman parte de las Lógicas Clásicas y constituyen el sustento de la Programación Lógica. El razonamiento lógico se emplea en matemática para demostrar teoremas; en ciencias de la computación para verificar si los programas son correctos o no, en las ciencias físicas y naturales, para obtener conclusiones de experimentos; en las ciencias sociales y en la vida cotidiana, para resolver una increíble cantidad de problemas. Nos referiremos a la Lógica Proposicional Clásica con la abreviatura LPC y utilizaremos LPO para denotar la Lógica de Primer Orden.

La LPC es una importante rama de la Matemática que se formaliza en el siglo XIX. Se atribuye a Aristóteles (384-322 a.c.) el primer estudio sistemático y formal del razonamiento lógico y el empleo del término *Lógica* para referirse al estudio de los argumentos o a la ver-

dad en la ciencia. El filósofo y matemático alemán G. W. Leibniz (1646-1716) ha realizado importantes contribuciones durante el siglo XVII y fue G. Boole (1815-1864), quien la redescubre y extiende su desarrollo. Enfoques más modernos se deben a G. Frege (1848-1925); A. N. Whitehead (1861-1947) y B. Russell (1872-1970), entre otros matemáticos.

1.2 El lenguaje

En el desarrollo de toda teoría (matemática o no) se hacen afirmaciones en forma de oraciones. En la evaluación numérica que se realiza en las computadoras, en el análisis del lenguaje desde el punto de vista lógico, en la construcción, prueba y verificación de programas computacionales, interesa un tipo especial de oraciones a las que daremos el nombre de *proposiciones*. Intuitivamente, las *proposiciones* son oraciones o afirmaciones *que tienen un único valor de verdad: o son verdaderas o son falsas.* Son proposiciones: "el triángulo es un polígono de tres lados"; "Avatar y Titanic fueron dirigidas por James Cameron"; "2010 es un año bisiesto"; "El matemático G. Boole falleció en 1964".

Es decir, las proposiciones son oraciones que asumen alguno de estos dos valores: verdadero o falso (pero no ambos simultáneamente). Las siguientes afirmaciones son ejemplos de proposiciones: "1 es un número primo", "8 es múltiplo de 2", "En el idioma español, las vocales son a, e, i, o, u", "El identificador 'A5matdis' es válido en el lenguaje Pascal", "La garza es un mamífero" y sus valores de verdad son "falso", "verdadero", "verdadero", "verdadero" y "falso" respectivamente. Estas proposiciones se llaman *primitivas o atómicas*, ya que no pueden descomponerse o subdividirse en partes más simples. La LPC no asigna los valores de verdad (verdadero o falso) a cada proposición atómica. Éstas ya tienen un valor asignado cuando se las analiza lógicamente. La proposición "El identificador 'A5matdis' es válido en el lenguaje Pascal" es verdadera porque así lo establecen las reglas que tiene el Lenguaje Pascal para aceptar a una sucesión de caracteres alfanuméricos como un identificador. La proposición "La garza es un mamífero" es falsa porque las Ciencias Biológicas así lo han establecido.

En este análisis sólo tendremos en cuenta oraciones que puedan clasificarse como verdaderas o falsas; no nos interesan aquellas que se refieran a una opinión individual, por ejemplo, "creo que el espíritu de la resolución es perjudicar a la prensa independiente"; "creo que mi profesor se equivoca"; "estimo que es necesario un cambio en la política agropecuaria"; también quedarán excluidas las oraciones imperativas, tales como "Vete a la cama" o "Cierra la puerta", así como las oraciones interrogativas "¿Quieres ir al cine?", "¿Vienes a cenar?" o las exclamativas "¡Cómo llueve!", "¡Qué calor!", etc. Más adelante trabajaremos con expresiones del tipo "x es mayor o igual que y"; "$x^2 + y^2 = z^2$"; "x es número par", pero hasta el momento, no son proposiciones.

A partir de las proposiciones primitivas podemos obtener nuevas *proposiciones,* llamadas *compuestas*, combinándolas por medio de operadores lógicos llamados conectivos. Una propiedad básica es que la verdad o falsedad de la proposición compuesta depende de la verdad o la falsedad de las proposiciones primitivas o atómicas que la componen.

La LPC da un modelo matemático para los razonamientos que pueden hacerse en base a distintas formas de operar con las proposiciones. Este modelo matemático nos ayuda a comprender mejor las formas básicas del pensamiento racional. Para ello debemos tener en cuenta dos aspectos de la LPC: la sintaxis o gramática, que especifica qué secuencias de símbolos se consideran fórmulas bien formadas y la semántica, que permite interpretar las fórmulas y asignarles su verdad o falsedad.

En matemática, en lógica clásica y en ciencias de la computación, un lenguaje formal es definido a partir de un conjunto (finito o infinito numerable) de símbolos y de reglas que permiten operar esos símbolos. El concepto más primitivo es el de *símbolo*; son símbolos: a, b, /, 0, 3, *, i. Consideraremos como *alfabeto* un conjunto no vacío y finito de símbolos; por ejemplo, son alfabetos $\Sigma_1 = \{0, 1, *, \hat{}\}$ o $\Sigma_2 = \{a, b, c\}$ (Σ se lee sigma). Con los símbolos del alfabeto podemos formar *cadenas de símbolos o palabras*; éstas contienen un número finito de símbolos del alfabeto. Pero, ¿cómo formamos las palabras? El conjunto de las reglas que permiten formar las palabras del lenguaje se llama *la gramática formal (o sintaxis)*. A una cadena de símbolos formada de acuerdo a una gramática dada se la llama una *fórmula bien formada* (palabra, fórmula bien definida o simplemente fórmula) del lenguaje. Hablar de un lenguaje formal definido a partir de los símbolos de un alfabeto, es análogo a hablar del conjunto de todas sus fórmulas bien formadas. A diferencia de lo que ocurre con el alfabeto (que debe ser un conjunto finito) y con cada fórmula bien formada (que debe tener una longitud también finita), un lenguaje formal puede estar compuesto por un número infinito de fórmulas bien formadas.

Por ejemplo, dado el alfabeto $\Sigma_2 = \{a, b, c\}$, y la gramática que permite identificar las fórmulas bien formadas como aquellas que utilizan cantidades iguales de símbolos a y b, entonces, algunas fórmulas bien formadas del lenguaje generado a partir de Σ_2 son: ab, bca, c, cacbcacbc, cababbabac, bbbaaa, ccc, etc.; el lenguaje formal es el conjunto de todas las fórmulas bien formadas. Un caso especial de fórmula bien formada para este ejemplo es la cadena vacía. Esta cadena vacía la simbolizaremos como λ (lambda) y no contiene símbolos del alfabeto; en este caso tiene cero símbolos a y b.

Para algunos lenguajes formales existe una semántica formal que puede interpretar y dar significado a las fórmulas bien formadas del lenguaje. Sin embargo, una semántica formal no es condición necesaria para definir un lenguaje formal, y eso es una diferencia esencial con los lenguajes naturales.

Si el alfabeto es $\Sigma = \{p, q, \wedge, (,)\}$ y las únicas reglas de la sintaxis son:

i) p, q son fórmulas.

ii) Si m y n son fórmulas entonces $(m \wedge n)$ es una fórmula.

Entonces son fórmulas: p, q, $(p \wedge p)$, $(p \wedge q)$, $((p \wedge q) \wedge (q \wedge p))$, $((q \wedge q) \wedge q)$, etc.

La semántica permite dar significado a estas fórmulas. Podemos interpretarlas, y al interpretarlas les asignaremos un valor de verdad. Por ejemplo:

p: "2 es número par"; q: "3 es número impar"; $\wedge$: "y" , luego interpretamos a $(p \wedge q)$ como "2 es número par y 3 es número impar". Esta interpretación convierte a p, q, $(p \wedge q)$ en proposiciones.

Resumiendo, en LPC debemos considerar dos aspectos importantes: por un lado tenemos la sintaxis (o gramática) que es la parte que especifica las secuencias de símbolos del alfabeto que están bien definidas o bien formadas y es la que define las fórmulas del lenguaje. Por otro lado está la semántica, que interpreta las fórmulas, dándoles un significado que permite decidir su valor de verdad (verdadero o falso).

1.3 Sintaxis

Cada lenguaje formal tiene *símbolos* propios y una *sintaxis*, es decir una especificación rigurosa de las secuencias de símbolos que están permitidas. Así, la sintaxis comienza con una especificación del *alfabeto* del lenguaje, esto es, el conjunto de símbolos con los cuales se construyen secuencias bien formadas. Veamos a continuación las definiciones sintácticas correspondientes a la LPC.

Definición 1.1

El *alfabeto* $\sum = \{p, |, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$ de la LPC consiste en un conjunto no vacío y finito de símbolos, que describimos a continuación:

i) Variables proposicionales: p, p|, p||, p|||, …. Para mayor simplicidad las indicaremos como $p_0 = p$; $p_1 = p|$; $p_2 = p||$, etc o simplemente p, q, r, s, etc. Usaremos las últimas letras de nuestro abecedario para las variables proposicionales. Indicaremos con Var al conjunto de variables proposicionales.

ii) Conectivos lógicos: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.

iii) Símbolos auxiliares: (,). Estos símbolos se llaman de agrupación.

A cada posible conjunto de variables proposicionales corresponde un alfabeto diferente; es decir, las variables proposicionales son símbolos propios de cada alfabeto; pero cada alfabeto de la LPC tiene los mismos cinco conectivos y los mismos símbolos auxiliares; es decir, los conectivos y los símbolos auxiliares son símbolos comunes para todos los alfabetos de la LPC. Usando los símbolos del alfabeto, podemos formar secuencias de símbolos. El conjunto de secuencias bien formadas (fórmulas) se define como sigue:

Definición 1.2

Una *fórmula bien formada* (o simplemente *fórmula*), que indicaremos con fbf, se define *recursivamente* como:

i) Una variable proposicional es una fbf.

ii) Si θ (leemos theta o tita) es una fbf, $\neg\theta$ es una fbf.

iii) Si θ y φ (leemos phi , fi) son fórmulas bien formadas (en adelante, fbfs) entonces también son fbfs $(\theta \wedge \varphi)$, $(\theta \vee \varphi)$, $(\theta \rightarrow \varphi)$ y $(\theta \leftrightarrow \varphi)$

iv) Una sucesión o cadena de variables proposicionales, conectivos o símbolos del alfabeto es una fbf si y sólo si puede obtenerse mediante un número finito de aplicaciones de las reglas i), ii) y iii).

Indicaremos con Form al conjunto de todas las fórmulas.

Ejemplo 1

Si el conjunto de variables proposicionales es: Var = {p, q, r}, entonces son fórmulas bien formadas p, $\neg$q, $((p \vee q) \rightarrow r)$, $(p \wedge (q \vee r))$, entre otras. La fórmula más simple es un *átomo*. Las variables proposicionales son átomos y son fbfs. Las fórmulas que no son átomos, por ejemplo $\neg$p, $(p \vee q)$, se llaman *fórmulas compuestas*.

Las siguientes no son fórmulas bien formadas dado que no pueden ser "generadas" aplicando un número finito de veces, las reglas dadas:

a) (p), porque los paréntesis deberían quitarse.

b) $p \vee q$, no es fórmula porque, estrictamente hablando, debería encerrarse por paréntesis de acuerdo a la regla iii) de la definición.

c) $(p \vee q \vee r)$, no es una fórmula, dado que la ubicación de los paréntesis no está de acuerdo con la definición.

d) $(p \vee (q \wedge \rightarrow r))$, la secuencia "$\wedge \rightarrow$" no puede ser generada por las reglas de la definición de sintaxis dada.

e) $((p \vee q) \leftrightarrow \neg\neg p \rightarrow q)$, faltan paréntesis. Hay lugares donde se pueden insertar pares de paréntesis en esta secuencia para convertirla en una fórmula.

Por ejemplo, $((p \vee q) \leftrightarrow (\neg\neg p \rightarrow q))$, $((p \vee q) \leftrightarrow \neg(\neg p \rightarrow q))$, $(((p \vee q) \leftrightarrow \neg\neg p) \rightarrow q)$ son fórmulas.

Ejemplo 2

La cadena de símbolos $(p \vee (q \rightarrow r))$ es una fbf. Puede obtenerse aplicando un número finito de veces las reglas i) y iii).

En efecto: p, q, r $\in$ Var y luego son fbfs por i); como q y r son fbfs, entonces $(q \rightarrow r)$ es fbf por iii) y finalmente como p y $(q \rightarrow r)$ son fbfs entonces $(p \vee (q \rightarrow r))$ es fbf nuevamente por iii).

Definición 1.3

El lenguaje proposicional **L** dado por el alfabeto $\sum$ es el conjunto de todas las fórmulas (bien formadas) que pueden construirse partiendo de los símbolos del alfabeto. Esta definición justifica que identifiquemos al lenguaje L con Form, que es el conjunto de sus fórmulas.

Si el alfabeto Σ_1 y el alfabeto Σ_2 son diferentes, es decir, si el conjunto de átomos pertenecientes a Σ_1 es diferente del conjunto de átomos pertenecientes a Σ_2, luego el lenguaje proposicional dado por Σ_1 es diferente del lenguaje proposicional dado por Σ_2. Podemos observar que un lenguaje proposicional es siempre un conjunto infinito, aún cuando el conjunto de variables proposicionales del alfabeto pueda ser unitario.

Actividad 1

Problema Nº1: Sea $\Sigma = \{*, \&\}$ un alfabeto. Las siguientes reglas definen las fbfs:

i) * es una fórmula.

ii) Si X es una fórmula, &X y *X también lo son.

iii) X es una fórmula si y sólo si se la puede obtener aplicando un número finito de veces las reglas anteriores.

1.1) Escribe tres expresiones que no sean fórmulas y tres expresiones que lo sean.

1.2) Decide si cada expresión es o no es, una fórmula. En caso de serlo muestra la sucesión de reglas aplicadas para obtenerla:

a) * b) ***& c) & d) &&& e) &&*

1.3) ¿Cómo caracterizarías las fbfs con tus propias palabras?

Problema Nº2: Sea $\Sigma = \{a, b, *\}$ un alfabeto. Las siguientes reglas definen las fbfs:

i) Las fórmulas del lenguaje contienen un símbolo a y un símbolo b.

ii) En cada fórmula, a aparece antes que b.

iii) X es una fórmula si y sólo si se la puede obtener aplicando un número finito de veces las reglas anteriores.

2.1) Escribe tres expresiones que no sean fórmulas y tres expresiones que lo sean.

2.2) Para cada una de las siguientes expresiones, decide si es o no una fórmula:

a) * b) a*b c) ab d) ba e) *a***b

Problema Nº3: Decide si las siguientes expresiones son fórmulas de la LPC. Justifica. (Asumimos que el lenguaje consta de un alfabeto donde Var = {p, q, r, s})

3.1) $(p \vee q) \rightarrow r$

3.2) $((p \vee (q \rightarrow r)) \rightarrow s)$

3.3) $((\neg\neg p \rightarrow p) \rightarrow q)$

3.4) $(p \vee \neg(q \wedge r))$

1.4 Semántica

En la sección anterior identificamos el lenguaje proposicional definido sobre un alfabeto dado con el conjunto de fórmulas bien formadas que se pueden construir partiendo de ese alfabeto. En esta sección definiremos la semántica de este conjunto. Es decir el significado que adquiere una fórmula. *Trataremos a las fórmulas como proposiciones*, como enunciados

a los que se puede asignar uno de los valores de verdad “verdadero” o “falso”. En adelante, al referirnos a “proposiciones” haremos referencia a fórmulas con significado.

1.4.1 Informalmente

Una fórmula puede ser o verdadera o falsa, dependiendo de la verdad o falsedad de las fórmulas más simples que son sus componentes.

Por ejemplo, la verdad o falsedad de $(p \vee q)$ depende de los valores de verdad de sus átomos p y q. Así, podemos determinar la verdad o falsedad de una fórmula recurriendo a los valores de verdad dados por alguna *interpretación* de sus átomos. A continuación vamos a explicar informalmente cómo las fórmulas de la LPC adquieren sus valores de verdad, es decir, vamos a ver cómo las interpretamos.

i) Consideramos una fórmula de la forma $\neg\theta$, donde θ es una fórmula arbitraria. El conectivo usado “$\neg$” se llama *negación*. Decimos que la fórmula $\neg\theta$ (se lee no tita) es verdadera si la fórmula θ es falsa y que la fórmula $\neg\theta$ es falsa si la fórmula θ es verdadera. La siguiente tabla, llamada *tabla de verdad* ayuda a la comprensión de lo expresado, donde escribimos 0 para indicar que la proposición es falsa y 1 para indicar que es verdadera:

Tabla 1.1

θ	$\neg\theta$
0	1
1	0

Este conectivo se usa para modelar la palabra “no” de nuestro lenguaje cotidiano. Es posible utilizar otras expresiones como nunca, jamás, no es cierto, es falso que, sin, carece de, etc. Siendo p: “Juan es simpático”, entonces $\neg$p: “Juan no es simpático” o bien $\neg$p: “Juan es antipático”.

Por ejemplo, si interpretamos a “p” como “Lógica Proposicional es uno de los temas de Matemática Discreta”, entonces “$\neg$p” es “Lógica Proposicional no es uno de los temas de Matemática Discreta”. En este caso “p” admite como valor de verdad el verdadero mientras que “$\neg$p” tiene como valor de verdad el falso. Otra interpretación para “p” puede convertirla en una fórmula falsa, tal es el caso para p: “2 es número impar”; luego $\neg$p será “2 es número par” que es verdadera.

ii) Consideramos ahora la fórmula de la forma $(\theta \wedge \varphi)$, siendo θ y φ fórmulas arbitrarias. El conectivo “$\wedge$” se llama *conjunción* y la fórmula $(\theta \wedge \varphi)$ se lee θ y φ (tita y fi). Una fórmula del tipo $(\theta \wedge \varphi)$, es verdadera sólo cuando ambas, “θ” y “φ” son verdaderas. Esto es análogo al uso de la palabra “y” del lenguaje usual. Es posible usar otros conectivos con el mismo significado tales como pero, e, aunque, aún cuando, sin embargo, además, mientras que, etc. La oración “La palabra ‘los’ es un artículo mientras que ‘ante’ es una preposición” es verdadera, mientras que la oración “2 es número primo y 3 es número par” es falsa. La oración

"ingresé a la Universidad aunque no aprobé el examen de física" admite como modelo la fórmula ($p \wedge \neg q$), siendo p: "ingresé a la Universidad" y q: "aprobé el examen de física".

La Tabla 1.2 expresa los valores de verdad de la conjunción ($\theta \wedge \varphi$) mediante su correspondiente tabla de verdad.

Tabla 1.2

θ	φ	$(\theta \wedge \varphi)$
0	0	0
0	1	0
1	0	0
1	1	1

La expresión "10 es múltiplo de 2 y de 6" es falsa porque es la conjunción de las proposiciones atómicas: "10 es múltiplo de 2", que es verdadera, con la proposición: "10 es múltiplo de 6", que es falsa. La proposición compuesta "10 es un múltiplo de 2 y de 5" es verdadera porque es la conjunción de proposiciones atómicas verdaderas.

La conjunción aparece como el primer conectivo lógico que es un operador binario, ya que para poder definir la fórmula ($\theta \wedge \varphi$) se necesitan dos fbfs. Éstas pueden tener valores de verdad "0" o "1", por lo que necesitamos describir lo que ocurre en cada caso. En la *tabla de verdad*, la lista de los cuatro pares de asignaciones de verdad posibles para "θ" y "φ" se puede hacer en cualquier orden. Sin embargo, preferimos que adhieran al orden aquí propuesto porque es el que usaremos a lo largo del texto.

"Trabajo despacio pero sin pausas" es una conjunción donde explícitamente no está la "y". Existen oraciones donde la "y" no tiene un uso conjuntivo, por ejemplo en: "Pedro y Juan son primos"; "Valeria y Juan son hermanos"; "Ana y Rosalía son cuñadas". Estas expresiones relacionales serán tratadas en LPO mientras que en LPC sólo pueden ser operadas como los átomos p: "Pedro y Juan son primos", q: "Valeria y Juan son hermanos" y r: "Ana y Rosalía son cuñadas".

iii) Continuamos con una fórmula de la forma ($\theta \vee \varphi$), siendo θ y φ fórmulas arbitrarias. El conectivo "$\vee$" se llama *disyunción (incluyente)* y la fórmula ($\theta \vee \varphi$), se lee θ o φ (tita o fi). La fórmula del tipo ($\theta \vee \varphi$), es falsa sólo cuando ambas, "θ" y "φ" son falsas. Esto es análogo al uso de la palabra "o" del lenguaje usual. Son disyunciones oraciones como "La demostración puede hacerse por el método directo o por el absurdo"; "El problema puede resolverse en forma iterativa o recursiva".

Es posible usar otros conectivos con el mismo significado tales como "y/o". Esta disyunción es incluyente, en el sentido siguiente: la fórmula del tipo ($\theta \vee \varphi$) es verdadera cuando al menos una de las fórmulas resulta verdadera; es decir es verdadera si θ es verdadera; si φ es verdadera o si θ y φ son ambas verdaderas.

La Tabla 1.3 describe en forma tabular la disyunción ($\theta \vee \varphi$):

Tabla 1.3

θ	φ	$(\theta \vee \varphi)$
0	0	0
0	1	1
1	0	1
1	1	1

"El programa utilizado es erróneo o la salida fue mal seleccionada" es la disyunción entre "El programa utilizado es erróneo" con "la salida fue mal seleccionada".

En la expresión "Mi mascota está viva o muerta", la "o" es excluyente, en el sentido que la verdad de una de ellas excluye la posibilidad de verdad en la otra; si la proposición p:"mi mascota está viva" es verdadera, entonces la proposición q: "mi mascota está muerta" es falsa. Esta "o" excluyente puede ser marcada en el lenguaje diario al decir "Mi mascota está, o viva o muerta". Mientras que en la expresión "Soy profesora o alumna" la o es incluyente dado que la proposición verdadera r:"soy profesora" no excluye la posibilidad que s:"soy alumna" también pueda ser verdadera. Ejemplos similares son "Juan es mayor o menor de edad" con una o excluyente mientras que en "me comunico por teléfono o fax" se trata de una o incluyente.

Debe quedar claro que el alfabeto del lenguaje de la LPC no cuenta con un conectivo que represente a esta disyunción excluyente. Sin embargo vamos a crear un símbolo para este *"o exclusivo",* también llamado *"diferencia simétrica",* que se denotará por $(\theta \veebar \varphi)$ y que leeremos como "o θ o φ". Más adelante justificaremos esta introducción cuando identifiquemos los valores de verdad de la fórmula $(\theta \veebar \varphi)$ con exactamente los mismos valores de verdad de la fórmula $((\theta \wedge \neg\varphi) \vee (\neg\theta \wedge \varphi))$. La fbf $(\theta \veebar \varphi)$ es verdadera si una o la otra, pero *no ambas* fórmulas son verdaderas.

La fórmula "10 es múltiplo de 2 o de 5" es verdadera (disyunción incluyente). Mientras que la proposición "O 10 es múltiplo de 2 o 10 es múltiplo de 5" es falsa (disyunción excluyente).

Son fórmulas disyuntivas: "Juan nació en 1982 o en 1983"; "$5 \geq 2$"; "Por la mañana, concurro a la Facultad o resuelvo problemas de Matemática"; "Desayuno mate o café";

"Juan comprará o una chomba o un jeans".

La Tabla 1.4 interpreta todos los valores de verdad $((\theta \wedge \neg\varphi) \vee (\varphi \wedge \neg\theta))$. La Tabla 1.5 sintetiza los valores de verdad de la fórmula $(\theta \veebar \varphi)$.

Tabla 1.4

θ	φ	$\neg\theta$	$\neg\varphi$	$(\theta \wedge \neg\varphi)$	$(\neg\theta \wedge \varphi)$	$((\theta \wedge \neg\varphi) \vee (\neg\theta \wedge \varphi))$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

Tabla 1.5

θ	φ	$(\theta \veebar \varphi)$
0	0	0
0	1	1
1	0	1
1	1	0

Con el objeto de no confundir los símbolos utilizados precedentemente, llamaremos *disyunción* a "$(\theta \vee \varphi)$" y *diferencia simétrica* o *disyunción excluyente* a "$(\theta \veebar \varphi)$".

Existen renglones donde estas tablas coinciden. Numerosos ejemplos tomados de la matemática nos muestran estas coincidencias. Cuando decimos "$5 \geq 0$" estamos hablando de una proposición verdadera porque la expresión "$5 \geq 0$" se traduce, en términos de operaciones lógicas, como "5 es mayor que 0" o "5 es igual a 0"; en este caso, la disyunción puede pensarse como incluyente o excluyente y en cualquiera de ellos, la fórmula es verdadera porque sólo la proposición "5 es mayor que 0" es verdadera.

iv) En la fórmula de la forma $(\theta \rightarrow \varphi)$, siendo θ y φ fórmulas arbitrarias, encontramos el conectivo "$\rightarrow$" que se llama *implicación* y la fórmula $(\theta \rightarrow \varphi)$ se lee si θ entonces φ, si θ luego φ o bien, φ si θ . Es verdadera en todos los casos, excepto donde "θ" es verdadera y "φ" es falsa. (No se pretende que una fórmula verdadera implique una fórmula falsa). Es verdadera la fórmula "si $3 + 4 = 8$, entonces $4 + 5 = 10$", aunque "$3 + 4 = 8$" y "$4 + 5 = 10$" son falsas; es falsa la fórmula "si $3 + 4 = 7$, entonces $4 + 5 = 10$". Podemos observar que la fórmula de la forma $(\theta \rightarrow \varphi)$ es verdadera toda vez que θ sea falsa, sin importar la interpretación de φ. La fórmula "θ" se denomina *hipótesis, antecedente o premisa* de la implicación, y "φ" se denomina *conclusión o consecuente* de la implicación.

En el lenguaje diario escuchamos oraciones que se modelan mediante implicaciones: "Si paga con débito, hacemos descuento"; "Pido un taxi, si te demorás"; "Si bebe alcohol, no conduzca"; "Desconectaremos todos los electrodomésticos, si continúa esta tormenta"; "Si una botella contiene ácido, debería colocarse una etiqueta de advertencia"; "Siempre que una botella contenga ácido, debería llevar una etiqueta de advertencia"; "Es necesario colocar una etiqueta de advertencia para las botellas que contienen ácido".

La Tabla 1.6 muestra la *tabla de verdad* de la implicación $(\theta \rightarrow \varphi)$.

Tabla 1.6

θ	φ	$(\theta \rightarrow \varphi)$
0	0	1
0	1	1
1	0	0
1	1	1

Si el antecedente es verdadero, entonces el valor de verdad del condicional es igual al valor de verdad del consecuente y si el antecedente es falso el valor del condicional es verdadero. Son verdaderas las expresiones: "si el semáforo está en rojo, los autos se detienen"; "si 5 es raíz del polinomio p(x) entonces p(5) = 0"; pero las siguientes expresiones son falsas: "si $2^2 = 4$, entonces la suma de los ángulos interiores de un triángulo es 360^o"; "si el General San Martín cruzó los Andes, entonces el General Manuel Belgrano leyó Rayuela, de J. Cortázar". Cuando operamos fórmulas con este conectivo, no es necesario que haya una relación causal entre ellas. Escribimos "$(\theta \rightarrow \varphi)$" tanto si "$\theta$" y "$\varphi$" están relacionadas como causa-efecto, como si no lo están. Veamos lo mencionado en el párrafo anterior en los siguientes ejemplos, en los cuales no existe tal relación causa-efecto y sin embargo le podemos asignar un valor de verdad a la fórmula. En efecto, "Si $5 < 3$ entonces Richard Johnsonbaugh escribió el texto Matemáticas Discretas" es verdadera porque su antecedente $(5 < 3)$ es falso, mientras que "Si $3 < 5$ entonces 8 es múltiplo de 5" es falsa porque "$3 < 5$" es verdadero y "8 es múltiplo de 5" es falso.

En el lenguaje diario, la hipótesis y la conclusión en una implicación están normalmente relacionadas. Muchas veces escuchamos expresiones como "Si apruebo Matemática Discreta y Álgebra y Geometría Analítica, entonces puedo sentarme a estudiar Algoritmos y Estructuras de Datos" o bien "Si tengo dinero entonces saldré de vacaciones", que muestran una relación de causa efecto. Pero expresiones como "Si el árbol tiene tres hojas entonces la pared es azul" o bien "Si miro televisión entonces $2 < 1$" que no tienen sentido en el lenguaje diario, sí lo tienen desde el punto de vista de la lógica.

v) En la fórmula de la forma $(\theta \leftrightarrow \varphi)$, siendo θ y φ fórmulas arbitrarias, el conectivo $\leftrightarrow$ se llama *equivalencia, bicondicional o doble implicación* y la fórmula se lee θ si y sólo si φ, o bien, θ es equivalente a φ. La fórmula de la forma $(\theta \leftrightarrow \varphi)$ es verdadera precisamente cuando "θ" y "φ" tienen el mismo valor de verdad.

Por ejemplo: "$5 < 3$ si y sólo si $2 < 1$" es una fórmula verdadera porque ambas proposiciones atómicas "$5 < 3$" y "$2 < 1$" son falsas, mientras que "$5 > 3$ si y sólo si $2 < 1$" es una proposición falsa porque tienen distintos valores de verdad las proposiciones atómicas "$5 > 3$" (verdadera) y "$2 < 1$" (falsa) que la forman.

La Tabla 1.7 muestra la *tabla de verdad* del bicondicional.

Tabla 1.7

θ	φ	$(\theta \leftrightarrow \varphi)$
0	0	1
0	1	0
1	0	0
1	1	1

Ejemplo 3

a) Construyamos la tabla de verdad para la fórmula $\theta = (q \wedge (\neg r \rightarrow p))$. La última columna de la Tabla 1.8 corresponde a los valores de verdad de θ, mientras que las primeras columnas muestran cómo se va construyendo la tabla teniendo en cuenta cada una de las interpretaciones posibles de sus átomos. Observamos que al contar con 3 variables proposicionales, existen $2^3 = 8$ interpretaciones posibles, es decir, existen 8 posibles renglones que muestran todos los valores de verdad que pueden asumir las 3 variables proposicionales dadas.

Tabla 1.8

p	q	r	$\neg r$	$(\neg r \rightarrow p)$	$\theta = (q \wedge (\neg r \rightarrow p))$
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	0	1	1

b) Los valores de verdad de la fórmula $(((\theta \rightarrow \varphi) \wedge (\varphi \rightarrow \theta)) \leftrightarrow (\theta \leftrightarrow \varphi))$ se muestran en la siguiente tabla.

Tabla 1.9

θ	φ	$(\theta \rightarrow \varphi)$	$(\varphi \rightarrow \theta)$	$((\theta \rightarrow \varphi) \wedge (\varphi \rightarrow \theta))$	$(\theta \leftrightarrow \varphi)$	$(((\theta \rightarrow \varphi) \wedge (\varphi \rightarrow \theta)) \leftrightarrow (\theta \leftrightarrow \varphi))$
0	0	1	1	1	1	1
0	1	1	0	0	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	1

c) La construcción de la Tabla 1.10 nos muestra los valores de verdad de la fórmula: $(((p \vee q) \wedge \neg(p \wedge q)) \leftrightarrow (p \leftrightarrow q))$.

Tabla 1.10

p	q	$(p \vee q)$	$(p \wedge q)$	$\neg(p \wedge q)$	$((p \vee q) \wedge \neg(p \wedge q))$	$(p \leftrightarrow q)$	$(((p \vee q) \wedge \neg(p \wedge q)) \leftrightarrow (p \leftrightarrow q))$
0	0	0	0	1	0	1	0
0	1	1	0	1	1	0	0
1	0	1	0	1	1	0	0
1	1	1	1	0	0	1	0

Antes de iniciar la formalización del trabajo precedente, podemos sintetizar las definiciones de los valores de verdad de las fórmulas estudiadas, siendo θ y φ fórmulas arbitrarias (recordemos además que el símbolo $\veebar$ no es sintáctico):

Tabla 1.11

θ	φ	$\neg\theta$	$\neg\varphi$	$(\theta \wedge \varphi)$	$(\theta \vee \varphi)$	$(\theta \rightarrow \varphi)$	$(\theta \leftrightarrow \varphi)$	$(\theta \veebar \varphi)$
0	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	0	0	1
1	1	0	0	1	1	1	1	0

Actividad 2

Problema Nº1: Para cada oración identifica las variables proposicionales y escribe una fórmula para formalizarla en LPC.

1.1) Si no hay control de natalidad, la población crece. Pero si la población crece aumentará el índice de pobreza.

1.2) La construcción de programas se basa en argumentos racionales y parece razonable aplicar técnicas de comprobación formales.

1.3) Si Andrea gana las Olimpíadas Internacionales de Informática, todos la admirarán y ella ganará una beca para estudiar en la Universidad; pero si no gana, todo su esfuerzo fue en vano.

1.4) Juan sabe o C, o Pascal, o Prolog, y disfruta trabajando con la gente. De otro modo, él no sería un programador destacado.

1.5) Si el costo de las utilidades crece o se niega la requisición de fondos adicionales, entonces compraremos una nueva computadora si y sólo si, podemos mostrar que los recursos de cómputo son, en efecto, insuficientes.

Problema Nº2: Construye una tabla de verdad para cada una de las fórmulas:

2.1) $(p \rightarrow (p \vee q))$

2.2) $(p \rightarrow (q \rightarrow r))$

2.3) $((p \rightarrow q) \leftrightarrow r)$

2.4) $((p \rightarrow q) \wedge (q \rightarrow \neg p))$

2.5) $((p \wedge (p \rightarrow q)) \rightarrow q)$

2.6) $((p \wedge q) \rightarrow p)$

2.7) $(((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r))$

Problema Nº3: Si el valor de $(p \rightarrow q)$ es falso, ¿puedes completar la siguiente tabla?:

p	q	$(p \wedge q)$	$\neg p$	$\neg q$	$(\neg p \vee \neg q)$	$(p \rightarrow \neg q)$	$((p \wedge q) \vee \neg q)$

Problema Nº4: Responde las preguntas de los siguientes ítems y justifica las respuestas.

4.1) El valor de verdad de la fórmula $(p \rightarrow q)$ es falso. ¿Cuál es el valor de verdad de la fórmula $((\neg p \vee \neg q) \rightarrow q)$?

4.2) Sabiendo que exactamente dos de las tres variables proposicionales p, q, r, son verdaderas, ¿cuál es el valor de verdad de la siguiente proposición?

$$((((\neg p \wedge q) \wedge r) \vee ((p \wedge \neg q) \wedge r)) \vee (p \wedge (q \wedge \neg r)))$$

Problema Nº5: Determina todos los valores de verdad, si es posible, para p, q, r, s y t de modo que las siguientes fórmulas sean falsas.

5.1) $(((p \wedge q) \wedge r) \rightarrow (s \vee t))$

5.2) $(((p \wedge q) \wedge r) \rightarrow (\neg r \vee \neg t))$

Problema Nº6: Una fórmula θ de tres variables proposicionales r, s, t, toma el valor 0 cuando s tiene el valor 1 y la variable r no tiene el valor 1. En los demás casos toma el valor 1. Completa la tabla de verdad de la fórmula θ.

Problema Nº7: Sean θ y φ fórmulas. Si llamamos a la fórmula $(\theta \rightarrow \varphi)$ como *implicación directa,* la fórmula $(\varphi \rightarrow \theta)$ se llama *recíproca* de $(\theta \rightarrow \varphi)$; la implicación $(\neg \theta \rightarrow \neg \varphi)$ es su *contraria*, mientras que $(\neg \varphi \rightarrow \neg \theta)$ es su *contra recíproca*. Se dice que las implicaciones recíproca, contraria y contra recíproca son implicaciones asociadas a la directa. Veamos algunos ejemplos:

a) Si la proposición directa es "Si estudio 8 horas diarias entonces apruebo mis exámenes", las implicaciones asociadas son:

Recíproca: "Si apruebo mis exámenes entonces estudio 8 horas diarias".

Contraria: "Si no estudio 8 horas diarias entonces no apruebo mis exámenes".

Contra recíproca: "Si no apruebo mis exámenes entonces no estudio 8 horas diarias".

b) Si la proposición directa es "Si se me hace tarde para ir a la Facultad entonces no tomo la línea XX", las implicaciones asociadas son:

Recíproca: "Si no tomo la línea XX entonces se me hace tarde para ir a la Facultad".

Contraria: "Si no se me hace tarde para ir a la Facultad entonces tomo la línea XX".

Contra recíproca: "Si tomo la línea XX entonces no se me hace tarde para ir a la Facultad".

Para cada caso, escribe las implicaciones asociadas a la dada:

7.1) Si la ruta de Rosario a Bs. As es peligrosa, viajo de día.

7.2) Si los índices de producción y de exportación de Argentina aumentan entonces aumenta el ingreso per cápita de sus habitantes.

Problema Nº8: En cada enunciado identifica las proposiciones atómicas. Escribe en forma simbólica y en lenguaje coloquial las proposiciones recíproca, contraria y contra recíproca de cada una de las implicaciones directas dadas a continuación:

8.1) Si el programa está bien escrito y bien documentado entonces es muy probable que satisfaga las normas de calidad.

8.2) Andrés no es honesto si recibió el telegrama de la empresa.

8.3) Si no estoy equivocado, ella conducía un coche rojo y había un hombre sentado a su lado.

1.4.2 Interpretaciones

Ahora formalizaremos la explicación que dimos en la sección anterior. Un concepto fundamental en semántica es el de *Interpretación*. Cada interpretación define qué fórmulas del lenguaje son falsas y cuáles son verdaderas. Los valores de verdad son dos: falso o verdadero, y pueden ser indicados con 0 o 1; off u on; no o si, F o T, respectivamente. Nosotros hemos usado 0 (falso) y 1 (verdadero). Vamos a definir inicialmente una interpretación (o valuación) para las variables proposicionales y luego haremos lo propio para las restantes fórmulas.

Definición 1.4

Sea Var el conjunto de átomos del lenguaje L.

Una *interpretación* es una función de Var en {0, 1}, que asigna a cada elemento de Var un 0 o un 1.

Una interpretación puede ser explícitamente identificada por el subconjunto de átomos de Var que tienen como imagen 1. Bajo esta representación cada interpretación se identifica con los átomos de Var a los que se ha asignado un 1.

Ejemplo 4

Si Var = {p, q, r}, una interpretación I_1 puede estar dada por I_1= {p}, esto significa que p tiene como imagen 1 mientras que q y r tienen como imagen 0. Otro ejemplo es I_2 = {p, q}, donde p y q tienen como imagen 1 mientras que r tiene como imagen 0, es decir $I_2(p)=1$; $I_2(q)=1$; $I_2(r)=0$; una nueva valuación, es I_3 = {p, q, r} donde todas las variables proposiciones tienen como imagen el 1.

La siguiente tabla muestra estos ejemplos de interpretaciones:

Tabla 1.12

Ejemplos de Interpretaciones	p	q	r
I_1 = {p}	1	0	0
I_2={p, q}	1	1	0
I_3= {p, q, r}	1	1	1

Observaciones:

1. Si el conjunto de átomos tiene 3 elementos, es decir Var = {p, q, r}, como cada uno de ellos puede ser 0 o 1, existirán $2^3 = 8$ interpretaciones: {}(donde todos los átomos tienen como imagen 0, {p}, {q}, {r}, {p, q}, {p, r}, {q, r}, {p, q, r} (donde todos los átomos tienen como imagen 1. De esta manera, cada renglón de la tabla de verdad es una interpretación. La interpretación {} es el primer renglón y los valores de verdad de todas las variables son 0; la interpretación {p} corresponde al renglón donde los valores de verdad de las variables son 1, 0, 0, respectivamente; la interpretación {p, q} corresponde al renglón donde las variables proposicionales p, q, r tienen como imágenes 1, 1, 0 respectivamente.

Si el conjunto de átomos es n, entonces será posible definir 2^n interpretaciones.

2. Usualmente trabajaremos con un conjunto finito de átomos. Cuando demos una fórmula asumiremos que el conjunto de átomos del lenguaje coincide con el conjunto de átomos usado en el ejemplo, salvo que lo indiquemos explícitamente de otra manera.

3. Indistintamente, usaremos los vocablos *interpretación* o *valuación*.

4. El valor de verdad de los átomos no depende del valor de verdad que tengan otros átomos de L, es decir, la fórmula $(p \wedge q)$ es verdadera cuando ambas p y q lo son y no depende del valor de otros átomos r, s, t que puedan formar parte del conjunto Var de L. El valor de verdad (0 o 1) de una fórmula depende completamente de los valores de verdad de los átomos que intervienen y de los conectivos utilizados.

A continuación vamos a extender la definición de interpretación (o valuación) para el caso de las fórmulas definidas en L.

Definición 1.5

Sea I una interpretación. Sea Form el conjunto de fórmulas de L. Una *valuación* bajo una interpretación I, es cualquier función v_I de Form en {0, 1}, que satisface las siguientes reglas:

v.1) $v_I(p_n) = I(p_n)$ para cada variable proposicional. Es decir, cuando la fórmula es una variable proposicional, su valuación coincide con la asignación que le hace la interpretación.

v.2) Siendo φ y ρ fórmulas arbitrarias, se define la valuación de una fórmula θ como sigue:

Tabla 1.13 a)

φ	ρ	$\theta = \neg\varphi$	$\theta = (\varphi \wedge \rho)$	$\theta = (\varphi \vee \rho)$	$\theta = (\varphi \rightarrow \rho)$	$\theta = (\varphi \leftrightarrow \rho)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Dicho de otra manera:

Se define el valor de verdad de una fórmula θ bajo I como una función *valuación* $v_I(\theta)$ de la siguiente manera:

a) Si θ es una variable proposicional entonces el valor de verdad de θ, $v_I(\theta)$, es la imagen (0 o 1) que le corresponde a la variable proposicional bajo la interpretación I.

b) Si θ es de la forma $\neg\varphi$ entonces $v_I(\theta) = 1$ si $v_I(\varphi) = 0$ y $v_I(\theta) = 0$ si $v_I(\varphi) = 1$.

c) Si θ es de la forma $(\varphi \wedge \rho)$ entonces $v_I(\theta) = 1$ si $v_I(\varphi) = 1$ y $v_I(\rho) = 1$ y $v_I(\theta) = 0$ en cualquier otro caso.

d) Si θ es de la forma $(\varphi \vee \rho)$ entonces $v_I(\theta) = 0$ si $v_I(\varphi) = 0$ y $v_I(\rho) = 0$ y $v_I(\theta) = 1$ en cualquier otro caso.

e) Si θ es de la forma $(\varphi \rightarrow \rho)$ entonces $v_I(\theta) = 0$ si $v_I(\varphi) = 1$ y $v_I(\rho) = 0$ y $v_I(\theta) = 1$ en cualquier otro caso.

f) Si θ es de la forma $(\varphi \leftrightarrow \rho)$ entonces $v_I(\theta) = 1$ si $v_I(\varphi) = v_I(\rho)$ y $v_I(\theta) = 0$ en cualquier otro caso.

Ejemplo 5

a) Sea Var = {p, q, r} y J una interpretación dada por J = {q}. La valuación de la fórmula $\theta = (p \wedge (q \rightarrow \neg r))$ bajo esta interpretación es $v_J(\theta) = v_J(p \wedge (q \rightarrow \neg r)) = 0$ porque $v_J(p) = 0$. Sin embargo, bajo I = {p} la valuación de la fórmula es 1 porque $v_I(p) = 1$ y $v_I(q \rightarrow \neg r) = 1$ dado que $v_I(q) = 0$.

b) Veamos las valuaciones de las fórmulas $\theta_1 = (\varphi \rightarrow (\varphi \wedge \rho))$ y $\theta_2 = (\varphi \wedge (\neg\varphi \wedge \rho))$ para cada una de las posibles valuaciones de las fórmulas φ y ρ, esto es:

Tabla 1.13 b)

Valuaciones	φ	ρ	$(\varphi \vee \rho)$	$\theta_1 = (\varphi \rightarrow (\varphi \vee \rho))$	$\neg\varphi$	$(\neg\varphi \wedge \rho)$	$\theta_2 = (\varphi \wedge (\neg\varphi \wedge \rho))$
v_1	0	0	0	1	1	0	0
v_2	0	1	1	1	1	1	0
v_3	1	0	1	1	0	0	0
v_4	1	1	1	1	0	0	0

Podemos notar que independientemente de las valuaciones que tengan φ y ρ, la fórmula θ_1 es siempre verdadera mientras que la fórmula θ_2 es siempre falsa. La tabla sugiere pensar la existencia de fórmulas verdaderas bajo toda interpretación; de fórmulas falsas bajo cualquier interpretación y de fórmulas cuyos valores de verdad dependen de cada interpretación.

Observaciones:

1. Sean φ y ρ fórmulas arbitrarias; la fórmula $\theta = ((\varphi \wedge \neg\rho) \vee (\rho \wedge \neg\varphi))$ será identificada como $(\varphi \veebar \rho)$, es decir como la disyunción excluyente de φ y ρ. Si θ es de la forma $(\varphi \veebar \rho)$ entonces $v_I(\theta) = 0$ si $v_I(\varphi) = v_I(\rho)$ y $v_I(\theta) = 1$ en cualquier otro caso. A partir de este momento vamos a incorporar y usar el símbolo $\veebar$ (o excluyente) como un conectivo binario aunque sepamos que no es un símbolo del lenguaje de la LPC.

2. Existen otras maneras que permiten interpretar las fórmulas. Veamos una de ellas en la siguiente definición. Aunque no lo probemos, 1.5 y 1.6 son definiciones equivalentes.

Definición 1.6

Sea I una interpretación y Form un conjunto de fórmulas. Interpretamos inicialmente los conectivos unarios ($\neg$) y binarios ($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\underline{\vee}$) por las tablas:

$\neg$	
0	1
1	0

$\wedge$	**0**	**1**
0	0	0
1	0	1

$\vee$	**0**	**1**
0	0	1
1	1	1

$\rightarrow$	**0**	**1**
0	1	1
1	0	1

$\leftrightarrow$	**0**	**1**
0	1	0
1	0	1

$\underline{\vee}$	**0**	**1**
0	0	1
1	1	0

Luego toda *función* v_I de Form en {0, 1} que satisface las siguientes reglas es una *valuación*:

v.1) $v_I(p_n) = I(p_n)$ para cada variable proposicional. Es decir, cuando la fórmula es una variable proposicional, su imagen coincide con la asignación que le hace la interpretación.

v.2) Si θ es de la forma $\neg\varphi$ entonces $v_I(\neg\varphi) = \neg v_I(\varphi)$.

v.3) Si θ es de la forma $(\varphi * \rho)$, donde $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \underline{\vee}\}$, luego $v_I(\varphi * \rho) = v_I(\varphi) * v_I(\rho)$.

Ejemplo 6

Sea Var = {p, q, r} e I = {p}, una interpretación.

Entonces la valuación de la fórmula $\theta = (p \wedge (q \rightarrow \neg r))$ bajo esta interpretación se puede obtener de acuerdo a los siguientes pasos: $v_I(\theta) = v_I(p \wedge (q \rightarrow \neg r)) = v_I(p) \wedge v_I(q \rightarrow \neg r) = v_I(p) \wedge (v_I(q) \rightarrow v_I(\neg r)) = v_I(p) \wedge (v_I(q) \rightarrow \neg v_I(r)) = I(p) \wedge (I(q) \rightarrow \neg I(r)) = 1$.

Ejemplo 7

Sea Var = $\{p_1, p_2, p_3\}$.

a) Sea θ la fórmula $\theta = (p_1 \rightarrow (p_2 \vee \neg p_3))$. Aplicando las reglas v1), v2) o v3) de la definición precedente obtenemos que $v(\theta) = v(p_1) \rightarrow v(p_2 \vee \neg p_3) = v(p_1) \rightarrow (v(p_2) \vee \neg v(p_3))$. Para completar la valuación de la fórmula, necesitamos conocer la interpretación (valuación) de las fórmulas atómicas que la componen. Por ejemplo, siendo $v(p_1)=0$, $v(p_2)=1$ y $v(p_3)=1$ obtenemos que $v(\theta)=1$. La fórmula es verdadera bajo esa interpretación. Esta interpretación no es la única que hace verdadera a la fórmula. Toda interpretación donde p_1 asuma el valor 0 hará que la valuación de θ sea verdadera. Diremos que la fórmula es satisfacible cuando una valuación la hace verdadera.

b) Sea ahora la fórmula $\varphi = \neg(p_1 \vee \neg p_3)$. Bajo la misma valuación definida en a) se verifica que $v(\varphi) = \neg v(p_1 \vee \neg p_3) = \neg(v(p_1) \vee \neg v(p_3)) = 1$.

c) Sea S un subconjunto de Form formado por las fórmulas dadas en a) y b). $S = \{\theta, \varphi\}$; existe una valuación (definida en a)) tales que $v(p_1) = 0$, $v(p_2) = 1$ y $v(p_3) = 1$ que hace verdaderas a todas las fórmulas de S, luego diremos que el conjunto de fórmulas *S es satisfacible*.

Estos ejemplos nos llevan a las siguientes definiciones.

Definición 1.7

Sea θ una fórmula y sea I una interpretación. Se dice que θ es verdadera bajo I si su valor de verdad es 1. Cuando θ es verdadera bajo I, se dice que I *satisface* a θ; o que θ es *satisfacible* por I. Escribiremos que I satisface a θ como $I \models \theta$. Se dice que θ es falsa bajo I si su valor de verdad es 0. Cuando θ es falsa bajo I, se dice que I *no satisface* a θ.

Actividad 3

Problema Nº1: Sea Var = {p, q}. Decide cuál es la valuación de la fórmula bajo cada interpretación dada:

1.1) $\theta = ((p \rightarrow q) \wedge \neg q)$; $I_1 = \{p, q\}$

1.2) $\theta = ((p \rightarrow q) \wedge \neg q)$; $I_2 = \{p\}$

1.3) $\theta = ((p \rightarrow q) \wedge \neg q)$; $I_3 = \{q\}$

1.4) $\theta = ((p \rightarrow q) \wedge \neg q)$; $I_4 = \{\}$

Problema Nº2: Sea Var = {p, q, r}. Decide, para cada caso si $I \models \theta$.

2.1) $\theta = (((p \rightarrow q) \wedge \neg q) \rightarrow r)$; $I_1 = \{p, q\}$

2.2) $\theta = (r \rightarrow ((p \rightarrow q) \wedge \neg q))$; $I_2 = \{p\}$

2.3) $\theta = ((r \rightarrow \neg r) \vee ((p \rightarrow q) \wedge \neg q))$; $I_3 = \{q\}$

2.4) $\theta = (((p \rightarrow q) \wedge \neg q) \leftrightarrow r)$; $I_4 = \{\}$

1.4.3 Modelos. Fórmulas satisfacibles. Fórmulas tautológicas

Usualmente el valor de verdad de una fórmula depende de las interpretaciones; bajo algunas interpretaciones la fórmula es verdadera; bajo otras, la fórmula es falsa. Si el valor de verdad de una fórmula es 1 bajo alguna interpretación particular I, se dice que I es *modelo* para la fórmula.

Definición 1.8

Sea θ una fórmula y sea I una interpretación. I es un *modelo* para θ si I *satisface* a θ; se dice también que θ tiene a I como *modelo*.

Definición 1.9

Una fórmula θ se llama *tautología* si todas las interpretaciones son modelos para la fórmula. Se utiliza el símbolo T_0 para denotar cualquier tautología. Cuando una fórmula es tautología vamos a anotar $\models \theta$.

Una fórmula se llama *contradicción*, *insatisfacible* o *inconsistente* si es falsa para todas las interpretaciones posibles. Se utiliza F_0 para designar toda contradicción. Una contradicción no tiene interpretaciones que sean modelos.

Una proposición lógica que no es una *tautología* ni es una *contradicción* se denomina *contingencia*. Una contingencia es satisfacible pero no es tautología. Esto significa que existirá alguna interpretación I para la cual $v_I(\theta) = 1$ y alguna interpretación J donde $v_J(\theta) = 0$.

El método más directo, hasta el momento, para determinar si una fórmula es una tautología (contradicción) es mediante su tabla de verdad. Si todas sus filas producen un "1" ("0") en la última columna se trata de una tautología (contradicción); si algunas filas producen "1" mientras que otras producen "0" se trata de una contingencia.

Ejemplo 8

a) Leamos la siguiente tabla:

Tabla 1.14

p	q	$\neg p$	$\theta = \neg(p \wedge q)$	$\varphi = (\neg(p \wedge q) \vee q)$	$\rho = (p \wedge \neg p)$	$\omega = (p \vee \neg p)$
0	0	1	1	1	0	1
0	1	1	1	1	0	1
1	0	0	1	1	0	1
1	1	0	0	1	0	1
			Contingencia	Tautología	Contradicción	Tautología

Pueden enunciarse algunas conclusiones:

i) La interpretación I = {} (primer renglón) es un modelo para θ, φ, ω. Las fórmulas son satisfacibles porque esta interpretación las hace verdaderas.

ii) El conjunto de fórmulas $S_1 = \{\theta, \varphi, \omega\}$ es satisfacible bajo I = {} porque todas las fórmulas de S_1 tienen a I como modelo.

iii) Todas las interpretaciones para Var = {p, q} hacen verdadera a φ, luego φ es tautología. Estas mismas interpretaciones hacen falsa a ρ, luego ρ es una contradicción.

iv) El conjunto de fórmulas $S_2 = \{\theta, \varphi, \rho, \omega\}$ es insatisfacible porque no es posible encontrar una valuación (interpretación) que sea un modelo para todas las fórmulas de S_2.

b) La fórmula $(p \vee \neg p)$ es tautología porque toda interpretación es modelo para la fórmula. Bajo cualquier interpretación, o p o $\neg p$ es verdadera.

Todas las interpretaciones $I_1 = \{\}$ e $I_2 = \{p\}$ son modelos para la fórmula. Es decir:

p	$\neg p$	$\omega = (p \vee \neg p)$
0	1	1
1	0	1

c) La fórmula $(((\theta \rightarrow \varphi) \wedge (\varphi \rightarrow \theta)) \leftrightarrow (\theta \leftrightarrow \varphi))$ es tautología.

d) La fórmula $(((p \vee q) \wedge \neg(p \wedge q)) \leftrightarrow (p \leftrightarrow q))$ es insatisfacible.

Definición 1.10

Sea S un conjunto de fórmulas, $S \subseteq$ Form y v una valuación. Diremos que una valuación v satisface un conjunto S de fórmulas, si $v(\varphi)=1$ para toda $\varphi \in S$. Es decir, la valuación es modelo para S si es modelo para todas las fórmulas de S.

Ejemplo 9

Sea S = {(p→q), (q→r)}. Busquemos una valuación que satisfaga a S.

Tabla 1.15

p	q	r	(p → q)	(q → r)
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0

La valuación I(p) = 0; I(q) = 0 y I(r) = 0, es decir I = {} satisface al conjunto S porque toda fórmula de S es verdadera para esta interpretación. El hecho de encontrar un renglón de la tabla donde todas las fórmulas de S sean verdaderas nos garantiza que el conjunto de fórmulas S es satisfacible. La interpretación I = {} es modelo para todas las fórmulas de S. Esta interpretación no es única. La interpretación J = {r} es también modelo para las fórmulas de S.

Definición 1.11

Un conjunto de fórmulas se dice satisfacible si existe una valuación que satisfaga a todas las fórmulas del conjunto e insatisfacible en caso contrario.

Ejemplo 10

a) El conjunto vacío es satisfacible; toda valuación satisface al conjunto vacío.

b) El conjunto {(p∧q), ¬(p∧q)} es insatisfacible. También lo es {p, (p→q), ¬p}, porque no existe una valuación que satisfaga simultáneamente a p y ¬p.

c) Sea S = {¬(p → q); (p ∧ ¬q); (r → (p → q)); (¬r ∧ p)}. S es satisfacible bajo la interpretación I = {p}. Esta interpretación es la única que hace verdaderas simultáneamente a todas las fórmulas de S. En efecto, como v(¬r ∧ p) = 1 entonces v(p) = 1 y v(r) = 0; para que v(¬(p → q)) = 1 entonces v(p → q)= 0 y como v(p) = 1, luego v(q) = 0.

Para esta interpretación I = {p} las restantes fórmulas de S son verdaderas.

Ejemplo 11

Sea S ⊆ Form, S = {p, (q∨r), (q→r)} y sean I_1 = {p, r}, I_2 = {p, q, r}, I_3 = {p, q} interpretaciones definidas en Var = {p, q, r}. Las interpretaciones I_1 e I_2 satisfacen las fórmulas de S, luego son modelos de S. La interpretación I_3 hace falsa a la fórmula (q→r), luego no es un modelo para S.

S es un conjunto satisfacible de fórmulas porque al menos una interpretación hace a todas sus fórmulas verdaderas.

La tabla 1.16 ayuda a leer lo expresado.

Tabla 1.16

p	q	r	$(q \vee r)$	$(q \rightarrow r)$
1	0	1	1	1
1	1	1	1	1
1	1	0	1	0

De lo dicho anteriormente, resulta sencillo verificar que:

Propiedad 1.1

Un conjunto finito y no vacío de fórmulas es satisfacible si y sólo si la conjunción de todas las fórmulas del conjunto es satisfacible.

Demostración:

La propiedad brinda un procedimiento efectivo para determinar si un conjunto de fórmulas es satisfacible o no: hacemos la conjunción de las fórmulas del conjunto y luego podemos ir construyendo la tabla de verdad de esta conjunción hasta obtener una fila que en la última columna tenga un 1. Si existe tal fila el conjunto de fórmulas es satisfacible. Si en todas las filas de la conjunción (última columna) figura sólo el 0 el conjunto es insatisfacible.

Ejemplo 12

El conjunto $S = \{(p \rightarrow q), (\neg p \rightarrow q)\}$ es satisfacible.

Por ejemplo, las interpretaciones $I_1 = \{q\}$ e $I_2 = \{p, q\}$ satisfacen a S ya que ambas satisfacen la conjunción da las fórmulas dadas, es decir $((p \rightarrow q) \wedge (\neg p \rightarrow q))$ es verdadera bajo I_1 e I_2.

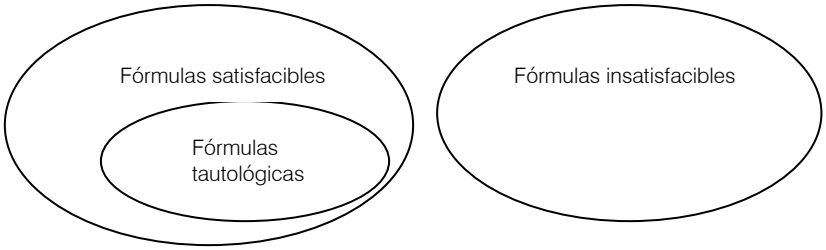
Tabla 1.17

p	q	$p \rightarrow q$	$\neg p \rightarrow q$	$(p \rightarrow q) \wedge (\neg p \rightarrow q)$
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	1	1	1

Si bien la construcción de tablas de verdad es un procedimiento aceptable para determinar si una fórmula o conjunto de fórmulas es o no satisfacible porque se hace en un número finito de pasos, este número puede ser muy grande y el método impracticable. Por ello, más adelante y utilizando estructuras de árboles describiremos otro procedimiento para determinar si un conjunto finito y no vacío de fórmulas es o no es satisfacible; tal procedimiento puede generalizarse para el cálculo de predicados, lo que no ocurre con las tablas de verdad.

Concluyendo:

Una fórmula es o *satisfacible o insatisfacible*. Un subconjunto de las fórmulas satisfacibles son aquellas que se llaman tautologías, que siempre son verdaderas para cualquier interpretación I.



TEOREMA 1.1

Una fórmula es tautológica (o válida) si y sólo si su negación es insatisfacible.

Demostración:

Si θ es tautológica, toda interpretación es un modelo para θ, luego $v_I(\theta) = 1$ para toda I. Por lo tanto $0 = \neg v_I(\theta) = v_I(\neg\theta)$ para toda I. Luego, no existe una interpretación que sea un modelo para $\neg\theta$, de donde $\neg\theta$ es insatisfacible.

Actividad 4

Problema Nº1: En cada enunciado, identifica las proposiciones primitivas, escríbelo en forma simbólica, construye la tabla de verdad para cada fórmula e indica si tal enunciado es una tautología, una contradicción o una contingencia.

1.1) Si Juan estuvo ayer en el partido, necesita dormir. Juan no necesita dormir. Por consiguiente no fue al partido.

1.2) Si el programa para resolver el problema es eficaz, entonces no tiene error. El programa es eficaz. Por lo tanto, no tiene error.

1.3) Si los programas tienen errores lógicos, no funcionan correctamente. Si no funcionan correctamente entonces no son eficaces. Por lo tanto, los programas que tienen errores lógicos no son eficaces.

1.4) Si Agustín no cumple con el horario de trabajo, será despedido. Agustín fue despedido. Por tanto no cumplió con el horario de trabajo.

1.5) Si en el mes de abril estamos en otoño entonces se caen las hojas, pero abril pertenece al otoño y no se caen las hojas.

Problema Nº2: Sean p, q variables proposiciones.

2.1) Muestra que la fórmula $((p \vee q) \wedge (\neg p \wedge \neg q))$ es una contradicción.

2.2) Muestra que $(\neg(p \rightarrow q) \leftrightarrow (p \wedge \neg q))$ es una tautología.

Problema Nº3: Justifica que las siguientes fórmulas son tautologías:

3.1) $(\neg p \vee \neg\neg p)$, siendo p una variable proposicional.

3.2) $((p \to q) \vee \neg(p \to q))$, siendo p, q variables proposicionales.

Problema Nº4: Dados los siguientes conjuntos de fórmulas, decide cuáles son satisfacibles y cuáles insatisfacibles. Justifica la respuesta.

4.1) $S = \{\neg p, (p \to q), (q \to r), r\}$

4.2) $S = \{\neg q, (p \to q)\}$

4.3) $S = \{(p \to (q \to r)), (\neg p \to \neg r), (p \to \neg q)\}$

4.4) $S = \{(p \leftrightarrow q), (\neg p \to \neg q), (p \to \neg q)\}$

Problema Nº5: Decide si las siguientes afirmaciones son falsas o verdaderas:

5.1) Si una fbf no es tautología, su negación lo es.

5.2) Si una fbf no es satisfacible, su negación lo es.

5.3) Si un conjunto de fbfs es satisfacible, cada variable proposicional lo es.

Problema Nº6: Demuestra que las siguientes fbfs son tautologías:

6.1) $(p \to (q \to p))$

6.2) $((p \to (q \to r)) \to ((p \to q) \to (p \to r)))$

6.3) $((p \to q) \leftrightarrow (\neg q \to \neg p))$

Problema Nº7: Dadas las siguientes fórmulas o conjuntos de fórmulas del cálculo proposicional, decide si son o no son satisfacibles. Si son satisfacibles, da una valuación que las satisfaga.

7.1) $((p_0 \to (p_2 \to p_0)) \to \neg(p_1 \vee \neg p_0))$

7.2) $((p_1 \to (p_1 \wedge p_2)) \vee ((p_3 \wedge \neg p_3) \vee (p_0 \to p_4)))$

7.3) $\{(p \wedge q), (\neg p \wedge q)\}$

7.4) $\{(p \wedge q), (\neg p \vee q)\}$

7.5) $\{(p \to q), (p \vee q), \neg q\}$

7.6) $\{(p \to q), (q \to r), (r \to s), (p \to s)\}$

7.7) $\{(p \to q), (q \to r), (r \to s), (p \wedge \neg s)\}$

7.8) $\{(p \vee q), (p \vee (q \wedge r)), (p \to \neg r)\}$

7.9) $\{(p \to q), ((p \wedge q) \to r), (q \to \neg p)\}$

Problema Nº8: Decide si las siguientes fórmulas son tautologías, contradicciones o contingencias. Para las contingencias halla una valuación que las satisfaga y otra que no las satisfaga.

8.1) $(\neg(p_0 \vee p_1) \to ((p_2 \wedge p_0) \vee (p_1 \to p_2)))$

8.2) $\neg((p_1 \to p_3) \to ((p_2 \to p_3) \to ((p_1 \vee p_2) \to p_3)))$

8.3) $(((p_1 \to p_2) \to p_1) \to p_1)$

8.4) $((\neg\neg\neg(p_1 \wedge p_2) \vee p_3) \rightarrow p_4)$

8.5) $((p_1 \vee \neg p_1) \vee (p_2 \wedge \neg p_2))$

1.4.4 La implicación y la consecuencia lógicas

Un importante concepto que vamos a explicitar es el de consecuencia lógica. Una fórmula φ es consecuencia lógica de un conjunto S de fórmulas si la verdad de todas las fórmulas de S implican la verdad de φ. Seguidamente, vamos a formalizar estos comentarios.

Definición 1.12

Sea S un conjunto de fórmulas y φ una fórmula.

Se dice que φ es una *consecuencia lógica* de S si toda interpretación que es un modelo de S es también un modelo para φ. Podemos escribir esto como $S \models \varphi$. A veces diremos que S *implica lógicamen*te a φ o simplemente que S *implica* a φ.

Dicho de otra manera, φ es *consecuencia lógica* de S si toda interpretación I que hace verdadera a todas las fórmulas de S, también hace verdadera a φ.

Es decir, si $v(\theta) = 1$ para toda $\theta \in S$, entonces $v(\varphi) = 1$.

Indicaremos Con $S = \{\varphi \in Form / S \models \varphi\}$ al conjunto de las consecuencias de S. El conjunto Con S puede ser infinito.

En particular si $S = \{\theta\}$ podemos escribir $\theta \models \varphi$ o bien $\theta \Rightarrow \varphi$ y leemos θ implica (lógicamente) a φ, en cualquier caso.

Ejemplo 13

a) Sea $S = \{p_1, p_2, p_3\}$.

Luego $((p_1 \wedge p_2) \wedge p_3) \in$ Con S; $(p_1 \wedge p_2) \in$ Con S; $(p_3 \rightarrow p_2) \in$ Con S; $(p_1 \wedge \neg p_2) \notin$ Con S; $(p_2 \rightarrow \neg p_3) \notin$ Con S.

b) Sea $S = \{(p \wedge q), (p \rightarrow r)\}$. Las fórmulas p, q, r $\in$ Con S.

c) Se verifica que $(p \wedge (p \rightarrow q)) \models q$. En efecto, $v(p) = 1$; como además $v(p \rightarrow q) = 1$, entonces necesariamente $v(q) = 1$; por lo que toda interpretación que hace verdaderas a las fórmulas p y $(p \rightarrow q)$ también hace verdadera a q.

Escribimos también $(p \wedge (p \rightarrow q)) \Rightarrow q$.

d) $p \models p$; $p \models (p \wedge p)$; $p \models (p \wedge (p \vee q))$; $p \models (p \vee (p \wedge q))$; $p \models (p \wedge (q \vee \neg q))$, etc. Es decir $\{ p; (p \wedge p); (p \wedge (p \vee q); (p \vee (p \wedge q); (p \wedge (q \vee \neg q))\}$ está incluido en Con $\{p\}$.

Observaciones:

1. Si $S = \{\}$, Con S es el conjunto de las tautologías porque toda valuación satisface al conjunto vacío.

2. El símbolo implicación "$\rightarrow$" es sintáctico. El valor de verdad de la fórmula "$(\theta \rightarrow \varphi)$" depende de una *particular* interpretación.

3. La consecuencia lógica o implicación lógica "$\models$" es un concepto semántico. Está definido en término de todas las interpretaciones: "$\theta \models \varphi$" si toda interpretación que es modelo para θ también es modelo para φ.

4. Incorporamos el símbolo "$\Rightarrow$". La expresión "$\theta \Rightarrow \varphi$" que se lee "si θ entonces φ" o "θ implica φ" es usada en la prueba de teoremas y significa que la verdad de θ es suficiente para la verdad de φ o bien que la verdad de φ es necesaria para θ.

Definición 1.13

Sean S_1 y S_2 dos conjuntos de fórmulas. Luego S_2 es *consecuencia lógica* de S_1 y escribimos $S_1 \models S_2$ si $S_1 \models \varphi$ para toda fórmula $\varphi \in S_2$. También podemos decir que S_1 implica (lógicamente) a S_2.

Ejemplo 14

a) Sean $S_1 = \{(p \rightarrow q), (q \rightarrow r)\}$ y $S_2 = \{(p \rightarrow r)\}$. Las interpretaciones donde las fórmulas de S_1 son verdaderas son $I_1 = \{p, q, r\}$; $I_2 = \{q, r\}$; $I_3 = \{r\}$; $I_4 = \{\}$ y todas ellas son modelos para S_2. Luego $S_1 \models S_2$.

b) Sean $S_1 = \{(p \vee q), (q \rightarrow t), (p \rightarrow r)\}$ y $S_2 = \{(r \vee t)\}$. Dejamos a cargo del lector verificar que $S_1 \models S_2$. Nosotros lo haremos más adelante.

TEOREMA 1.2 (Teorema de la Deducción)

Sea S un conjunto de fórmulas y θ y φ fórmulas arbitrarias.

Luego $S \cup \{\theta\} \models \varphi$ si y sólo si $S \models (\theta \rightarrow \varphi)$.

También podemos enunciarlo así: $\varphi \in Con(S \cup \{\theta\})$ si y sólo si $(\theta \rightarrow \varphi) \in Con\ S$.

Demostración:

Sea $\varphi \in Con(S \cup \{\theta\})$ entonces toda valuación v que satisface a $S \cup \{\theta\}$ también satisface a φ. Sea v una valuación que satisface a S y que no satisface a $(\theta \rightarrow \varphi)$, entonces $v(\theta \rightarrow \varphi) = 0$ de donde $v(\varphi) = 0$ y $v(\theta) = 1$. Esta valuación v satisface a $S \cup \{\theta\}$ pero $v(\varphi) = 0$, entonces $\varphi \notin Con(S \cup \{\varphi\})$, lo que es una contradicción con lo dicho al comienzo de nuestra demostración.

Suponemos ahora que $(\theta \rightarrow \varphi) \in Con\ S$. Es decir, toda valuación que satisface a S también satisface a $(\theta \rightarrow \varphi)$. Sea v una valuación que satisface a $S \cup \{\theta\}$. Luego satisface a todas las fórmulas de S y también satisface a θ, es decir, $v(\theta) = 1$. Probaremos que $v(\varphi) = 1$.

Como $v(\neg\theta) = 0$, luego $v(\varphi) = v(\varphi) \vee v(\neg\theta) = v(\theta \rightarrow \varphi) = 1$. La fórmula $(\theta \rightarrow \varphi) \in Con\ S$. Entonces $\varphi \in Con\ (S \cup \{\theta\})$.

Observación:

Como corolario del teorema de la deducción, surge que para todo par de fórmulas θ, φ se tiene que $\{\theta\} \models \varphi$ si y sólo si $\models (\theta \rightarrow \varphi)$, es decir, si y sólo si $(\theta \rightarrow \varphi)$ es una tautología. O bien, $\varphi \in Con\ \{\theta\}$ si y sólo si $(\theta \rightarrow \varphi) \in Con\ \{\}$ si y sólo si $(\theta \rightarrow \varphi)$ es tautología.

Ejemplo 15

Sean θ, φ, ρ fórmulas. Veamos que $((\theta \to \varphi) \to ((\varphi \to \rho) \to (\theta \to \rho)))$ es una tautología.

Para demostrar que $\models ((\theta \to \varphi) \to ((\varphi \to \rho) \to (\theta \to \rho)))$, vamos a escribir una estructura deductiva equivalente, utilizando el teorema de la deducción. Son equivalentes:

$\models ((\theta \to \varphi) \to ((\varphi \to \rho) \to (\theta \to \rho)))$

$(\theta \to \varphi) \models ((\varphi \to \rho) \to (\theta \to \rho))$

$\{(\theta \to \varphi), (\varphi \to \rho)\} \models (\theta \to \rho)$

$\{(\theta \to \varphi), (\varphi \to \rho), \theta\} \models \rho$

Mostraremos esta última implicación lógica. Sea v cualquier valuación que satisfaga al conjunto, luego $v(\theta) = 1$; para que $v(\theta \to \varphi) = 1$ debe ser $v(\varphi) = 1$. Tomando ahora $v(\varphi) = 1$ y sabiendo que $v(\varphi \to \rho) = 1$, necesariamente $v(\rho) = 1$.

Ejemplo 16

La Tabla 1.18 muestra que $((p \to q) \wedge (\neg p \to q)) \models q$. Observamos que toda interpretación que es modelo para $((p \to q) \wedge (\neg p \to q))$ también es modelo para q.

O bien $((p \to q) \wedge (\neg p \to q)) \models q$ si y sólo si $\models (((p \to q) \wedge (\neg p \to q)) \to q)$.

Tabla 1.18

p	q	$p \to q$	$\neg p \to q$	$((p \to q) \wedge (\neg p \to q))$	$(((p \to q) \wedge (\neg p \to q)) \to q)$
0	0	1	0	0	1
0	1	1	1	1	1
1	0	0	1	0	1
1	1	1	1	1	1

Remarcamos que en los casos donde θ implica lógicamente a φ decimos que "θ *es condición suficiente para* φ" o que "φ *es condición necesaria para* θ" o bien *"*θ *sólo si* φ*"* o *"*φ*, si* θ*"*.

Ejemplo 17

a) En la proposición "Si un triángulo es equilátero entonces es isósceles", la proposición "es triángulo equilátero" es suficiente para la proposición "es triángulo isósceles".

b) En la propiedad de los números naturales que se expresa: "Para que la suma de varios números impares sea un número par debe existir un número par de sumandos impares", la proposición "un número par de sumandos impares" es condición suficiente para que "el resultado de la suma sea par".

c) Y refiriéndonos a ejemplos sobre fórmulas y conectivos lógicos, la proposición "$(p \wedge q)$ es verdadera" es suficiente para la proposición "$(p \vee q)$ es verdadera"; análogamente, la proposición "$(\theta \to \varphi)$ es falsa" es condición suficiente para la proposición "θ es verdadera y φ es falsa".

d) Sean a, b números reales, "$a = b$" es condición suficiente para "$a^2 = b^2$".
Pero "$a^2 = b^2$" no es condición suficiente para "$a = b$".

TEOREMA 1.3

Para todo conjunto de fórmulas S, $S \subseteq$ Form y para toda fórmula θ, se tiene que $S \models \theta$ si y sólo si $S \cup \{\neg\theta\}$ es insatisfacible.

Demostración:

$\theta \in$ Con S si y sólo si toda valuación que satisface a S también satisface a θ si y sólo si toda valuación que satisface a S verifica que $\neg v(\theta) = v(\neg\theta) = 0$ si y sólo si toda valuación que satisface a S no satisface a $\neg\theta$, es decir, $S \cup \{\neg\theta\}$ es insatisfacible.

Observación:

Tomando S = {} en el teorema anterior, resulta que una fórmula θ es una tautología si y sólo si $\neg\theta$ es insatisfacible (como ya hemos desarrollado).

1.4.5 La equivalencia lógica

Definición 1.14

Dos fórmulas θ y φ son *lógicamente equivalentes* si tienen los mismos modelos, es decir, si para toda interpretación I, los valores de verdad coinciden; es decir $v_I(\theta) = v_I(\varphi)$ para toda interpretación.

También podemos decir que dos fórmulas θ y φ son *lógicamente equivalentes* si y sólo se verifican $\theta \models \varphi$ y $\varphi \models \theta$. Se expresa $\theta \Leftrightarrow \varphi$ o bien $\theta \equiv \varphi$ y se lee θ es equivalente a φ o bien θ si y sólo si φ.

Similarmente dos conjuntos de fórmulas S_1 y S_2 se dicen *lógicamente equivalentes* si y sólo si $S_1 \models S_2$ y además $S_2 \models S_1$.

La Tabla 1.19 muestra los valores de verdad para "$(\neg p \vee q)$" y "$(p \rightarrow q)$". Observamos que las fórmulas "$(\neg p \vee q)$" y "$(p \rightarrow q)$" tienen exactamente los mismos valores de verdad. Luego son equivalentes y escribimos $(\neg p \vee q) \Leftrightarrow (p \rightarrow q)$.

Tabla 1.19

p	q	$\neg p$	$(\neg p \vee q)$	$(p \rightarrow q)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Veamos una relación entre la equivalencia lógica y las tautologías.

TEOREMA 1.4

Si θ y φ son fórmulas arbitrarias, entonces $\theta \Leftrightarrow \varphi$ si y sólo si $\models (\theta \leftrightarrow \varphi)$.

Demostración:

$\theta \Leftrightarrow \varphi$ si y sólo si θ y φ tienen los mismos modelos si y sólo si

Toda interpretación es modelo para θ y φ o es modelo para $\neg\theta$ y $\neg\varphi$ si y sólo si

Toda interpretación es modelo para $(\theta \leftrightarrow \varphi)$ si y sólo si $\models(\theta\leftrightarrow\varphi)$.

Observaciones:

1. El símbolo "$\leftrightarrow$" es sintáctico. El valor de verdad de la fórmula "$(\theta\leftrightarrow\varphi)$" depende de una *particular* interpretación.

2. La equivalencia lógica es un concepto semántico. Está definido en término de todas las interpretaciones: "$\theta \models \varphi$" y "$\varphi \models \theta$", es decir, toda interpretación es modelo para θ si y sólo si es modelo para φ.

3. Incorporamos el símbolo "$\Leftrightarrow$". La expresión "$\theta \Leftrightarrow \varphi$" que se lee "$\theta$ es equivalente φ" o "θ si y sólo si φ" es usada en la prueba de teoremas y significa que la verdad de θ es suficiente y necesaria para la verdad de φ o bien que la verdad de φ es suficiente y necesaria para θ.

Ejemplo 18

a) La proposición p es lógicamente equivalente a p porque $\models(p \leftrightarrow p)$.

b) La proposición p es lógicamente equivalente a $\neg\neg p$ porque $\models(p \leftrightarrow \neg\neg p)$.

c) Los conjuntos de fórmulas $S_1 = \{p, \neg q, (p \vee r)\}$ y $S_2 = \{(r \vee p), (\neg r \vee \neg q), p, (p\rightarrow\neg q)\}$ son equivalentes. En efecto, veamos que tienen los mismos modelos. Sea I una interpretación que satisface a la conjunción de fórmulas de S_1, luego $I(p) = 1$; $I(q) = 0$ y el valor de r puede ser 0 o 1. Luego S_1 tiene dos interpretaciones $I_1 = \{p\}$ e $I_2 = \{p, r\}$ que son modelos para todas sus fórmulas. Dejamos a cargo del lector verificar que ambas interpretaciones son los únicos modelos que tiene S_2. Otra forma de hacerlo es construyendo la tabla de verdad.

Tabla 1.20

p	q	r	¬q	¬r	(p∨r)	(¬r∨¬q)	(p→¬q)	((p∧¬q)∧(p∨r))	((((r∨p)∧(¬r∨¬q))∧p)∧(p→¬q))
0	0	0	1	1	0	1	1	0	0
0	0	1	1	0	1	1	1	0	0
0	1	0	0	1	0	1	1	0	0
0	1	1	0	0	1	0	1	0	0
1	0	0	1	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	1	0	0	1	1	1	0	0	0
1	1	1	0	0	1	0	0	0	0

Las últimas dos columnas muestran que las conjunciones de las fórmulas de S_1 y de S_2,

tienen los mismos valores de verdad y en consecuencia las interpretaciones que satisfacen a las fórmulas de S_1, esto es $I_1 = \{p\}$ y $I_2 = \{p, r\}$ también satisfacen a S_2 y viceversa.

Ejemplo 19

Veamos que se verifica esta equivalencia lógica: $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$. Lo que significa demostrar que $\models (\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q))$.

Tabla 1.21

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$(\neg p \wedge \neg q)$	$(\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q))$
0	0	0	1	1	1	1	1
0	1	1	0	1	0	0	1
1	0	1	0	0	1	0	1
1	1	1	0	0	0	0	1

A lo largo del curso usaremos numerosas equivalencias lógicas. A continuación exponemos una lista de fórmulas que son lógicamente equivalentes. Son sencillas pero muy importantes y es conveniente verificarlas. Sean θ, φ y ρ fórmulas; se verifican:

Tabla 1.22

1	$\neg\neg\theta \Leftrightarrow \theta$	Ley de la doble negación o Ley de Involución
2	$\neg(\theta \vee \varphi) \Leftrightarrow (\neg\theta \wedge \neg\varphi)$ $\neg(\theta \wedge \varphi) \Leftrightarrow (\neg\theta \vee \neg\varphi)$	Leyes de De Morgan
3	$(\theta \vee \varphi) \Leftrightarrow (\varphi \vee \theta)$ $(\theta \wedge \varphi) \Leftrightarrow (\varphi \wedge \theta)$	Leyes conmutativas
4	$(\theta \vee (\varphi \vee \rho)) \Leftrightarrow ((\theta \vee \varphi) \vee \rho)$ $(\theta \wedge (\varphi \wedge \rho)) \Leftrightarrow ((\theta \wedge \varphi) \wedge \rho)$	Leyes asociativas
5	$(\theta \vee (\varphi \wedge \rho)) \Leftrightarrow ((\theta \vee \varphi) \wedge (\theta \vee \rho))$ $(\theta \wedge (\varphi \vee \rho)) \Leftrightarrow ((\theta \wedge \varphi) \vee (\theta \wedge \rho))$	Leyes distributivas
6	$(\theta \vee \theta) \Leftrightarrow \theta$ $(\theta \wedge \theta) \Leftrightarrow \theta$	Leyes idempotentes
7	$(\theta \vee F_0) \Leftrightarrow \theta$ $(\theta \wedge T_0) \Leftrightarrow \theta$	Leyes de identidad
8	$(\theta \vee \neg\theta) \Leftrightarrow T_0$ $(\theta \wedge \neg\theta) \Leftrightarrow F_0$	Leyes inversas
9	$(\theta \vee T_0) \Leftrightarrow T_0$ $(\theta \wedge F_0) \Leftrightarrow F_0$	Leyes de dominación
10	$(\theta \vee (\theta \wedge \varphi)) \Leftrightarrow \theta$ $(\theta \wedge (\theta \vee \varphi)) \Leftrightarrow \theta$	Leyes de absorción

Notemos que la fórmula $(\theta \vee F_0) \Leftrightarrow \theta$, nos dice que la valuación de la disyunción $(\theta \vee F_0)$ depende sólo de la valuación de la fórmula θ. Observemos también que $(\theta \vee \neg\theta)$ es siempre

una tautología.

Salvo la ley de la doble negación, estas leyes parecen agruparse de manera natural en pares de fórmulas y esto da origen a nuevas ideas. Observa los pares de fórmulas numerados desde 2 hasta 10 en la tabla precedente y los cambios de símbolos que implican. Este problema será tratado más adelante, cuando estudiemos Álgebras de Boole. Las álgebras de Boole constituyen una generalización de los resultados desarrollados hasta aquí.

Ejemplo 20

Sean $S_1 = \{(p \vee q), (q \rightarrow t), (p \rightarrow r)\}$ y $S_2 = \{(r \vee t)\}$. Veamos que $S_1 \models S_2$. Esto equivale a demostrar que $\{(p \vee q), (q \rightarrow t), (p \rightarrow r)\} \models (r \vee t)$. Como $\models ((r \vee t) \leftrightarrow (\neg r \rightarrow t))$, vamos a demostrar que $\{(p \vee q), (q \rightarrow t), (p \rightarrow r)\} \models (\neg r \rightarrow t)$. Por el teorema de la deducción es equivalente a demostrar que $\{(p \vee q), (q \rightarrow t), (p \rightarrow r)\} \cup \neg r \models t$. Sea I cualquier interpretación que hace verdadero el antecedente. Luego $v(\neg r) = 1$ y en consecuencia $v(r) = 0$. Como $v(p \rightarrow r) = 1$, entonces $v(p) = 0$; dado que $v(p \vee q) = 1$ entonces $v(q) = 1$. Finalmente como $v(q \rightarrow t) = 1$ y $v(q) = 1$ entonces se deduce que $v(t) = 1$.

1.4.6 Síntesis semántica

Es importante que revisemos algunas fbfs, a modo de fijación de los aprendizajes. Por ejemplo, las siguientes expresiones lógicas son tautologías, es decir, son verdaderas bajo toda interpretación. Sean las fórmulas θ, φ y ρ, luego se verifican las siguientes implicaciones ($\Rightarrow$) o equivalencias lógicas ($\Leftrightarrow$):

a) $(\theta \wedge (\theta \rightarrow \varphi)) \Rightarrow \varphi$

b) $(\theta \rightarrow \varphi) \Leftrightarrow (\neg\theta \vee \varphi)$

c) $((\theta \rightarrow \varphi) \wedge (\varphi \rightarrow \rho)) \Rightarrow (\theta \rightarrow \rho)$

d) $(\theta \leftrightarrow \varphi) \Leftrightarrow ((\theta \wedge \varphi) \vee (\neg\theta \wedge \neg\varphi))$

e) $\neg(\theta \rightarrow \varphi) \Rightarrow \theta$

f) $\neg(\theta \rightarrow \varphi) \Rightarrow \neg\varphi$

Veamos el caso e). Sabemos que una implicación es siempre verdadera, excepto en un único caso, aquel donde el antecedente es verdadero y el consecuente falso. Luego, esta implicación será tautológica siempre que probemos que admitiendo un antecedente verdadero el consecuente también lo es; es decir que no existirán casos en que pueda concluirse algo falso si el antecedente es verdadero.

Sea entonces I cualquier interpretación tales que $v_I(\neg(\theta \rightarrow \varphi)) = 1$, entonces $v_I(\theta \rightarrow \varphi) = 0$, de donde $v_I(\theta) = 1$ y $v_I(\varphi) = 0$. Como $v_I(\theta) = 1$, luego $(\neg(\theta \rightarrow \varphi) \rightarrow \theta)$ es tautología, como queríamos probar. Además al ser $v_I(\varphi) = 0$, entonces $v_I(\neg\varphi) = 1$ con lo que tenemos demostrado que f) es también una implicación lógica (o tautología).

Argumentos de este tipo son útiles para continuar probando las siguientes implicaciones por lo que recomendamos no usar tablas de verdad.

Otras implicaciones o equivalencias lógicas son:

g) $(\neg\varphi \wedge (\theta \to \varphi)) \Rightarrow \neg\theta$

h) $(\neg\theta \wedge (\theta \vee \varphi)) \Rightarrow \varphi$

i) $((\theta \vee \varphi) \wedge ((\theta \to \rho) \wedge (\varphi \to \rho))) \Rightarrow \rho$

j) $((\theta \to \varphi) \wedge (\theta \to \rho)) \Rightarrow (\theta \to (\varphi \wedge \rho))$

k) $((\theta \wedge \varphi) \to \rho) \Rightarrow (\theta \to (\varphi \to \rho))$

l) $(\theta \to \varphi) \Leftrightarrow ((\theta \wedge \neg\varphi) \to F_0)$

El enunciado k) es un caso particular del Teorema de la Deducción.

También es una implicación lógica el recíproco del Teorema de la Deducción. Por lo que $((\theta \wedge \varphi) \to \rho) \Leftrightarrow (\theta \to (\varphi \to \rho))$.

En efecto:

$((\theta \wedge \varphi) \to \rho) \Leftrightarrow (\neg(\theta \wedge \varphi) \vee \rho) \Leftrightarrow ((\neg\theta \vee \neg\varphi) \vee \rho) \Leftrightarrow (\neg\theta \vee (\neg\varphi \vee \rho)) \Leftrightarrow (\neg\theta \vee (\varphi \to \rho)) \Leftrightarrow (\theta \to (\varphi \to \rho))$.

Es importante atender a la equivalencia l) que utilizaremos habitualmente en Matemática para justificar una demostración por "*reducción al absurdo*"; F_0 es cualquier contradicción, equivalente a $(\theta \wedge \neg\theta)$, por ejemplo.

Finalmente, recordamos la equivalencia $(\theta \to \varphi) \Leftrightarrow (\neg\varphi \to \neg\theta)$, entre el condicional directo con su contra recíproco. Muchos teoremas en Matemática utilizan esta equivalencia para justificar la "*demostración por contraposición*".

Actividad 5

Problema Nº1: Sean p, q proposiciones. Demuestra las siguientes equivalencias:

1.1) $(p \to q) \Leftrightarrow (\neg p \vee q)$

1.2) $((p \to q) \wedge (q \to p)) \Leftrightarrow (p \leftrightarrow q)$

1.3) $(p \veebar q) \Leftrightarrow ((p \wedge \neg q) \vee (\neg p \wedge q))$

1.4) $(p \veebar q) \Leftrightarrow ((p \vee q) \wedge \neg(p \wedge q))$

1.5) $(p \leftrightarrow q) \Leftrightarrow ((p \wedge q) \vee (\neg p \wedge \neg q))$

Problema Nº2: Sean p, q proposiciones. Demuestra las equivalencias lógicas:

2.1) $(p \to q) \Leftrightarrow (\neg q \to \neg p)$.

2.2) $(q \to p) \Leftrightarrow (\neg p \to \neg q)$.

2.3) $((p \wedge q) \to r) \Leftrightarrow (p \to (q \to r))$. (Caso particular del teorema de la deducción).

Problema Nº3: Simplifica las proposiciones siguientes, si es posible:

3.1) $((p \vee q) \vee (\neg p \wedge \neg q))$

3.2) $((T_0 \wedge (T_0 \vee ((\neg p \vee \neg q) \vee r))) \wedge (\neg p \wedge p))$

3.3) $(T_0 \to F_0)$

3.4) $\neg(\neg(r \wedge p) \vee (q \wedge (\neg r \vee \neg p)))$

Problema Nº4: Escribe cada enunciado en forma simbólica y determina uno equivalente al dado, en lenguaje coloquial.

4.1) No es verdad que, si el software falla entonces la causa es el error de sintaxis.

4.2) Si la ruta de Rosario a Bs. As es peligrosa entonces viajaré de día.

4.3) Si los índices de producción y de exportación de Argentina aumentan entonces aumenta el ingreso per cápita de sus habitantes.

4.4) En sección alumnado de la Facultad me dijeron: Cursas las asignaturas por la mañana o por la tarde, pero no en ambos turnos.

Problema Nº5: Sean θ, φ y ρ fbfs. Prueba las siguientes afirmaciones:

5.1) $\{(\neg\varphi \rightarrow \theta), (\neg\theta \vee \rho), (\neg\varphi \rightarrow \neg\rho)\} \models \varphi$

5.2) $\{(\theta \wedge \varphi)\} \models \theta$; $\{(\theta \wedge \varphi)\} \models \varphi$

5.3) $\{(\theta \vee \varphi), (\neg\theta \vee \rho)\} \models (\varphi \vee \rho)$

Problema Nº6: Escribe 5 fórmulas que pertenezcan al conjunto Con $\{p, (p \leftrightarrow \neg q)\}$.

1.4.7 Conectivos adecuados

En esta sección trabajaremos con fórmulas escritas en términos de proposiciones atómicas. Pero no se pierde generalidad ya que los resultados se extienden para fórmulas. La fórmula $(p \rightarrow q)$ es lógicamente equivalente a $(\neg p \vee q)$. Luego, conociendo el significado de los conectivos negación ($\neg$) y disyunción ($\vee$) no se necesitaría la definición del conectivo implicación ($\rightarrow$), dado que la implicación se puede expresar en términos de negación y disyunción.

En la tabla siguiente observamos una equivalencia entre el bicondicional y la conjunción de dos condicionales, es decir: $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$.

Tabla 1.23

p	q	$(p \rightarrow q)$	$(q \rightarrow p)$	$((p \rightarrow q) \wedge (q \rightarrow p))$	$(p \leftrightarrow q)$	$(((p \rightarrow q) \wedge (q \rightarrow p)) \leftrightarrow (p \leftrightarrow q))$
0	0	1	1	1	1	1
0	1	1	0	0	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	1

Esto sugiere que también el bicondicional puede ser expresado usando negaciones, conjunciones y disyunciones. Estos ejemplos muestran que es posible reducir el número de conectivos lógicos que requiere nuestro alfabeto, ya que "$\rightarrow$" y "$\leftrightarrow$" pueden expresarse en términos de negaciones, conjunciones y disyunciones.

Finalmente, podemos expresar que dado un conjunto de conectivos, se dice que es *adecuado* si a partir de sus elementos pueden definirse todos los demás conectivos. El conjunto $\{\neg, \wedge, \vee\}$ es adecuado.

En consecuencia, toda fórmula del LPC puede ser construida a partir de este conjunto adecuado de conectivos.

En efecto $(\theta \to \varphi) \equiv (\neg\theta \vee \varphi)$; $(\theta \leftrightarrow \varphi) \equiv (\theta \to \varphi) \wedge (\varphi \to \theta) \equiv (\neg\theta \vee \varphi) \wedge (\neg\varphi \vee \theta)$

Es posible demostrar que $\{\neg, \wedge\}$, $\{\neg, \vee\}$, $\{\neg, \to\}$ son conjuntos de conectivos adecuados y que no lo son $\{\neg\}$, $\{\wedge, \vee\}$, $\{\to, \vee\}$.

1.5 Convenciones para simplificar la notación

En general, las fórmulas de la LPC requieren de paréntesis para estar sintánticamente bien construidas. A efectos de simplificar las notaciones y de limitar el número de paréntesis requeridos por estas definiciones sintácticas, vamos a hacer nuevos acuerdos sobre su uso, siempre que no sean causales de ambigüedades o confusiones.

En lo sucesivo, para simplificar la notación omitiremos los paréntesis exteriores al escribir fórmulas. Escribiremos $(p \vee q)$, como $p \vee q$; análogamente, escribiremos $(p \to q)$ como la implicación $p \to q$. Anotaremos $p \wedge (q \vee r)$ en lugar de $(p \wedge (q \vee r))$. En este caso, mantenemos los paréntesis en $(q \vee r)$. Al quitarlos obtenemos la expresión $p \wedge q \vee r$ que se torna ambigua.

Surge el primer problema: ¿qué sentido le vamos a asignar a $p \wedge q \vee r$?, ¿lo tiene? Sí; pero debemos priorizar el orden en que se van ejecutando las operaciones lógicas. Por ejemplo, en la proposición $p \wedge q \vee r$, ¿qué conectivo resolvemos primero?

Para el caso $p \to q \to r$, ¿en qué orden ejecutamos las operaciones indicadas? Podemos pensar en $(p \to q) \to r$ o en $p \to (q \to r)$, pero inmediatamente surge la cuestión de analizar los valores de verdad que se obtienen en cada caso, ¿son los mismos? Veamos los criterios que vamos a aplicar para determinar el orden de evaluación de los operadores.

Reglas de prioridad

Al trabajar con fbfs, hemos decidido eliminar los paréntesis exteriores de las fórmulas para simplificar su escritura. Esta simplificación, en general, no ocasiona ambigüedades; por ejemplo, al escribir $(p \vee (q \wedge r)) \to s$ no caben dudas que nos estamos refiriendo a la fbf $((p \vee (q \wedge r)) \to s)$.

Para simplificar aún más la escritura de las fbfs, es decir, para eliminar de ellas una mayor cantidad de paréntesis sin dar lugar a confusiones, introducimos las llamadas *reglas de prioridad o precedencia*.

Dichas reglas establecen una prioridad para los conectivos, lo que nos permiten reconstruir unívocamente a toda fbf que se le hayan suprimido los paréntesis no necesarios, de acuerdo a estas reglas.

La conexión $\neg$ (operador unario) tiene siempre la prioridad más alta.

Para los restantes conectivos (operadores binarios), la prioridad más alta se le da a $\wedge$, seguida por $\vee$, $\to$, $\leftrightarrow$ en ese orden.

Al decir prioridad mas alta hacemos referencia a que, si no aparecen paréntesis, inferimos que en la construcción recursiva de la fórmula primero se negaron todas las variables proposicionales a las que antecede el conectivo $\neg$, después se realizaron todas las conjunciones

que aparecen indicadas, luego las disyunciones, luego las implicancias y por último las equivalencias.

Al escribir la expresión $p \to q \wedge t$, y tener en claro que estamos trabajando con fbfs, hacemos referencia a la fbf $(p \to (q \wedge t))$ ya que el conectivo $\wedge$ tiene prioridad sobre el conectivo $\to$. Como $\leftrightarrow$ tiene la prioridad más baja, la expresión $p \leftrightarrow q \to r$ debe ser entendida como la fbf $(p \leftrightarrow (q \to r))$.

Al escribir la expresión $p \to \neg q \wedge r \vee s \leftrightarrow \neg t$, y tener en claro que estamos trabajando con fbfs, hacemos referencia a la fbf $((p \to ((\neg q \wedge r) \vee s)) \leftrightarrow \neg t)$.

Si en una expresión los conectivos tienen el mismo orden de precedencia, la evaluación de tales expresiones se realiza de izquierda a derecha. Por ejemplo la expresión $p \to q \to r$ hace referencia a $((p \to q) \to r)$.

Por otra parte, si queremos hacer referencia a una fbf que en su construcción recursiva se haya invertido alguno de los órdenes supuestos por las reglas de prioridad, no podemos eliminar todos los paréntesis sino que debemos dejar los que indican la precedencia en la construcción.

Para hacer referencia a la fbfs $((p \wedge (r \vee s)) \leftrightarrow t)$ podemos eliminar algunos paréntesis teniendo en cuenta las reglas de prioridad y simplemente escribir $p \wedge (r \vee s) \leftrightarrow t$; sin embargo no podemos eliminar todos los paréntesis y escribir $p \wedge r \vee s \leftrightarrow t$ porque esta expresión hace referencia a la fbf $(((p \wedge r) \vee s) \leftrightarrow t)$.

El uso de corchetes

Otro abuso de notación que utilizaremos consiste en la sustitución de paréntesis por corchetes. Así, la fbf $((p \wedge (r \vee s)) \leftrightarrow t)$ puede escribirse como $[p \wedge (r \vee s)] \leftrightarrow t$.

Actividad 6

Problema Nº1: Demuestra que $\{\neg, \wedge\}$, $\{\neg, \vee\}$, $\{\neg, \to\}$ son conjuntos adecuados de conectivos.

Problema Nº2: Considera las siguientes fórmulas bien formadas:

2.1) $(p \to (q \to r))$

2.2) $(\neg p \to \neg(q \to r))$

2.3) $(p \vee q)$

Encuentra fórmulas equivalentes a ellas que usen sólo los conectivos:

a) $\{\neg, \wedge\}$ b) $\{\neg, \vee\}$ c) $\{\neg, \to\}$

Problema Nº3: Sean p, q, r, t variables proposicionales:

3.1) Mediante la inserción de paréntesis (y corchetes), indica el orden en que se ejecutan los conectivos de acuerdo a lo establecido por las reglas de precedencia.

a) $p \wedge q \to p$ b) $p \wedge q \vee r \leftrightarrow p \wedge t$

3.2) Elimina los paréntesis en los casos en que sea posible, de acuerdo a lo fijado por las reglas de precedencia.

a) $((p \wedge q) \rightarrow (p \rightarrow t))$ c) $((p \wedge (p \rightarrow q)) \rightarrow q)$

b) $((p \wedge q) \vee ((r \leftrightarrow p) \wedge t))$ d) $((p \wedge q) \vee (\neg p \wedge \neg q))$

1.6 Redes de Conmutación

A continuación desarrollaremos una aplicación importante de la lógica proposicional en la construcción y simplificación de redes de conmutación.

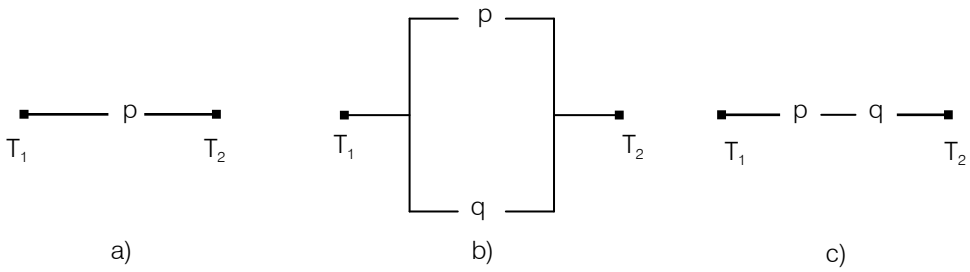
Una red de conmutación está formada por cables e interruptores que conectan dos terminales T_1 y T_2. Si un interruptor está abierto, entonces no pasa la corriente por él y lo indicaremos con un "0", mientras que un "1" indicará que el interruptor está cerrado y por consiguiente pasa la corriente por él.

En la Figura 1.1, dada a continuación, expresamos distintas formas de colocar los interruptores. En a) existe sólo un interruptor indicado con la letra "p" mientras que en las redes b) y c) se tienen dos interruptores (independientes), que indicamos como "p" y "q".

Para la red de b), la corriente pasa de T_1 a T_2 si alguno de los interruptores p, q está cerrado. Aquí los interruptores están *en paralelo* y este esquema se puede representar mediante la proposición "$p \vee q$". La red de c) necesita que los dos interruptores p, q estén cerrados para que la corriente circule de T_1 a T_2. Aquí los interruptores están *en serie* y esta red se representa por la proposición "$p \wedge q$".

Los interruptores de una red no tienen por qué actuar independientemente unos de otros. Éstos se acoplan de manera que "p" está abierto (cerrado) si y sólo si, "$\neg p$" está cerrado (abierto) simultáneamente.

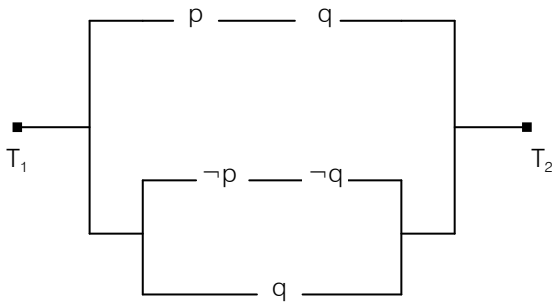
Figura 1.1



Ejemplo 21

La Figura 1.2 muestra una red de conmutación y nuestro objetivo es simplificarla. Esto significa analizar la posibilidad de obtener otra red equivalente a la dada con menor cantidad de interruptores y/o conexiones.

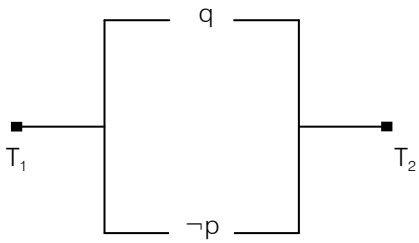
La red de la Figura 1.2 se expresa como $((p \wedge q) \vee ((\neg p \wedge \neg q) \vee q))$
Figura 1.2



Aplicando las equivalencias lógicas obtenemos:

$(p \wedge q) \vee [(\neg p \wedge \neg q) \vee q]$	**JUSTIFICACIÓN**
$\Leftrightarrow (p \wedge q) \vee [(\neg p \vee q) \wedge (\neg q \vee q)]$	Ley distributiva de $\vee$ respecto de $\wedge$
$\Leftrightarrow (p \wedge q) \vee [(\neg p \vee q) \wedge T_0]$	Ley inversa de $\vee$
$\Leftrightarrow (p \wedge q) \vee (\neg p \vee q)$	Ley del elemento neutro o identidad de $\wedge$
$\Leftrightarrow [(p \wedge q) \vee q] \vee \neg p$	Leyes conmutativa y asociativa de $\vee$
$\Leftrightarrow (q \vee \neg p)$	Ley de absorción

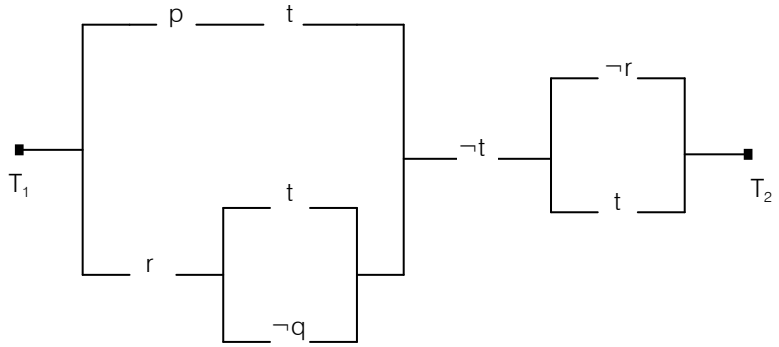
La red simplificada se expresa como $(q \vee \neg p)$ y aparece dibujada en la siguiente figura.
Figura 1.3



Ejemplo 22
La red de la Figura 1.4 a) se representa por:

$$((p \wedge t) \vee (r \wedge (t \vee \neg q))) \wedge \neg t \wedge (\neg r \vee t)$$

Figura 1.4 a)



Usando las leyes lógicas, se simplifica esta proposición, de la manera siguiente:

$((p \wedge t) \vee (r \wedge (t \vee \neg q))) \wedge \neg t \wedge (\neg r \vee t)$	**JUSTIFICACIÓN**
$\Leftrightarrow ((p \wedge t \wedge \neg t) \vee (r \wedge (t \vee \neg q) \wedge \neg t)) \wedge (\neg r \vee t)$	Ley distributiva de $\wedge$ sobre $\vee$
$\Leftrightarrow ((p \wedge F_0) \vee (r \wedge (t \vee \neg q) \wedge \neg t)) \wedge (\neg r \vee t)$	Ley inversa ($t \wedge \neg t \Leftrightarrow F_0$)
$\Leftrightarrow (F_0 \vee (r \wedge ((t \wedge \neg t) \vee (\neg q \wedge \neg t)))) \wedge (\neg r \vee t)$	Ley dominación $\wedge$ y distributiva
$\Leftrightarrow r \wedge (F_0 \vee (\neg q \wedge \neg t)) \wedge (\neg r \vee t)$	Ley identidad $\vee$ Ley inversa ($t \wedge \neg t \Leftrightarrow F_0$)
$\Leftrightarrow (r \wedge \neg q \wedge \neg t) \wedge (\neg r \vee t)$	Ley identidad $\vee$
$\Leftrightarrow (r \wedge \neg q \wedge \neg t \wedge \neg r) \vee (r \wedge \neg q \wedge \neg t \wedge t)$	Ley distributiva de $\wedge$ sobre $\vee$
$\Leftrightarrow F_0 \vee F_0$	Ley inversa $\wedge$ Ley dominación $\wedge$
$\Leftrightarrow F_0$	Ley idempotente

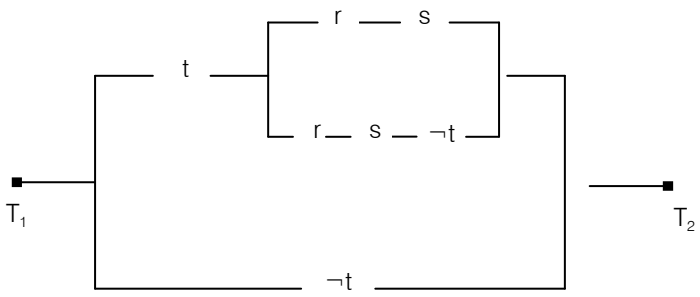
Entonces, $((p \wedge t) \vee (r \wedge (t \vee \neg q))) \wedge \neg t \wedge (\neg r \vee t) \Leftrightarrow F_0$, y la red de la Figura 1.4.b) es equivalente a la red original. La red de conmutación simplificada queda representada como lo indica la figura 1.4 b) por dos terminales T_1 y T_2, y un interruptor abierto. Esto significa que cuando el terminal T_1 de la figura 1.4 a) recibe un estímulo eléctrico, éste nunca llega a T_2.

Figura 1.4 b)

Ejemplo 23

Si consideramos la red de la siguiente figura:

Figura 1.5 a)

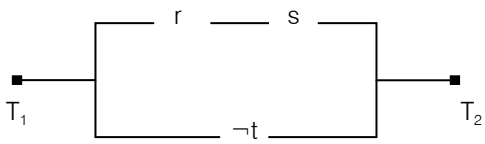


Esta red se representa por $(t \wedge ((r \wedge s) \vee (r \wedge s \wedge \neg t))) \vee \neg t$. Usando las leyes lógicas, se simplifica esta proposición, de la siguiente manera:

$(t \wedge ((r \wedge s) \vee (r \wedge s \wedge \neg t))) \vee \neg t$	**JUSTIFICACIÓN**
$\Leftrightarrow (t \wedge (r \wedge s)) \vee \neg t$	Ley de absorción
$\Leftrightarrow (t \vee \neg t) \wedge ((r \wedge s) \vee \neg t)$	Ley distributiva de $\vee$ sobre $\wedge$
$\Leftrightarrow T_0 \wedge ((r \wedge s) \vee \neg t)$	Ley inversa
$\Leftrightarrow (r \wedge s) \vee \neg t$	Ley identidad de $\wedge$

Entonces, $(t \wedge ((r \wedge s) \vee (r \wedge s \wedge \neg t))) \vee \neg t \Leftrightarrow (r \wedge s) \vee \neg t$, es decir, la red de la figura 1.5 b) es equivalente a la red original.

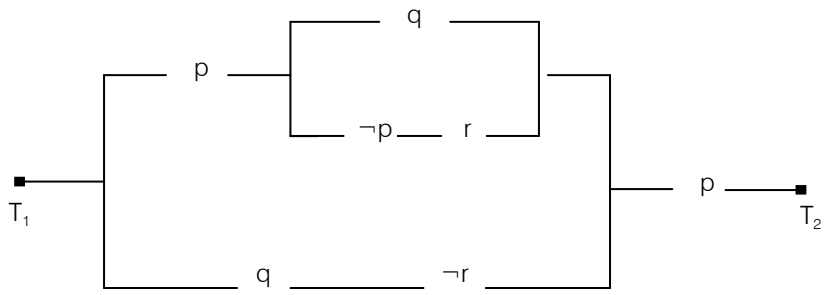
Figura 1.5 b)



La ventaja de haber simplificado la red de la figura 1.5 a) reside en el hecho de que la red de la figura 1.5 b) tiene sólo tres interruptores, cuatro menos que la red original.

Actividad 7

Problema Nº1: Dada la siguiente red de conmutación:



1.1) Indica para que combinaciones de interruptores todo estímulo eléctrico que recibe T_1 se propaga por la red y llega a T_2.

1.2) Simplifica la red.

1.3) Grafica la red obtenida en el ítem anterior.

Problema Nº2: Completa la columna justificación, indicando la ley utilizada:

$t \wedge (t \vee \neg r) \wedge (s \vee \neg t) \wedge (s \vee r)$	**JUSTIFICACIÓN**
$\Leftrightarrow t \wedge (s \vee \neg t) \wedge (s \vee r)$	
$\Leftrightarrow ((t \wedge s) \vee (t \wedge \neg t)) \wedge (s \vee r)$	
$\Leftrightarrow ((t \wedge s) \vee F_0) \wedge (s \vee r)$	
$\Leftrightarrow (t \wedge s) \wedge (s \vee r)$	
$\Leftrightarrow (t \wedge s \wedge s) \vee (t \wedge s \wedge r)$	
$\Leftrightarrow (t \wedge s) \vee (t \wedge s \wedge r)$	
$\Leftrightarrow t \wedge s$	

Problema Nº3: Dada la siguiente proposición lógica:

$$(s \wedge p \wedge ((s \wedge p) \vee t)) \vee ((p \vee \neg p) \wedge t)$$

3.1) Dibuja la red de conmutación que modela.

3.2) Simplifica la proposición lógica.

3.3) Dibuja la red de conmutación que modela la proposición lógica simplificada.

Problema Nº4: Sea $[(\neg s \vee (r \wedge p) \vee r) \wedge \neg s] \vee (\neg s \vee (s \wedge r))$

4.1) Diseña un circuito que represente la expresión simbólica dada.

4.2) Encuentra una red de conmutación que sea equivalente a la original mediante la simplificación de la expresión simbólica dada.

Problema Nº5: Sea $(p \vee q \vee r) \wedge (p \vee t \vee \neg q) \wedge (p \vee \neg t \vee r)$

5.1) Diseña un circuito que represente la expresión simbólica dada.

5.2) Encuentra una red de conmutación que sea equivalente a la original mediante la simplificación de la expresión dada.

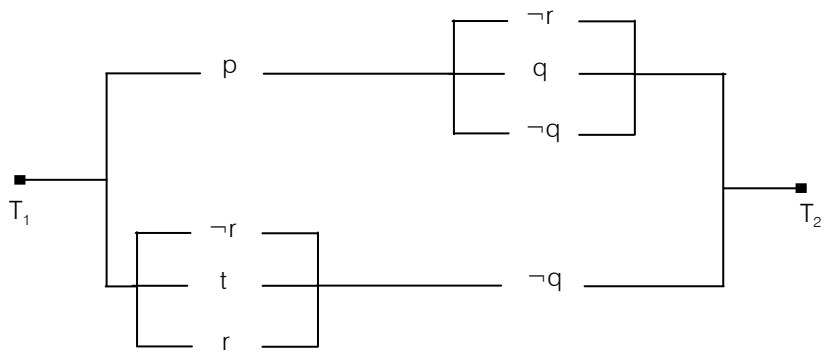
5.3) Dibuja la red simplificada.

Problema Nº6: Dada la siguiente red de conmutación

6.1) Escribe la expresión lógica que representa la figura dada.

6.2) Simplifica la expresión obtenida.

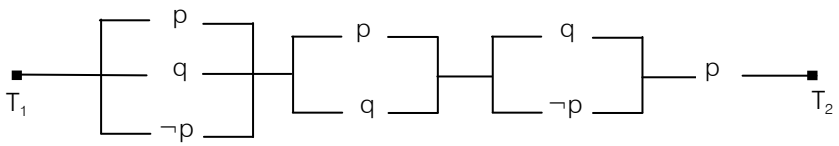
6.3) Grafica la red simplificada.



Problema Nº7: Indica los pasos seguidos y las leyes lógicas utilizadas para justificar la siguiente simplificación:

$$p \wedge [(\neg q \rightarrow (r \wedge r)) \vee \neg [q \vee ((r \wedge s) \vee (r \wedge \neg s))]\,] \Leftrightarrow p$$

Problema Nº8: Dada la siguiente red, simplifica y dibuja una equivalente:



1.7 Predicados, relaciones y cuantificadores. Primer acercamiento a la LPO.

Tanto en matemática como en las ciencias de la computación aparecen muy a menudo *expresiones relacionales o predicados* del tipo "$x \leq 4$", "$y^2 \geq 9$", "$|\ x\ | > 0$", "$x < 0$ mientras que $y > 0$", "$x \geq y$", "ABC es triángulo rectángulo", "María es hermana de José". Tales *expresiones relacionales o predicados expresan propiedades de un objeto o relaciones entre obje-*

tos. Se convierten en proposiciones lógicas que tienen valores de verdad 0 o 1 cuando se reemplaza a las variables x, y por valores constantes o cuando se cuantifica. Tomando $x = 2$ en "$x \leq 4$" obtenemos la proposición verdadera "$2 \leq 4$" mientras que si $x = 5$, "$5 \leq 4$" es una proposición falsa; "todo x es menor o igual que 4" es falsa, mientras que "existe un número real cuyo cuadrado es mayor o igual que 9" es verdadera.

Estas expresiones relacionales juegan un papel importante en estructuras de decisión que pueden aparecer en algunos lenguajes de programación y se pueden manipular como proposiciones lógicas aunque no lo sean. Más adelante, en LPO, formalizaremos y estudiaremos en detalle estos predicados. Por ahora, veamos informalmente algunos ejemplos.

Ejemplo 24

a) Trabajaremos en el conjunto de los números reales. Las expresiones p(x): "$| x | = 2$" y q(x): "$x = 2 \vee x = -2$" son expresiones relacionales o predicados. En este ejemplo serán trabajadas como proposiciones.

Sean p: "$| x | = 2$" y q: "$x = 2 \vee x = -2$". La implicación $p \rightarrow q$ es una tautología que se expresa diciendo "Si el valor absoluto de un número real es 2 entonces dicho número es 2 o es su opuesto – 2". Aquí decimos que "p es condición suficiente para q" o bien que "q es condición necesaria para p". La implicación $q \rightarrow p$ es también una tautología, de donde "q es condición suficiente para p" o bien que "p es condición necesaria para q". Luego podemos concluir que p es condición necesaria y suficiente para q.

b) Sean p: "$x^2 = 4$" y q: "$x = 2$". La implicación $p \rightarrow q$ no es una tautología dado que para la sustitución $x = -2$, p es verdadera mientras que q es falsa. Luego no podemos hablar de condiciones necesarias y suficientes.

Sin embargo, la implicación $q \rightarrow p$ es una tautología que se expresa diciendo "Si un número es 2 entonces su cuadrado es 4", de donde "q es condición suficiente para p" o bien que "p es condición necesaria para q".

c) Mencionamos el hecho de que existen oraciones que contienen una o más variables y no son proposiciones. Oraciones como "x es un número mayor que 5", "el cubo del número x es el número y", "z es solución de la ecuación $x^2 + x - 1 = 0$" no son proposiciones. Tales expresiones se convierten en proposiciones lógicas cuando se reemplaza a la(s) variable(s) por un valor correspondiente al dominio donde se encuentra(n) definida(s).

Si en la oración "x es un número mayor que 5" elegimos como dominio de la variable x el conjunto de los números naturales. Sustituimos x por 2 y obtenemos la proposición "2 es un número mayor que 5" que es falsa, mientras que si x es igual a 6, la oración "6 es un número mayor que 5" es una proposición verdadera.

Notar que es necesario especificar a x para determinar la verdad o falsedad de la expresión "x es un número mayor que 5".

d) Es importante indicar la pertenencia de x un dominio D. En la expresión "$x \in \mathbf{N}$ y x es un número par", x representa una variable en el conjunto de los números naturales que es su

dominio de definición y el predicado P(x): x es un número par, representa la propiedad relativa al objeto x.

e) En la expresión: "$x^2 + x - 1 = 0$, donde $x \in \mathbf{R}$", el predicado Q(x): "$x^2 + x - 1 = 0$" expresa un atributo o propiedad de la variable x, y la expresión $x \in \mathbf{R}$ indica que el dominio para x es el conjunto de los números reales.

Estas expresiones no son proposiciones, pero se pueden convertir en proposiciones. Una forma de hacerlo es sustituyendo las variables por alguna constante perteneciente al dominio. Más adelante, estudiaremos detalladamente este tema.

1.7.1 El dominio de las variables

Veamos en el siguiente ejemplo el rol del dominio al que pertenecen las variables. La verdad o falsedad de una expresión puede depender del dominio al que pertenezcan las variables.

Ejemplo 25

Consideremos la expresión Q(z): "z es solución de la ecuación $x^2 + x - 1 = 0$".

En "z es solución de la ecuación $x^2 + x - 1 = 0$, donde $z \in \mathbf{Z}$", que puede ser expresada como "z es una solución entera de $x^2 + x - 1 = 0$", observamos que para cualquier valor entero "a" que tome la variable z, resulta Q(a): "a es solución de la ecuación $x^2 + x - 1 = 0$" es una proposición siempre falsa. Por ejemplo, son proposiciones falsas Q(–5), Q(0), Q(3).

En cambio, si elegimos como dominio el conjunto de los números reales la sustitución de z por 0 es Q(0): "0 es solución de la ecuación $x^2 + x - 1 = 0$", que es una proposición falsa; pero si $z = \frac{-1-\sqrt{5}}{2}$ resulta que $Q(\frac{-1-\sqrt{5}}{2})$ es una proposición verdadera.

1.7.2 ¿Cómo se obtienen proposiciones?

Veremos dos formas para obtener proposiciones a partir de los predicados.

Forma 1: *Sustitución*

La variable se reemplaza en el predicado por una constante perteneciente a un dominio o universo dado. Esto significa que la elección de la constante es importante, y que debemos atender además al dominio o universo al cual pertenece.

Ejemplo 26

Consideremos la siguiente expresión en dos variables:

S(x, y): "Los números $x - 3$, $y + 2$, $x - y$, $2x - y$ son números positivos, $x \in \mathbf{Z}$, $y \in \mathbf{Z}$". Cada una de ellas aparece más de una vez. Cuando sustituimos una de las letras x por una constante de nuestro conjunto dominio o universo, convenimos en sustituir *todas* las apariciones que hace esta letra x en la expresión. De esta manera si $x = 3$ entonces S(3, y) resulta:

S(3, y): “Los números 0, y + 2, 3 – y, 6 – y son números positivos, y ∈ **Z**”.

Análogamente a x, cuando se sustituye la letra y por un valor constante, la sustitución afecta a *todas* sus apariciones. En efecto, si x = 3 e y = 5 entonces:

S(3, 5): “Los números 0, 7, –2, 1 son números positivos” es una proposición falsa.

Notemos que para obtener una proposición es necesario sustituir todas las variables por un valor perteneciente al dominio.

Las *Expresiones Relacionales o Predicados* se operan lógicamente como si fuesen proposiciones, obteniéndose nuevas expresiones relacionales. Es decir, si P(x) y Q(x) son expresiones relacionales, también lo son:

$$\neg P(x);\ Q(x) \wedge P(x);\ Q(x) \vee P(x);\ P(x) \rightarrow Q(x).$$

Por sustitución, estas nuevas expresiones relacionales también se convierten en proposiciones cuyos valores de verdad dependen de los valores de verdad de las proposiciones que las componen. Es decir, algunas sustituciones producen proposiciones verdaderas y otros reemplazos producen proposiciones falsas.

Ejemplo 27

a) La negación de R(x): “x es un número que es un cubo perfecto, x ∈ **N**” es la expresión ¬R(x): “x es un número que no es un cubo perfecto, x ∈ **N**” o bien “no es cierto que el número x sea un cubo perfecto, x ∈ **N**”.

b) La negación de T(x, y): “ x es divisor de y, x ∈ **N** e y ∈ **N** ” es la expresión ¬T(x, y): “x no es divisor de y, x ∈ **N** e y ∈ **N**” o bien “no es cierto que x es divisor de y, x ∈ **N** e y ∈ **N**”.

c) U(x): “x es un número mayor que 5 y x es un cubo perfecto, x ∈ **N**” resulta de la conjunción de:

P(x): “x es un número mayor que 5, x ∈ **N**”

R(x): “x es un número que es un cubo perfecto, x ∈ **N**”.

d) S(x, y): “Los números x – 3, y + 2, x – y, 2 x – y son números positivos, x ∈ **Z**, y ∈ **Z**”, puede pensarse como la conjunción de cuatro proposiciones,

P(x): “el número x – 3 es un número positivo, x ∈ **Z**”.

Q(y): “el número y + 2 es un número positivo, y ∈ **Z**”.

R(x, y): “el número x – y es un número positivo, x ∈ **Z**, y ∈ **Z**”.

T(x, y): “el número 2 x – y es un número positivo, x ∈ **Z**, y ∈ **Z**”.

e) Al reemplazar x por 3 e y por 5 en d), obtenemos las proposiciones:

P(3): “el número 0 es un número positivo”.

Q(5): “el número 7 es un número positivo”.

R(3, 5): “el número –2 es un número positivo”.

T(3, 5): “el número 1 es un número positivo”.

Luego S(3, 5) = P(3) ∧ Q(5) ∧ R(3, 5) ∧ T(3, 5), es una proposición falsa, pues P(3) y R(3,5) son dos proposiciones falsas.

f) Ahora, si reemplazamos x por 10 e y por 2 en S(x, y) del item d) obtenemos la proposición S(10, 2): “Los números 7, 4, 8, 18 son números positivos”. Para $x = 10$ e $y = 2$ obtenemos una proposición verdadera $P(10) \wedge Q(2) \wedge R(10, 2) \wedge T(10, 2)$ porque las proposiciones

P(10): “el número 7 es un número positivo”

Q(2): “el número 4 es un número positivo”

R(10, 2): “el número 8 es un número positivo”

T(10, 2): “el número 18 es un número positivo”, son todas verdaderas.

Ejemplo 28

Ahora, si reemplazamos las variables x e y por cualquier valor entero en W(x, y): “Los números $x – 3$, $y + 2$, $x – y$, $2x – y$ son números enteros”, como la adición, la sustracción y la multiplicación de números enteros es otro número entero resulta que, W(a, b): “Los números $a – 3$, $b + 2$, $a – b$, $2a – b$ son números enteros” una proposición siempre verdadera.

Hemos visto en los ejemplos que podemos construir expresiones de manera que, cuando se sustituyen las variables por algunos o todos los valores pertenecientes al dominio, resulta una proposición *verdadera*; cuando se sustituyen las variables por algunos o todos los valores pertenecientes al dominio, resulta una proposición *falsa*.

Forma 2: *Cuantificación*

Otra forma de obtener proposiciones a partir de expresiones relacionales es introduciendo el uso de símbolos, denominados *cuantificadores*.

Diariamente solemos escuchar este tipo de oraciones:

Todos los ingresantes a la Universidad deben completar su ficha médica.

Todas las materias tienen cursado anual.

Algunas materias tienen cursado cuatrimestral.

Existen alumnos que disfrutan estudiando.

Estas frases “Todos los ingresantes.....”; “Algunas materias.....” indican la frecuencia con la cual todos o algún sujeto de un dominio cumplen una propiedad.

“Para algún x”, “Para algunos x, y”, “Para todo x”, “Para todos x, y” aparecen como *cuantificadores* de las expresiones proposicionales. Los ejemplos precedentes justifican una segunda forma de obtener proposiciones a partir del uso de estos *cuantificadores*: el existencial “Para algún x” y el universal “Para todo x”.

Cuantificador Existencial:

Se escribe “$\exists x$” y se lee de distintas maneras: “Para algún x se verifica”, “Existe un x tal que” o “Para al menos un x se verifica”.

Por ejemplo, si en la oración "Existen alumnos que disfrutan estudiando" representamos a un alumno con la letra x y con P(x) a "x disfruta estudiando", podemos escribir esta oración en forma simbólica usando el cuantificador existencial de la siguiente forma: "$\exists x\ P(x)$".

Cuantificador Universal:

Se escribe "$\forall x$" y se lee de distintas maneras: "Para todo x se verifica", "Para cada x se verifica" o "Para cualquier x se verifica".

Por ejemplo, si en la oración "Todos los ingresantes a la Universidad deben completar su ficha médica" representamos a un ingresante con la letra x, al dominio (Universidad) con la letra U y con R(x): "x debe completar su ficha médica", podemos escribir esta oración en forma simbólica usando el cuantificador universal y expresarla como "$\forall x\ (x \in U \rightarrow R(x))$". Cuando el dominio se define previamente, como en este caso, podemos escribir "$\forall x\ R(x)$".

La expresión en dos variables "$\exists x\ \exists y\ S(x, y)$" se lee "Para algunos x, y se cumple S(x, y)" y la expresión "$\forall x\ \forall y\ S(x, y)$" se lee "Para todos x, y se verifica S(x, y)".

Cuando las expresiones proposicionales están cuantificadas (existencialmente o universalmente) tienen un valor de verdad y se convierten en proposiciones.

$\exists x\ P(x)$, es verdadera si existe al menos un elemento a, que pertenece al dominio, tal que P(a) es verdadera.

$\exists x\ P(x)$, es falsa cuando P(a) es falsa para cualquier a del dominio.

$\forall x\ P(x)$, es verdadera cuando P(a) es verdadera para cualquier sustitución de x por un sujeto a del dominio.

$\forall x\ P(x)$, es falsa cuando al menos se encuentre una sustitución de x por un sujeto a del dominio tal que P(a) es falsa.

Ejemplo 29

Sean las expresiones relacionales P(x), Q(x) y R(x), donde en todos los casos, x se encuentra en el conjunto de los números enteros.

$P(x): x \geq 2$ $\qquad$ $Q(x): |\ x\ | = 2$ $\qquad$ $R(x): x^2 - 5x + 6 = 0$

a) $\exists x\ P(x)$ es verdadera ya que si x se reemplaza por 3, P(3): $3 \geq 2$ es una proposición verdadera.

b) $\forall x\ P(x)$ es falsa porque existe una sustitución para x, por ejemplo, $x = 0$ donde ocurre que P(0): $0 \geq 2$ es una proposición falsa.

c) $\exists x\ (P(x) \wedge Q(x))$ es una proposición verdadera ya que existe $x = 2$ tal que P(2): $2 \geq 2$ y Q(2): $|\ 2\ | = 2$ es verdadera porque es la conjunción de proposiciones verdaderas.

d) $\forall x\ (R(x) \rightarrow P(x))$ es una proposición verdadera ya que: si x se sustituye por 2 o 3, entonces las proposiciones R(2), R(3), P(2) y P(3) son verdaderas, y por consiguiente $R(2) \rightarrow P(2)$ y $R(3) \rightarrow P(3)$ resultan verdaderas. Si x se sustituye por cualquier valor 'a' distinto de 2 y 3, la proposición $R(a) \rightarrow P(a)$ resulta siempre verdadera pues el antecedente del condicional es siempre falso.

1.7.3 Negación de expresiones relacionales cuantificadas

Decimos que R(x) *implica lógicamente* a P(x) cuando la proposición $\forall x\ (R(x) \rightarrow P(x))$ es verdadera. En símbolos: $\forall x\ (R(x) \Rightarrow P(x))$

Dos expresiones relacionales U(x) y V(x) son *lógicamente equivalentes* cuando todas las proposiciones bicondicionales $U(a) \leftrightarrow V(a)$ son verdaderas para cada sustitución de x por un elemento a del universo. En símbolos: $\forall x\ (U(x) \Leftrightarrow V(x))$

Las expresiones cuantificadas son proposiciones, luego podemos negarlas y obtener nuevas proposiciones.

La negación del Cuantificador Existencial se expresa: $\neg\exists x\ F(x) \Leftrightarrow \forall x\ \neg F(x)$. Esta negación se lee: Todo x es tal que no verifica la expresión relacional F(x).

La negación del Cuantificador Universal se expresa: $\neg\forall x\ F(x) \Leftrightarrow \exists x\ \neg F(x)$. Esta negación se lee: Existe algún x que no verifica la expresión relacional F(x).

Ejemplo 30

Proponemos a continuación varios ítems que muestran como usar y expresar la negación de expresiones cuantificadas.

a) Dada $\forall x\ (x \in Z_0^+ \rightarrow x \geq 0)$. Esta proposición es verdadera.

Su negación es $\neg\forall\ x\ (x \in \mathbf{Z}_0^+ \rightarrow x \geq 0) \Leftrightarrow \exists x\ (x \in \mathbf{Z}_0^+ \wedge x < 0)$; proposición que es falsa.

b) La proposición: "Existen ecuaciones de primer grado con coeficientes enteros cuyas soluciones no son enteras" es verdadera, ya que la ecuación $2x - 3 = 0$ tiene como solución $x = 3/2$. Su negación: "Todas las ecuaciones de primer grado con coeficientes enteros tienen soluciones enteras" es una proposición falsa.

c) La proposición: "Todos los números enteros son múltiplos de tres" es falsa. Su expresión, en símbolos, es $\forall x\ F(x)$, siendo F(x): x es un número entero múltiplo de tres. Su negación es una proposición verdadera que se expresa como: "Existen números enteros que no son múltiplos de tres" y en símbolos: $\exists x\ \neg F(x)$.

d) La negación de la proposición $\forall x\ (F(x) \rightarrow G(x))$ es $\exists x\ \neg(F(x) \rightarrow G(x))$ que es lógicamente equivalente a $\exists x\ (F(x) \wedge \neg G(x))$.

e) La negación de la proposición $\forall x\ (F(x) \wedge G(x))$ es $\exists x\ \neg(F(x) \wedge G(x))$ y es lógicamente equivalente a $\exists x\ (\neg F(x) \vee \neg G(x))$.

f) $\neg\exists\ x\ (F(x) \vee G(x)) \Leftrightarrow \forall x\ \neg(F(x) \vee G(x)) \Leftrightarrow \forall x\ (\neg F(x) \wedge \neg G(x))$.

1.7.4 Expresiones que contienen más de un cuantificador

Vamos a aclarar el significado de expresión cuantificada cuando aparecen dos cuantificadores.

Veamos el siguiente ejemplo.

Ejemplo 31

Dadas dos matrices A, B de orden m x n, se verifica la propiedad conmutativa para la adición de matrices que puede expresarse como:

$\forall A\ \forall B\ (A+B = B+A)$. Pero también puede expresarse así: $\forall B\ \forall A\ (A+B = B+A)$.

Definición 1.15

Si P(x, y) es una expresión relacional en dos variables x, y entonces la proposición $\forall x\ \forall y$ P(x, y) es lógicamente equivalente a la proposición $\forall y\ \forall x$ P(x, y). De aquí que podamos escribir que: $\forall x\ \forall y$ P(x, y) $\Leftrightarrow \forall y\ \forall x$ P(x, y)

Podemos simplificar la notación y escribir que:$\forall x\ \forall y$ P(x, y) $\Leftrightarrow \forall x$, y P(x, y)

Ejemplo 32

Para el universo de las matrices de orden n x n, existen matrices A, B que verifican A B = B A. En textos de Álgebra podemos encontrarnos con la expresión "existen matrices de tamaño n x n que conmutan". Esta proposición puede expresarse así: $\exists A\ \exists B$ (A B = B A). Pero también puede expresarse como: $\exists B\ \exists A$ (A B = B A).

Definición 1.16

Si P(x, y) es una expresión relacional en dos variables x, y entonces la proposición $\exists x\ \exists y$ P(x, y) es lógicamente equivalente a la proposición $\exists y\ \exists x$ P(x, y) y podemos escribir que: $\exists x\ \exists y$ P(x, y) $\Leftrightarrow \exists y\ \exists x$ P(x, y). Cuando trabajamos con el mismo cuantificador podemos simplificar la expresión y escribir $\exists x\ \exists y$ P(x, y) $\Leftrightarrow \exists x$, y P(x, y).

Cuando formulamos una proposición con cuantificadores distintos debemos tener especial cuidado porque no sucede lo mismo que en el caso anterior. Estudiaremos las proposiciones $\forall x\ \exists y$ P(x, y) y $\exists x\ \forall y$ P(x, y). La proposición $\forall x\ \exists y$ P(x, y) hace referencia a que "para cada x seleccionado existe un y tal que P(x, y)". Es así que, cada valor seleccionado para la variable x produce una elección de la variable y. La proposición $\exists x\ \forall y$ P(x, y) se refiere a que "existe un valor de x para todos los y tales que P(x, y)".

Ejemplo 33

Cuando decimos "para cada número entero x existe un entero y tal que $x+y = y+x = 0$", nos estamos refiriendo a que "para cada número entero x existe un entero y, tal que $y = -x$, que satisface $x + (-x) = (-x) + x = 0$".

Si ahora consideramos la expresión $\exists x\ \forall y\ (x + y = y + x = 0)$, que se lee "existe un entero x tal que para todos los enteros y, $x + y = y + x = 0$", es falsa ya que una vez elegido *un* entero x, por ejemplo, $x = 2$ el *único* valor de la variable y que satisface la igualdad es -2.

Del ejemplo concluimos que, en general, la proposición $\forall x\ \exists y$ P(x, y) no es lógicamente equivalente a la proposición $\exists x\ \forall y$ P(x, y).

Ejemplo 34

Cuando utilizamos la proposición $\exists x\ \forall y$ P (x, y) nos estamos refiriendo a "que existe un único valor de x para todos los valores de y tales que se cumple P(x, y)". El uso de esta

proposición lo vemos cuando enunciamos la propiedad de existencia del elemento neutro de la suma en los enteros:

"Existe un elemento $0 \in \mathbf{Z}$ tal que para todo $x \in \mathbf{Z}$ se cumple que $0 + x = x + 0 = x$".

Actividad 8

Problema Nº1: Sean las expresiones relacionales: $P(x)$: $x \leq 2$, $Q(x)$: $x\,(x - 1) = 0$. ¿Cuáles son los valores de verdad de las proposiciones dadas si el universo es el conjunto de los números enteros no negativos?

1.1) $\neg P(1)$
1.2) $P(0) \wedge Q(0)$
1.3) $P(2) \wedge Q(2)$
1.4) $\neg [P(0) \wedge Q(2)]$
1.5) $P(5) \rightarrow Q(2)$
1.6) $P(5) \rightarrow \neg Q(2)$

Problema Nº2: Considera las siguientes definiciones:

a) $f: A \rightarrow B$ es una función de A en B si se cumplen estas dos condiciones:

i) todo elemento de A tiene imagen en B.

ii) si dos elementos cualesquiera de A son iguales, sus imágenes también lo son. Esto se escribe como: Para cada elemento de A su imagen es única.

b) $f: A \rightarrow B$ es una función de A en B. Decimos que f es inyectiva si y sólo si para dos elementos distintos cualesquiera de A sus imágenes también lo son.

2.1) Traduce al lenguaje simbólico las definiciones dadas.

2.2) Investiga la definición de función sobreyectiva o suryectiva. Traduce esta definición al lenguaje simbólico.

Problema Nº3: Dado el siguiente enunciado: "Si el producto de dos números reales cualesquiera es cero entonces alguno de ellos es cero".

3.1) Expresa simbólicamente el enunciado.

3.2) Escribe simbólicamente y coloquialmente las proposiciones recíproca, contraria y contra recíproca de la dada.

Problema Nº4: Dada la expresión relacional $A(n)$: $n^2 - n$ y el universo de n que es el dominio $D = \{n: n \in \mathbf{N} \wedge 1 \leq n \leq 20\}$, coloca el valor de verdad de las siguientes proposiciones:

4.1) $\forall n\,(A(n) > 0)$
4.2) $\forall n\,(A(n) \leq A(20))$
4.3) $\exists n\,(A(n) = 0)$
4.4) $\exists n\,(A(n + 1) = A(n))$
4.5) $\forall m\,\forall n\,(A(m) \neq A(n))$, $m \in D$
4.6) $\forall m\,\forall n\,(m \neq n \rightarrow A(m) \neq A(n))$, $m \in D$

Problema Nº5: Expresa la negación de las proposiciones dadas:

5.1) $\exists x\,(F(x) \rightarrow (P(x) \wedge Q(x)))$
5.2) $\forall x\,((F(x) \vee Q(x)) \rightarrow S(x))$
5.3) $\forall x\,(P(x) \wedge (Q(x) \rightarrow R(x)))$

Problema Nº6: Analiza el uso de los cuantificadores en los siguientes enunciados y justifica el valor de verdad en cada uno:

6.1) $\exists 1\ \forall x\ (x\ 1 = 1\ x = x)$ en el universo de los números enteros.

6.2) $\forall x\ \exists y\ (x\ y = y\ x = 1)$ en el universo de los números racionales.

Problema Nº7: Escribe en forma simbólica cada enunciado, niega la expresión simbólica y expresa en forma coloquial dicha negación.

7.1) Todos los miembros del equipo de Fútbol de Argentina son grandes jugadores de fútbol.

7.2) Algunos alumnos no regulares de matemática discreta no aprobaron el último parcial de la asignatura.

7.3) Una condición suficiente para que f sea integrable es que f sea continua.

7.4) Algunos lenguajes libres de contexto no son lenguajes regulares.

Problema Nº8: Escribe en forma coloquial la propiedad referida a números naturales: $\forall n \in \mathbf{N}\ (1 + 2 + 3 + 4 + ... + n = n\ (n + 1)/2\ )$.

1.7.5 Formas implícitas de los cuantificadores

En numerosas ocasiones y aún en numerosos textos de Matemática escuchamos o leemos frases de este tipo:

Los números naturales múltiplos de seis son múltiplos de dos.

Si a y b son números reales entonces $a^2 + b^2 \geq 0$

El 2 es igual a la suma de los cuadrados de dos números naturales.

Estas frases ¿son proposiciones o expresiones relacionales?. En ellas, el uso del cuantificador es *implícito* en lugar de *explícito*. Ellas son proposiciones que están expresadas informalmente. Si colocamos los cuantificadores en forma explícita escribimos:

Todos los números naturales múltiplos de seis son múltiplos de dos.

Si a y b son números reales *cualesquiera* entonces $a^2 + b^2 \geq 0$

Existen dos números naturales tales que 2 es igual a la suma de sus cuadrados.

Simbólicamente podemos expresar las proposiciones cuantificadas como:

$\forall x\ ((x \in \mathbf{N} \wedge (x = 6\ m,\ m \in \mathbf{N})\ ) \rightarrow (\ x = 2\ n,\ n \in \mathbf{N}))$

$\forall a\ \forall b\ ((a \in \mathbf{R} \wedge b \in \mathbf{R}) \rightarrow (a^2 + b^2 \geq 0))$

$\exists m\ \exists n\ (m \in \mathbf{N} \wedge n \in \mathbf{N} \wedge\ 2\ =\ m^2 + n^2)$.

1.7.6 Los dominios finitos. Interpretaciones para los cuantificadores

Para interpretar una frase cuantificada, ésta debe tener la información suficiente para determinar si es verdadera o es falsa. ¿Qué necesitamos tener como información para establecer si la frase es verdadera o es falsa, cuando el dominio es finito? Veamos un ejemplo.

Ejemplo 35

En la frase "Todos los alumnos han aprobado Matemática Discreta" en primer lugar se necesita saber quienes son los alumnos y también conocer quienes han aprobado y quienes no. Analicemos las posibles interpretaciones de $\forall x\ P(x)$ en donde P(x) es la expresión relacional relativa a x: "x ha aprobado Matemática Discreta" y supongamos que hay tres alumnos A, B y C. Así, P(A) significa "A ha aprobado Matemática Discreta". Si A y B han aprobado y C no aprobó tenemos la siguiente tabla:

	A	B	C
P(x)	V	V	F

La frase $\forall x\ P(x)$ es falsa y decimos que $\forall x\ P(x)$ es verdadera si P(A), P(B) y P(C) son *todas* verdaderas. En general la expresión $\forall x\ P(x)$ es verdadera si dado un conjunto finito de n elementos $D = \{d_1, d_2, ..., d_n\}$, las sustituciones $P(d_1), P(d_2), ..., P(d_n)$ son todas verdaderas. Por lo tanto: $\forall x\ P(x) \equiv P(d_1) \wedge P(d_2) \wedge ... \wedge P(d_n)$.

Siguiendo con la interpretación de la misma frase diremos que $\exists\ x\ P(x)$ es verdadera si existe al menos un x para el cual P(x) es verdadera. Teniendo en cuenta que en el ejemplo previo el alumno A aprobó Matemática Discreta, esto es suficiente para que $\exists x\ P(x)$ sea verdadero. En consecuencia, $\exists x\ P(x) \equiv P(d_1) \vee P(d_2) \vee ... \vee P(d_n)$.

Con estas equivalencias podemos negar los cuantificadores y obtener otras equivalencias. En efecto,

$\neg\forall x\ P(x) \equiv \neg(P(d_1) \wedge P(d_2) \wedge ... \wedge P(d_n)) \equiv \neg P(d_1) \vee \neg P(d_2) \vee ... \vee \neg P(d_n) \equiv \exists x\ \neg P(x)$.

1.8 Las expresiones relacionales en los algoritmos

Es objetivo de este apartado conocer cómo se utilizan las expresiones relacionales en el desarrollo de algoritmos. Es éste, además, el mejor momento para estudiar las instrucciones básicas que aparecen en distintos segmentos de algoritmos.

Comenzaremos por definir el concepto de algoritmo.

1.8.1 Algoritmo

Un algoritmo es una secuencia finita de sentencias descriptas paso a paso para realizar alguna tarea.

En la vida cotidiana empleamos diversos procedimientos que son algoritmos en un sentido informal. En matemática también usamos algoritmos como el siguiente que calcula la inversa de una matriz cuadrada.

Sea A una matriz de orden nxn, inversible.

Algoritmo: *Calculo de la inversa de una matriz cuadrada.*

Paso 1: Calcular det (A).

Paso 2: Obtener la matriz adjunta de A, Adj A.

Paso 3: Escribir la inversa de A como $A^{-1} = \frac{1}{\det(A)} \text{Adj } A$

Los algoritmos son muy utilizados en la Ciencia de la Computación como una forma más simple y operativa de describir procesos de cálculos que pueden ser muy complejos, con el objetivo de resolver problemas que generalmente pueden expresarse en términos computacionales.

La idea básica de un *algoritmo* es la de realizar una tarea (debe ser *eficaz*), describir un proceso de manera clara e inequívoca (debe ser *preciso*), definir la secuencia de pasos requeridos para llevar a cabo la tarea, es decir, debe especificar los pasos en orden (debe ser *ordenado*). En ciencias de la computación se exige además que un algoritmo sea *finito* en los siguientes sentidos:

a) El número de pasos debe ser finito, o sea, el algoritmo debe terminar en algún momento.

b) Cada paso debe requerir sólo un tiempo finito y recursos de cómputo finitos.

1.8.2 Variables e Instrucciones

La mayoría de las computadoras constan de cuatro componentes principales: dispositivos de entrada, dispositivos de salida, unidad central de proceso (UCP) o procesador, y la memoria interna.

Los dispositivos de entrada permiten la comunicación entre la computadora y el usuario. Como lo indica su nombre los dispositivos de entrada sirven para introducir datos (información) en la computadora para su proceso. Dispositivos de entrada son: teclados, lápices ópticos, lectores de códigos de barra, mouse (ratón), etc.

Los dispositivos de salida permiten representar los resultados (salidas) del proceso de los datos. Los dispositivos de salida son: pantalla o monitor, impresoras, trazadores gráficos (ploters), reconocedores de voz, etc.

La unidad central de proceso (Central Processing Unit, CPU) dirige y controla el proceso de información realizado por la computadora. La CPU procesa o manipula la información guardada en memoria y también puede almacenar los resultados de estos procesos en memoria para su uso posterior.

La memoria interna o simplemente memoria se utiliza para guardar información. Siempre que una nueva información se almacena en una posición, se destruye cualquier información que en ella hubiera y no se puede recuperar. La dirección de una posición de memoria es permanente y única, el contenido puede cambiar mientras se ejecuta el programa. Para referirnos a las celdas de memoria le asociamos un nombre. Una referencia a una celda de memoria que puede cambiar el valor durante la ejecución de un programa, la llamaremos *variable*.

Una variable es, en realidad, una posición de memoria con nombre, el nombre de la posición se llama Nombre Variable y el valor almacenado en la celda de memoria se llama Valor de la variable. Los nombres de las variables (identificador) responden a reglas de formación de acuerdo al lenguaje que se utiliza, por ejemplo en el lenguaje Pascal Versión 7.0, los identificadores pueden tener una longitud (significativa) de hasta 63 caracteres y deben comenzar siempre con una letra y estar seguidos de hasta 62 símbolos que pueden ser letras o dígitos, y no puede contener blancos y sí el símbolo "_".

Podemos pensar a las variables como celdas, cajas o buzones, cada una de las cuales tiene un nombre y contiene un valor. Por ejemplo: son nombres o identificadores de variables los siguientes:

A AB A4C2 cuenta Suma_Posiciones Contador

Las operaciones realizables se llaman *"Sentencias Ejecutables"* o *"Instrucciones"* (especifican operaciones de cálculos aritméticos y entradas/salidas de datos) y su realización se llama *"Ejecución*". La instrucción de "*asignación"* se utiliza para asignar (almacenar) valores o variables. La operación de asignación se escribe con cualquiera de los siguientes formatos (según el lenguaje en el que se esté trabajando):

Variable ← expresión; variable : = expresión; variable = expresión;

Observación:

Consideraremos el caracter ";" como elemento separador de sentencias y el símbolo ":=" para identificar las asignaciones.

Ejemplo 36

Consideramos la ejecución de las siguientes asignaciones

a) *Cuenta := 1;*

b) *Suma := Contador + 1;*

c) *Ab := Ab + 2;*

Una instrucción de asignación expresa que primero se evalúa la expresión de la derecha del símbolo y el resultado se almacena en la variable que figura a la izquierda del símbolo de asignación.

El ejemplo 36 a) indica que se almacena el valor 1 en la variable *Cuenta*. El ítem b) indica que: al valor almacenado en la variable *Contador* se le suma uno y el valor resultante se almacena en la variable *Suma*; el valor almacenado en la variable *Contador* no resulta afectado por la asignación. La asignación del ítem c) a diferencia de las anteriores, al contenido de la variable *Ab* le suma dos y el resultado lo almacena en la variable *Ab*, eliminando el valor anterior de *Ab* (el valor anterior se pierde).

Pueden combinarse varias sentencias para realizar tareas más complejas. Esto consiste en expresar un conjunto de sentencias organizadas en forma secuencial, que se ejecutará una detrás de otra según el orden de su escritura, de izquierda a derecha y de arriba hacia

abajo. *Una secuencia de sentencias es una colección de sentencias que se ejecutan consecutivamente.*

Ejemplo 37

Sea la secuencia de asignaciones:

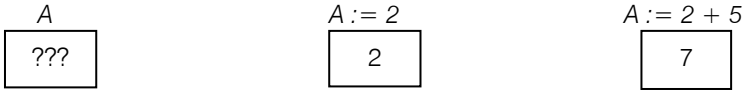
A := 2;

A := A + 5;

Al ejecutar esta secuencia, primero se realiza la primera asignación, que almacena 2 en la variable *A*. Posteriormente se ejecuta la segunda asignación, que almacena en la variable *A* el valor 7.

Es obvio que, en este ejemplo, la segunda asignación destruye el contenido anterior de la variable *A* y lo actualiza (cambia 2 por 7). También queda claro que el orden de las sentencias es fundamental.

Gráficamente, podemos visualizar la siguiente situación.



Actividad 9

Problema Nº1: Investiga como se forman identificadores válidos en los siguientes lenguajes de programación: Pascal, C++, ADA.

Problema Nº2: Indica los valores de las variables *A*, *B* y *C*, luego de ejecutarse la siguiente secuencia de sentencias.

A := 5;

B := 8;

*C := 2 * A – B;*

C := C – A;

B := A + C;

*A := B * C;*

B := B – 1;

1.8.3 Operaciones entrada/salida

Los datos se pueden almacenar en memoria asignándolos a una variable con una sentencia de asignación como las que vimos con anterioridad o con una sentencia de lectura. La sentencia de lectura es la más indicada si se desea manipular diferentes datos cada vez que se ejecuta el programa. La *lectura* de datos (*operación de entrada*) permite asignar valores desde un dispositivo de entrada (por ejemplo, un teclado o una unidad de disco) a distintas variables.

Así mismo a medida que se realizan cálculos en un programa, puede necesitarse visualizar los resultados. Esta operación se conoce como *operación de salida* o *escritura*.

En los algoritmos, las instrucciones de entrada/salida escritas en lenguaje natural (sin individualizar un dispositivo de entrada/salida determinado) son:

Read(lista de variables entrada)

Write(lista de variables salida)

Ejemplo 38

Si se desea leer el valor de una variable *A* desde un dispositivo de entrada se indica:

Read(A)

En caso de necesitar leer el valor de más de una variable, por ejemplo *A* y *B*, podemos optar por cualquiera de las siguientes maneras:

a) *Read(A)*
Read(B)

b) *Read(A,B)*

Si optamos por b) tenemos que tener en cuenta que cuando se lee más de una variable en una misma sentencia éstas se separan con una coma, y que en el momento de ingresar los valores de las mismas se debe respetar el orden en que se piden, en nuestro ejemplo el primer valor que ingrese le será asignado a *A* y el segundo a *B*.

Ejemplo 39

Si deseamos ingresar el número de documento (*NroDcto*) y edad (*Edad*) de una persona la sentencia sería:

Read(NroDcto, Edad)

Al ejecutarse se debe introducir

22624536 27 <enter>

De no respetar el orden de las variables de la lista de variables de entrada, es decir si introducimos: *27 22624536* <enter> estaríamos ingresando datos de una persona cuyo número de documento es *27* y tiene *22624536* años.

Ejemplo 40

Veamos la secuencia de sentencias y el resultado de la ejecución de cada una de ellas:

Sentencias	**Resultado**
Write('esto es un ejemplo')	*esto es un ejemplo*
*Write('********')*	********
A:=1; B:=2; C:=4;	
materia:= 'Matemática Discreta';	
Write(A)	*1*
Write(C)	*4*

Write(A,B,C)	*124*
Write('A')	*A*
Write('A', 'B', 'C')	*ABC*
Write('A', ' ','B', ' ', 'C')	*A B C*
Write('Curso:', materia)	*Curso: Matemática Discreta*
Write('el valor de A es:',A)	*el valor de A es:1*

Actividad 10

Problema Nº1: Indica qué visualizarían las siguientes sentencias de ser incorporadas como instrucciones finales de la secuencia del problema Nº2 de la actividad anterior:

1.1) Write (A)

1.2) Write ('El valor de B es:', B, ' y el valor de C es:', C)

Problema Nº2: Completa el siguiente cuadro, si la ejecución de las acciones es secuencial:

Acción	Contenido de las variables			Visualiza
	A	B	Suma	
A:= 2; B:= 4; Suma:=0;				
Suma := A + B;				
Write(A)				
Write(Suma)				
*Suma := (A+1)*B;*				
B := Suma;				
Write (B)				
A := B + 4;				

Problema Nº3: Escribe una secuencia de sentencias que permitan leer un número, multiplicarlo por 2 y escribir el resultado obtenido.

Problema Nº4: ¿Cuáles son los resultados visualizados por el siguiente bloque de sentencias, si los datos proporcionados son 5 y 8 para A y B, respectivamente?

.....

Read(A,B)

M := 6;

*C := 2 * A – B;*

C := C – M;

B := A + C – M;

```
A := B * M;
Write(A)
B := B – 1;
Write(B, ' ',C)
.....
```

1.8.4 Operadores aritméticos

Los algoritmos evalúan expresiones aritméticas, lógicas, expresiones relacionales, etc. donde las variables y constantes numéricas o alfanuméricas, se combinan o transforman utilizando operaciones y funciones adecuadas. Los lenguajes de programación definen los operadores que manipularán estos datos. A continuación describiremos algunos de ellos.

Operador Aritmético	Nombre	Función
+	Suma	Calcula la suma de dos valores
–	Resta	Calcula la diferencia de dos valores
*	Multiplicación	Calcula el producto de dos valores
/	División real	Calcula el cociente real de dos valores.
div	División entera	Calcula el cociente entero de dos valores enteros.
mod	Resto entero	Calcula el resto o residuo positivo de la división de dos valores enteros.

Estos operadores se combinan con las variables y/o constantes dando lugar a las expresiones aritméticas, como por ejemplo*: 3 + 4 * 5 mod 6, 3 / 4 – 6.*

Podemos observar que en la conformación de las operaciones básicas encontramos dos partes: el operador y los operandos.

Un operador es un símbolo, definido en el lenguaje, que indica la operación que se realizará, y los operandos son los datos sobre los cuales se efectuará la operación.

Decimos que una operación es binaria si opera con dos operandos y es unaria si lo hace con un sólo operando. Son ejemplos de operaciones binarias: la suma, resta, multiplicación, división entera, resto entero.

El orden de evaluación de los operadores aritméticos es el siguiente:

Los operadores **, div , mod* y / tienen la misma prioridad y se evalúan de izquierda a derecha. Por ejempo: 10 mod 2 * 7 = (10 mod 2) *7 = 0 *7 = 0

Los operadores +, – tienen la misma prioridad pero menor la de los operadores anteriores. Por ejemplo: 2 + 5 mod 3 div 4 = 2 + ((5 mod 3) div 4) = 2 + (2 div 4) = 2 + 0 = 2.

Sobre el conjunto de los enteros, los lenguajes de programación incluyen dos funciones denominadas: *Pred* y *Succ* que devuelven el valor anterior y el posterior del dado,

respectivamente. Por ejemplo, el *Pred(67)* es *66* y el *Succ(80)* es *81*. Observar que Pred y Succ son funciones unarias.

Ejemplo 41

Evaluemos algunas expresiones sencillas:

*6 * 2 = 12* *67 – 99 = – 32* *Succ (89) = 90*

7 + 15 = 22 *Pred (45) = 44* *Pred (0) = –1*

A continuación analizaremos más en detalle los operadores *div* y *mod*.

Si *x* es un entero e *y* es un entero positivo, se define *x div y* como el cociente entero de la división de *x* por *y* (con resto no negativo).

El dividendo puede ser positivo, negativo o nulo, mientras que el divisor debe ser un entero positivo.

Por ejemplo:

6 div 2 = 3 *5 div 1 = 5* *–8 div 3 = –3* *19 div 25 = 0* *17 div 5= 3*

Si *x* es un número entero e *y* es un entero positivo, se define *x mod y* como el residuo (no negativo) de la división entera de *x* por *y*. Este resto es siempre un entero no negativo y menor que el divisor.

Por ejemplo:

6 mod 2 = 0 *5 mod 1 = 0* *–8 mod 12 = 4* *25 mod 5 = 0* *19 mod 2 = 1*

Notar que *div* y *mod* operan sólo con datos enteros.

Ejemplo 42

Analicemos algunas situaciones referidas a los operadores div y mod.

a) *6 div 3 = 2*.

b) *–15 div 6 = –3*, porque de este modo el resto, 3, resulta positivo. Pensemos en el hecho de que –15 = (–3) * 6 + 3.

c) *23.0 div 6* da mensaje de *error de tipo de dato*, porque 23.0 es un tipo de dato real.

d) *1 div 3 = 0, 8 mod 2 = 0 y 17 mod 3 = 2*.

e) *–5 mod 3* da como resultado 1, porque –5 = (–2) * 3 + 1.

f) *–10 mod 5* es 0, porque –10 = (–2) * 5.

1.8.5 Operadores lógicos

Las expresiones lógicas pueden combinarse para formar expresiones más complejas utilizando los operadores lógicos*: not, and, or.*

La operación *not* (no) actúa sobre una sola proposición simple u operador, y simplemente niega (o invierte) su valor. La operación *and* (y) combina dos proposiciones simples y produce un resultado TRUE (verdadero) sólo si ambos operandos son verdaderos. La operación *or* (o) es verdadera cuando al menos uno de los operandos lo es. El orden de evaluación de los

operadores lógicos es el siguiente: *not; and; or*. Estos operadores fueron estudiados en las secciones anteriores.

1.8.6 Operadores relacionales

Un operador relacional se utiliza para comparar dos valores. Ellos son:

Operador	Significado
=	Igual
<	Menor que
<= o ≤	Menor o igual que
>	Mayor que
>= o ≥	Mayor o igual que
<>	Diferente

Ejemplo 43

(5 + 7) = 12 es verdadero.

(6 – 12 *mod* 2) > 4 es verdadero ya que 6 es mayor que 4.

not(2 < 6) es falso ya que 2 es menor que 6.

(3 > 1) *and* (2 > 3) es falso ya que (2 > 3) es falso.

(3 > 1) *or* (2 > 3) es verdadero ya que (3 > 1) es verdadero.

(3 > 1) *and* (2 < 3) es verdadero.

not('A'='B') es verdadero ya que 'A'='B' es falso.

(5 *mod* 2) = 0 es falso ya que 5 *mod* 2 = 1.

(5 < 5) or not(2 < 5) es falso.

Reglas de precedencia

Con expresiones aritméticas o relacionales más complejas es preciso establecer prioridades en la evaluación de los operadores.

El nivel de precedencia de *todos* los operadores vistos hasta el momento es el siguiente:

1. ( )
2. *not*
3. * / *div mod and*
4. + – *or*
5. = <> < <= > >=

Ejemplo 44

Considerando las siguientes asignaciones: *Si := True*, *No := False*, *Grande := 7*, *Pequeño := 3*, la instrucción:

a) *Write (not Si or No)* tiene como salida *False*.

b) *Write (No and not No)* produce *False* como salida.

c) *Write (Pequeño + Grande mod Pequeño > Pequeño div Grande)* produce la salida *True.*

d) *Resultado := Si or No and not No or Si*, asigna a *Resultado* el valor *True*.

Los operadores div y mod en los algoritmos

Las operaciones *div* y *mod* resultan útiles a la hora de especificar condiciones en los algoritmos. Veamos algunas de ellas en el siguiente ejemplo.

Ejemplo 45

a) La expresión *X mod 2 = 0* nos permite expresar que el número X es par.

b) Para expresar que el número X es divisible por 10, podemos emplear la expresión lógica: *X mod 10 = 0.*

c) Para expresar que el número Y es impar y menor que 100, podemos utilizar: *(Y mod 2 = 1) and (Y < 100).*

d) Para expresar que el número X, menor que Z, es múltiplo de 4, usamos la expresión lógica: *(X < Z) and (X mod 4 = 0).*

e) Para expresar que el número X es múltiplo de Y, usamos: *X mod Y = 0.*

f) Para comprobar si en un número X, positivo y de más de 3 cifras, la cifra de la unidad de mil es múltiplo de Y, podemos emplear la expresión lógica:

(X >1000) and ((X div 1000 mod 10 mod Y) = 0).

Este último ítem nos muestra como podemos usar el *div* y el *mod* para evaluar si una determinada cifra del número cumple una propiedad específica.

Actividad 11

Problema Nº1: Completa los resultados de la siguiente operatoria:

1.1)

5 mod 2 =	*134 mod 10 =*	*–5 mod 3 =*	*22 mod 5 =*
–13 mod 2 =	*65 mod 15 =*	*2 mod 5 =*	*27 mod 10 =*

1.2)

5 div 2 =	*134 div 10 =*	*–5 div 3 =*	*22 div 5 =*
–13 div 2 =	*65 div 15 =*	*2 div 5 =*	*27 div 10 =*

Problema Nº2: Determina el valor de las siguientes expresiones si A = 2, B = 6, C = 3, T = True, F = False.

2.1) *not(A < B) and (B < > 0) or (A = B – 4)*

2.2) *A + 2 * B – B/C + 7*

2.3) *T and (B < > (A + 4)) or not F*

Problema Nº3: Para las asignaciones *Si := True*, *No := False*, *Grande := 10*, *Pequeño := 6*, determina cuáles de las siguientes expresiones son legales y, en este caso calcula su valor.

3.1) *Si and Grande mod Pequeño.*

3.2*) not Grande – 1 <> Pequeño + 1.*

3.3) *Grande mod Pequeño and Si or not(Si).*

3.4) *(Pequeño = Grande) or Si and No.*

Problema Nº4: Utiliza paréntesis para indicar el orden de evaluación correcto de los operadores en las siguientes expresiones:

4.1) *2 – 89 * 12 + 3*

4.2) *Succ(45) * 6 + 46 mod 3 div 8*

4.3) *98 * (4 div 2) mod 3*

4.4) *Succ(–28 + 2 * 3 + 25 mod 5)*

4.5) *–6 mod 2 + 5647 div 8*

Problema Nº5: Evalúa las siguientes expresiones:

5.1) *9 div 5 – 5 * 7 + 23 mod 2*

5.2) *(–35 + 27 * 3) mod 3 div 6*

5.3) *–8963 div 10 mod 10*

5.4) *2 – Pred(3 * 4 mod 5 * 7)*

1.8.7 Estructuras de Control

En esta sección estudiaremos otras sentencias presentes en los algoritmos. Estas sentencias nos muestran las distintas formas en que los algoritmos pueden tomar decisiones y controlar sus propias acciones. Comenzaremos analizando la estructura de control denominada de "decisión o selección" y continuaremos por estructuras de control iterativas, tales como las estructuras "*WHILE*", "*FOR*", "*REPEAT*":

I - Estructura de decisión o de selección

Esta instrucción permite decidir si se ejecutan o no ciertas sentencias. Esta instrucción condicional tiene el siguiente formato:

```
IF    (expresión lógica)   THEN
                Sentencia1
          ELSE
                Sentencia2
ENDIF
```

Esta estructura de control permite optar por una u otra sentencia a ejecutar de acuerdo al valor que tome la expresión lógica (condición). Primero se evalúa la expresión lógica, si esta

toma el valor verdadero (true) se ejecutará la Sentencia1, en caso contrario, es decir si toma el valor falso (false), se ejecutará la Sentencia2.

Ejemplo 46

Sea la expresión lógica P(X): $X > 0$ y consideramos las siguientes sentencias,

```
A := 0;
Read (X)
IF (P(X)) THEN
    A := A + 4;
   ELSE
      A := A * 5;
ENDIF
```

Si el valor que está almacenado en X es positivo entonces, al valor contenido en la variable A se le suma 4 y si el valor que está almacenado en X es negativo o cero, al valor contenido en la variable A se lo multiplica por 5.

Ejemplo 47

Sea P(X,Y): (X > 0) and (Y > 0) y la secuencia:

```
Read (X, Y)
Resultado := 0;
IF (P(X,Y)) THEN
    Resultado := X mod Y;
    ELSE
      Resultado := X * Y;
ENDIF
```

Si ejecutamos cada una de las instrucciones de este trozo de programa después de ingresar, por ejemplo, para X el 2 y para Y el 5, el valor final de la variable *Resultado* es el resto de dividir 2 por 5, esto es, 2. Pero, si ejecutamos cada una de las instrucciones habiendo ingresado –5 para X, 5 para Y, el valor final de la variable *Resultado* es el producto –25.

Podemos notar además que la asignación *Resultado := 0* podría haber sido cualquier otra sin que por ello se modifique el valor final de la variable *Resultado*.

Omisión de la cláusula *ELSE*

En algunos casos se desea que una determinada sentencia (instrucción) se ejecute si una condición es verdadera y no realizar sentencia alguna en caso de que sea falsa. Este es el caso en el que se omite la cláusula *ELSE*.

Ejemplo 48

Siendo P(X): X mod 2 = 0 y la secuencia de sentencias,

```
Read (X)
```

```
IF  (P(X))   THEN
   Write(X, 'es múltiplo de 2')
ENDIF
Write ('terminó la secuencia')
```

Si se ingresa 4 para X, la salida es: 4 es múltiplo de 2, terminó la secuencia.
En cambio si se ingresa 5 para X, la salida es: terminó la secuencia.

Actividad 12

Problema N°1: Las variables enteras M y N reciben los valores 2 y 5 respectivamente, durante la ejecución de cierto programa en lenguaje Pascal. Durante la ejecución del programa se encuentran los siguientes *enunciados sucesivos* o instrucciones secuenciales. ¿Cuáles son los valores de M y N después de ejecutar cada uno de estos enunciados?

1.1)
```
IF  (M – N = 5) THEN
        M := M – 2;
   ELSE
        N := N – 2;
ENDIF
```

1.2)
```
IF ((2 * M = N) and (N div 4 = 1)) THEN
        N := 4 * M – 3;
   ELSE
        N := N – 2;
ENDIF
```

1.3)
```
IF (( N < 8) or (M div 2 = 2)) THEN
        N := 2 * M;
  ELSE
        M := 2 * N;
ENDIF
```

1.4)
```
IF ((M < 10) and (N div 6 = 0)) THEN
        M := M – N – 5;
ENDIF
```

1.5)
```
IF ((2 * M = N) or (N Div 2 = 1)) THEN
        M :=  M + 5;
ENDIF
```

Problema Nº2: Para cada segmento de programa contenido en los ejercicios siguientes, determina el número de veces que se ejecuta el enunciado X := X + 1.

2.1)
```
I := 1;
IF  ( (I < 2) or ( I > 0))  THEN
        X := X + 1;
   ELSE
        X := X+ 2;
ENDIF
```

```
2.2) I := 2;
     IF  ((I < 0) and (I >1) or (I  = 3))  THEN
               X := X + 1;
       ELSE
               X := X + 2;
     ENDIF
```

Problema Nº3: Escribe una secuencia de sentencias que permitan leer un número N e indicar si es mayor que 10 o negativo, visualizando un mensaje en ese sentido.

II - Estructuras de control iterativas

Como muchos inventos que ahorran trabajo, lo que mejor hacen las computadoras es realizar tareas repetitivas. Las manipulaciones repetitivas de datos tienen un patrón, y éste puede aprovecharse estableciendo una estructura de control repetitiva adecuada.

El proceso de ejecutar repetidamente el mismo bloque de instrucciones, efectuando las mismas acciones en el mismo orden se denomina ITERACIÓN.

Una secuencia de sentencias que se ejecuta varias veces es un ciclo, y el acto de ejecutar repetidamente esos enunciados se denomina iteración. La idea es establecer un ciclo que se ejecute repetidamente mientras se siga cumpliendo alguna condición, o que se ejecute un número preestablecido de veces, o que se ejecute hasta que alguna condición termine la repetición. Una sentencia iterativa también se denomina *bucle*. Los lenguajes de programación, en general, cuentan con tres estructuras de control repetitivas:

La estructura *WHILE*

Esta estructura se puede describir en español diciendo “mientras se cumpla la condición establecida, sigue repitiendo esta tarea”.

Es una instrucción iterativa que engloba una secuencia de sentencias que se escribe una sola vez. El formato es:

```
WHILE   (expresión lógica)   DO
   Secuencia de Sentencias
ENDWHILE
```

Cuando la sentencia *WHILE* se ejecuta, primero se evalúa la expresión lógica (condición). Si ésta es falsa, ninguna sentencia perteneciente al cuerpo del bucle se ejecuta y el programa continúa en la siguiente instrucción después del bucle. Si la expresión lógica es verdadera se ejecuta la secuencia de sentencias definidas dentro del bucle (es decir el cuerpo del bucle) y se evalúa de nuevo la expresión lógica.

Este proceso se repite mientras que la expresión lógica permanezca verdadera. Cuando ésta sea falsa la instrucción *WHILE* concluye y el programa continúa en la instrucción inmediata siguiente.

Ejemplo 49

Consideremos la expresión lógica P(X): $X > 0$ y la secuencia

```
Read (X)
Suma := 0;
WHILE  (P(X))  DO
       Suma :=  Suma + X mod 10;
       X :=  X div 10;
ENDWHILE
```

Analizamos las siguientes situaciones:

1) Supongamos que X es positivo, por ejemplo $X = 34$, entonces:

P(34) es verdadero, luego, a Suma se le asigna el valor 4 y a 'X' se le asigna el valor 3.

P(3) es verdadero; luego, a Suma se le asigna el valor $4 + 3 = 7$ y a 'X' se le asigna 0.

Como P(0) es falso, termina el ciclo. El cuerpo del bucle se ejecutó dos veces.

2) Supongamos que X es nulo ($X = 0$), entonces:

P(0) es falso, de modo que las instrucciones que se encuentran dentro del ciclo no se ejecutan, y el algoritmo continúa con la instrucción inmediata siguiente al ciclo. Lo mismo ocurriría para $X < 0$.

Ejemplo 50

Siendo P(I, N) : $I \leq N$, consideramos la siguiente secuencia,

```
N  := 5;
I  := 0;
X  :=  0;
WHILE  (P(I, N))  DO
   X  :=  X + 2;
   I  :=  I + 1;
ENDWHILE
```

Si realizamos un cuadro con los valores de las variables que intervienen en el mismo tendremos que:

	N	**I**	**X**
Valores iniciales	5	0	0
Iteración 1	5	1	2
Iteración 2	5	2	4
Iteración 3	5	3	6
Iteración 4	5	4	8
Iteración 5	5	5	10
Iteración 6	5	6	12

El ciclo de instrucciones fue ejecutado seis veces y los valores finales de X e I son respectivamente 12 y 6. La estructura *WHILE* puede ejecutar el conjunto de instrucciones, ninguna, una o más veces.

La estructura *REPEAT*

A veces requerimos por lo menos una iteración de un proceso que debe continuar hasta que ocurra algún acontecimiento, y para ello se necesita una estructura cíclica que siempre realice, por lo menos, una vuelta, probando al final del ciclo si es preciso realizar otra iteración. Tales estructuras se denominan ciclos con la prueba al final y generalmente se instrumentan con la estructura:

REPEAT
Secuencia de Sentencias
UNTIL (expresión lógica)

La sentencia *REPEAT* especifica un bucle condicional que se repite hasta que la expresión lógica se hace verdadera.

La expresión lógica se evalúa al final del bucle después de ejecutarse el cuerpo del bucle. Si la expresión es falsa, se repite el bucle. Si es verdadera el programa continúa con la siguiente sentencia a *UNTIL*.

Ejemplo 51

Vamos a determinar los valores finales de las variables de la siguiente secuencia, considerando que P(B): $B > 50$;

```
A := 1;
B := 0;
REPEAT
   B := 2 * A – 2;
   A := A + 3
UNTIL (P(B))
```

Un cuadro similar al del ejemplo previo nos puede resultar bastante útil para tener la información deseada:

	A	B	Condición
Valores iniciales	1	0	
Iteración 1	4	0	Falsa
Iteración 2	7	6	Falsa
Iteración 3	10	12	Falsa
Iteración 4	13	18	Falsa
Iteración 5	16	24	Falsa
Iteración 6	19	30	Falsa
Iteración 7	22	36	Falsa
Iteración 8	25	42	Falsa
Iteración 9	28	48	Falsa
Iteración 10	31	54	Verdadera

La estructura *FOR*

En muchas ocasiones se puede desear un bucle que se ejecute un número determinado de veces. Esta estructura se utiliza cuando se identifica que un conjunto de sentencias se debe ejecutar un número finito de veces.

```
FOR ..... TO ..... BY ..... DO
   Secuencia de Sentencias
ENDFOR
```

El ciclo *FOR* requiere la especificación de la cantidad de veces que se deben ejecutar dichas sentencias. Para ello es necesario que se especifique un valor inicial y un valor final para controlar el ciclo; las iteraciones de esta estructura se controlan por una variable cuyo valor representa a un contador y se denomina variable de control o variable de índice.

Explícitamente la estructura se describe:

```
FOR (variable de control := valor inicial) TO (valor final) BY (incremento) DO
   Secuencia de Sentencias
ENDFOR
```

Cuando se ejecuta la sentencia *FOR* a la *variable de control* se le asigna el valor inicial; al llegar a la sentencia *ENDFOR* se verifica si el valor de la *variable de control* incrementada según lo indique *incremento* es menor o igual que el de la variable final; en caso afirmativo, se incrementa el valor de la *variable de control*, y vuelve a ejecutarse el bucle hasta que la evaluación antes indicada resulte falsa.

Ejemplo 52

En este ejemplo se omite el valor de incremento, pues se supone que el mismo es uno.

```
FOR Control : = 1 TO 5 DO
        Write('aa')
        Write('bb')
ENDFOR
```

Al ejecutarse visualizará: aabbaabbaabbaabbaabb. Notemos que la variable de control (Control) se incrementa automáticamente en uno.

Ejemplo 53

La secuencia de instrucciones,

```
FOR I : = 1 TO 10 DO
   IF (I mod 2 = 0) THEN
      Write(I)
      Write(' ')
   ENDIF
ENDFOR
```

Al ejecutarse visualizará los números pares menores que 11: 2 4 6 8 10 y la variable de control, I, dentro del ciclo, tomará los valores: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Observación:

En los casos en que las sentencias controladas se ejecuten hasta que ocurra una determinada situación, se debe utilizar *WHILE* o *REPEAT*.

Hemos mostrado como se puede descomponer un número para luego observar si una determinada cifra cumple o no cierta propiedad.

A menudo, en la práctica, se presentan situaciones más complejas. La resolución de las mismas requiere del diseño y ejecución de un algoritmo. El siguiente ejemplo nos ilustra algunas de estas situaciones.

Ejemplo 54

Para determinar la cantidad de cifras de un entero positivo N que satisfacen alguna propiedad, se requiere de los siguientes procesos simples: descomponer el número en cada una de sus cifras y controlar cada una de ellas determinando si cumple o no con la propiedad solicitada. Además se deberá contar los casos en que se verifique el cumplimiento de dicha propiedad.

De modo general, el algoritmo debe realizar las siguientes tareas:

```
Algoritmo Contar_Cifras_con_Propiedad
Comenzar
   Repetir
      Mirar cada cifra
      Determinar si la cifra satisface la propiedad
      Si la satisface, se deberá contar
   Hasta controlar todas las cifras.
   Informar resultado
Fin.
```

a) El siguiente algoritmo determina la cantidad de cifras pares que posee un entero positivo N.

```
Algoritmo Contar_Cifras_Pares
BEGIN
   Read(N)
   Resultado := 0;  X := N;
   WHILE  ( N <> 0 )  DO
      IF  ( ( N mod 10 ) mod 2 = 0 ) THEN
               Resultado := Resultado + 1;
      ENDIF
      N := N div 10;
   ENDWHILE
   Write('El número ', X, ' posee ', Resultado, ' cifras pares.')
END
```

b) El siguiente algoritmo determina la cantidad de cifras múltiplo de tres que posee un número entero positivo N.

```
Algoritmo Contar_Cifras_MúltiploDeTres
BEGIN
   Read(N)
   Resultado := 0;  X := N;
   WHILE  ( N <> 0 )  DO
      IF  ( ( N mod 10 ) mod 3 = 0 )  THEN
               Resultado := Resultado + 1;
      ENDIF
      N := N div 10;
   ENDWHILE
   Write('El número ', X, ' posee ', Resultado, ' cifras múltiplo de tres.')
END
```

En general, para diseñar un algoritmo que determine la cantidad de cifras de un entero positivo N que son múltiplos de algún número entero positivo Y, debemos introducir dos modificaciones en los algoritmos anteriores: ingresar el número Y, es decir, debemos agregar la instrucción de entrada *Read*(Y), y luego modificar la sentencia de la estructura IF por la expresión *((N mod 10) mod Y = 0)*. El algoritmo resulta:

```
Algoritmo Contar_Cifras_MúltiploDeY
BEGIN
   Read(N, Y)
   Resultado := 0;
   WHILE  ( N <> 0 )  DO
      IF  ( ( N mod 10 ) mod Y = 0 )  THEN
         Resultado := Resultado + 1;
      ENDIF
      N := N div 10;
   ENDWHILE
   Write('El número ',N,' posee ',Resultado,' cifras que son múltiplos de ', Y)
END
```

Actividad 13

Problema Nº1: Sigue los siguientes trozos de programa informando los valores finales de las variables y las veces que se ejecuta cada ciclo.

1.1)
```
I := 1;   X := 0;
WHILE (I ≤ 15) DO
         X := X + I;
          I := I + 1;
ENDWHILE
```

1.2)
```
A := 1;  B := 1;
REPEAT
         B := 2 * B – A;
```

```
            A := A + 2;
        UNTIL  ( B < –25)
1.3)    I :=1;
        J :=1;
        WHILE (( I <2 and  J <5 ) or ( I  + J = 5 ))   DO
                I := I  + 2;
                J := J  + 1;
        ENDWHILE
```

Problema Nº2: Para cada segmento de programa, determina el número de veces que se ejecuta la instrucción *X:= X + 1*

```
2.1)    I := 1;
        WHILE ( I <3 ) DO
                X := X + 1;
                I := I + 1;
        ENDWHILE
2.2)    I :=  1;
        WHILE  ((I  > 0) and ( I <4 )) DO
                IF ( I <2)  THEN
                        I := I + 1;
                    ELSE
                        I := I + 2;
                ENDIF;
                X := X + 1;
        ENDWHILE
2.3)    I := 1;
        X := 1;
        WHILE ((I > 0) and ( I < 4 )) or (I = 0) DO
                X := X + 1;
                 I := (I + 1) mod 2;
        ENDWHILE
2.4)    I := 1;  X := 0;
        WHILE (I < 0) DO
                I := I –1;
                X := X + 1;
        ENDWHILE
```

Problema Nº3: Para cada segmento de programa, determina el número de veces que se ejecuta la instrucción *X := X + 1*

```
3.1)    FOR I : = 1 TO N DO
                FOR J : = 1 TO N DO
                        X := X + 1;
                ENDFOR
        ENDFOR
3.2)    FOR I : = 1 TO N DO
                FOR J : = 1 TO I  DO
                        X := X + 1;
```

```
            ENDFOR
      ENDFOR
3.3)  FOR I : = 1 TO N DO
            FOR J : = 1 TO N DO
                  FOR K :=  1 TO N DO
                        X := X + 1;
                  ENDFOR
            ENDFOR
      ENDFOR
3.4)  FOR  I  := 1 TO N–1 DO
            FOR J := N TO I+1 BY –1 DO
                  X :=  X + 1;
            ENDFOR
      ENDFOR
```

Problema Nº4: Sea *X* es un número real. Considera la siguiente secuencia:

```
Read(X)
IF (X > 1800) THEN
      Y := X  + 0.02(X – 1800.0);
ELSE
      Y := X + 0.03X;
ENDIF
Write(Y)
```

Explicita los valores de *Y* para

4.1) $X = 1200$ 4.2) $X = 1800$ 4.3) $X = 2000$

Problema Nº5: Un segmento de programa contiene un lazo *REPEAT-UNTIL* estructurado como sigue:

```
      REPEAT
     ..............
      UNTIL  ( (P(X) and Q(Y)) or not R(W,T) );
```

donde:

P(X): $X <> 0$; *Q(Y):* $Y > 0$; *R(W,T):* $(W > 0)$ and $(T = 3)$;

Determina si concluye o no el bucle para las siguientes asignaciones de las variables *X*, *Y*, *W*, *T*.

5.1) $X = 7$, $Y = 2$, $W = 2$, $T = 3$.

5.2) $X = 0$, $Y = 1$, $W = –2$, $T = 2$.

5.3) $X = 0$, $Y = –5$, $W = 1$, $T = 3$.

5.4) $X = 1$, $Y = –5$, $W = 1$, $T = 3$.

5.5) $X = 0$, $Y = 1$, $W = 7$, $T = 4$.

Problema Nº6: Las siguientes instrucciones se encontraron en un trozo de un algoritmo,

```
X := 0;
Y := 0;
```

```
WHILE (X < 10) DO
   X := X + 1;
   Y := Y + X;
ENDWHILE
Y := Y/2;
Write(Y)
```

6.1) Explicita el valor final de *Y*.

6.2) Explicita el valor final de *Y* si a *X* se le asigna al principio el valor 10.

6.3) En cada situación anterior, determina las veces que se ejecuta el ciclo *WHILE* y explica la tarea que realiza este trozo de algoritmo.

6.4) Escribe un trozo de algoritmo que realice la misma tarea utilizando la estructura *FOR....TO....BY.....DO.*

Problema Nº7: Declara la tarea que resuelve el siguiente algoritmo. La instrucción Read (L(J)), con J = 1, 2,....., N tiene el efecto de permitir ingresar una lista de valores numéricos, de a uno por vez. L(1) es el primer valor de la lista, L(2) es el segundo y L(N) es el enésimo valor de la lista. En este caso la lista contiene N números enteros.

```
BEGIN
   S := 0; Read (N)
   FOR J := 1 TO N DO
      Read  L(J)
      S :=  S + L(J);
   ENDFOR
   P := S / N;
   Write(S, N, P)
END
```

Problema Nº8: Declara la tarea que resuelve el siguiente algoritmo, sabiendo que N $\in \mathbb{Z}^+$:

```
BEGIN
   F:= 1; Read (N)
   IF  ((N = 1) or (N = 0))  THEN
        F:= 1;
       ELSE
          J:= 2;
          WHILE ( J ≤ N)  DO
                F:= F * J;
                J:= J + 1;
          ENDWHILE
   ENDIF
   Write(N, F)
END
```

Problema Nº9: Sigue los algoritmos siguientes y completa la tabla adjunta con los valores de las variables y de la expresión condicional. Informa cuál es la tarea que realiza cada algoritmo.

9.1) N es un número natural. Sugerimos que comiences con N = 562 y luego con N = 23.

```
Algoritmo uno
BEGIN
   M := 0; Read(N)
   WHILE ( N <> 0 ) DO
      M  := M + 1;
      N  := N div 10;
   ENDWHILE
   Write(M)
END
```

N	M	Condición	Escribir
562	0	T	
56	1		

9.2) En este segundo algoritmo, N es un número natural.

```
Algoritmo dos
BEGIN
   Read(N)
   C := 0; D := 0;
   WHILE ( N <> 0 ) DO
      D :=  N mod 10;
      IF  ( D mod 2 = 0 ) THEN
         C := C + 1;
      ENDIF
      N := N div 10;
   ENDWHILE
   Write(C)
END
```

N	C	D	Condición

Problema Nº10: Declara la tarea que resuelven cada uno de los siguientes algoritmos.

10.1) Considera que N es un número entero positivo.

```
BEGIN
   Read (N)
   X := N;
   Y := 1;
   WHILE (Y <> N) DO
      X := X + N;
      Y := Y + 1;
   ENDWHILE
   Write (X)
END
```

10.2) Considera que M es un número entero positivo y N es un número real.

```
BEGIN
   Read (N, M)
```

```
    K := 1;
    WHILE (M > 0) DO
        K := K * N;
        M := M – 1;
    ENDWHILE
    Write(K)
END
```

10.3) Considera que N es un número entero positivo.

```
BEGIN
Read (N)
Divisor := 2;
WHILE (Divisor  <=N div 2) DO
        IF (N mod Divisor =0) THEN
                Write(Divisor)
        ENDIF
        Divisor := Divisor +1;
ENDWHILE
END
```

1.9 Reglas de inferencia para LPC

La función principal de la lógica de proposiciones es proporcionar reglas de inferencia o razonamientos. La expresión de nuestros pensamientos o la expresión de una idea se realiza a través de la inferencia de una conclusión a partir de ciertas premisas. El conocimiento científico se obtiene mediante conexiones lógicas y conclusiones, es decir, se obtiene una conclusión a partir de otros conocimientos. Un razonamiento es básicamente una implicación lógica.

El antecedente de la implicación es una conjunción de fbfs que expresan un conocimiento ya obtenido; cada fbf es una *premisa.* El consecuente de la implicación es la *conclusión* y se obtiene a partir de la conjunción de las premisas, como un conocimiento nuevo.

La importancia de los razonamientos en las ciencias es advertida por cualquier estudiante de matemática, física o, incluso en la jurisprudencia, entre otras. Se dispone de enunciados que al menos, transitoriamente, no se discuten: los postulados de la geometría, las leyes de un código civil o penal, etc., a partir de los cuales, considerados como premisas, obtenemos conclusiones que proporcionan nuevos conocimientos. Por ello, es conveniente establecer una serie de conceptos y técnicas que nos permitan discriminar entre razonamientos correctos e incorrectos.

A continuación identificaremos distintas formas de razonamientos y analizaremos sus expresiones en diversos ejemplos

Definición 1.17

El *razonamiento* es una implicación cuyo antecedente es la conjunción de un número finito de proposiciones llamadas *premisas* y cuyo consecuente es la *conclusión*.

Tanto las premisas como la conclusión son fbfs y consecuentemente un razonamiento es una fbf. Simbolizamos las premisas con p_1, p_2, p_3, ..., p_n y la conclusión con la letra c.

Entonces, en símbolos, un razonamiento es expresado utilizando la implicación o condicional, en estos términos: $p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_n \rightarrow c$, o en forma vertical de la siguiente manera:

$$\begin{array}{l} p_1 \\ p_2 \\ p_3 \\ ... \\ \underline{p_n} \\ \therefore c \end{array}$$

Nos interesa conocer *la validez de un razonamiento*, es decir, cuáles son las condiciones para que la conclusión se deduzca de una o más proposiciones. La corrección (validación) o no de un razonamiento se decide a partir de la suposición de que las premisas son verdaderas ¿por qué?

La lógica nos permite discriminar entre razonamientos correctos o válidos e incorrectos o no válidos.

En esta sección abordaremos la siguiente cuestión ¿qué significa que un razonamiento sea válido?

Un razonamiento es válido cuando el condicional es tautológico, esto es, si las premisas son verdaderas entonces la conclusión debe ser necesariamente verdadera y si las premisas son falsas no importará lo que ocurre con la conclusión ya que el condicional será verdadero y por consiguiente el razonamiento será válido.

En términos lógicos, un razonamiento válido tiene esta forma:

$$p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_n \Rightarrow c$$

Un razonamiento válido también se denomina *regla de inferencia*.

Son ejemplos de razonamientos válidos:

a) Si un triángulo es rectángulo y un ángulo agudo mide 30^o entonces el otro ángulo agudo mide 60^o.

b) Si $n^2 + 3n - 4 = 0$ y $n \neq 1$ entonces $n = -4$.

c) Si un número no negativo a, divide al producto pq de enteros no negativos y el máximo común divisor de a y p es 1 (a y p son primos relativos) entonces a divide a q.

d) Si una palabra inglesa de tres letras no comparte ninguna letra con LEG (“pierna”), tiene una única letra en común con ERG (“ergio”) pero no en el sitio correcto, tiene una única letra en común con SIR (“caballero”) y está en el sitio correcto, tiene una letra en común con SIC (“textualmente”) pero no en el sitio correcto y tiene una única letra en común con AIL (“doler”) pero no en el sitio correcto, entonces es la palabra CAR (“auto”). (Ejemplo basado en el juego 11 de “100 Games of Logic”, de Pierre Berloquin, incluido por David Perkins en “La bañera de Arquímedes y otras historias del descubrimiento científico”).

Cuando una conclusión se obtiene a partir de un conjunto de premisas, usando reglas aceptadas de razonamiento, entonces tal proceso de derivación se conoce como *deducción o prueba formal*. En una prueba formal, cada regla de inferencia que se usa en alguna etapa de la deducción, se manifiesta.

A continuación estudiaremos en detalle algunas de las reglas de inferencia que se aplican habitualmente.

1.9.1 Modus Ponens

El razonamiento $p \wedge (p \rightarrow q) \rightarrow q$ contiene dos premisas.

p_1: p y p_2: $p \rightarrow q$ y una conclusión c: q

Tabla 1.24

p	q	$p \rightarrow q$	$p \wedge (p \rightarrow q)$	$p \wedge (p \rightarrow q) \rightarrow q$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

La implicación: $p \wedge (p \rightarrow q) \rightarrow q$ es una tautología, con lo cual, este razonamiento denominado Modus Ponens es válido.

Como en el razonamiento propuesto intervienen solamente dos proposiciones atómicas, la tabla de verdad es sencilla; pero el análisis de un razonamiento mediante tabla de verdad se complica cuando aparecen más de tres variables proposicionales.

De cualquier forma, es interesante ver que la tabla de verdad puede abreviarse eliminado los renglones en que intervienen premisas falsas. Esto es así porque si una premisa es falsa, la conjunción es falsa; entonces el antecedente del razonamiento resulta falso y la implicación es verdadera. Por ello, interesan los renglones en que todas las premisas son simultáneamente verdaderas. En este ejemplo, es necesario entonces completar sólo la última fila.

Veamos el siguiente ejemplo: Si dos ángulos interiores α y β de un triángulo T suman 90^o entonces el triángulo es rectángulo. $\alpha + \beta = 90^o$. Luego, T es rectángulo.

Este razonamiento surge de considerar:

p: dos ángulos interiores α y β de un triángulo T suman 90^o

q: el triángulo es rectángulo

Luego la proposición puede expresarse como $((p \rightarrow q) \wedge p) \rightarrow q$

La validez del razonamiento proviene de las reglas Modus Ponens y conmutatividad de la conjunción.

1.9.2 ModusTollens

Consideremos el siguiente razonamiento:

Si Pedro es bueno, es generoso.

Pedro no es generoso.

Luego, Pedro no es bueno.

Simbólicamente, considerando las variables proposicionales:

p: Pedro es bueno y q: Pedro es generoso

Se tiene: p_1: $p \rightarrow q$; p_2: $\neg q$; c: $\neg p$;

Y el razonamiento es: $((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$.

La tabla de verdad es la siguiente:

Tabla 1.25

p	q	$\neg p$	$\neg q$	$p \rightarrow q$	$(p \rightarrow q) \wedge \neg q$	$(p \rightarrow q) \wedge \neg q \rightarrow \neg p$
0	0	1	1	1	1	1
0	1	1	0	1	0	1
1	0	0	1	0	0	1
1	1	0	0	1	0	1

Deducimos que estamos en presencia de un razonamiento válido, denominado Modus tollens, ya que la tabla de verdad nos muestra que el condicional es una tautología o implicación lógica. También es posible observar, en esta situación, que: si las premisas son verdaderas la conclusión es verdadera.

a) Si dos ángulos interiores de un triángulo T suman 90^o, el triángulo es rectángulo. El triángulo T no es rectángulo. Luego la suma de dos ángulos interiores de T es distinta de 90^o.

b) Si dos matrices cuadradas A y B son similares entonces det (A) = det (B). Pero det (A) $\neq$ det (B). Luego A y B no son similares.

c) Si $2 > 5$ entonces $3 \geq 6$; pero $3 < 6$, luego $2 \leq 5$.

d) Si f: A$\rightarrow$B es inyectiva entonces $\forall$ x, y $\in$ A se cumple que si $x \neq y \Rightarrow f(x) \neq f(y)$.

$\exists x \, \exists y \in A$ tales que , $x \neq y \wedge f(x) = f(y)$. Luego f: A$\rightarrow$B no es inyectiva.

1.9.3 Silogismo hipotético

Consideremos el razonamiento:

Si aumentan los salarios entonces aumentará la demanda de las carnes rojas y si éstas aumentan entonces subirá el índice de inflación. Por lo tanto, si aumentan los salarios aumentará el índice de inflación.

Identificamos las proposiciones y las simbolizamos de la siguiente manera:

p: aumentan los salarios; q: aumentará la demanda de las carnes rojas; r: subirá el índice de inflación

Este razonamiento quedará expresado por:

p_1: $p \rightarrow q$

$\underline{p_2}$: $\underline{q \rightarrow r}$

c: $p \rightarrow r$

Si construimos la tabla de verdad notaremos que $((p \to q) \wedge (q \to r)) \to (p \to r)$ es una tautología. Pero construir una tabla de ocho renglones no es cómodo ni eficiente. Aplicando el Teorema de la Deducción:

$((p \to q) \wedge (q \to r)) \to (p \to r) \Leftrightarrow ((p \to q) \wedge (q \to r) \wedge p) \to r$

Luego

p_1: $p \to q$

p_3: p

Por Modus Ponens, obtenemos q

p_4: q

p_2: $q \to r$

Por Modus Ponens, obtenemos r

El razonamiento es usado en este ejemplo: Si un número es entero entonces es racional, si un número es racional entonces es real. Luego si un número es entero, es real.

1.9.4 Silogismo disyuntivo

Esta regla presenta la siguiente forma:

p_1: $p \vee q$

p_2: $\neg p$

c: q

Si $\neg p$ es una premisa verdadera entonces p es falsa. Como la disyunción $p \vee q$ es verdadera se concluye que q es verdadera. Luego el razonamiento es válido.

Mirando la tabla de verdad, también deducimos la validez del razonamiento ya que la implicación es una tautología:

Tabla 1.26

p	q	$p \vee q$	$\neg p$	$(p \vee q) \wedge \neg p$	$(p \vee q) \wedge \neg p \to q$
0	0	0	1	0	1
0	1	1	1	1	1
1	0	1	0	0	1
1	1	1	0	0	1

Este razonamiento se aplica en el ejemplo:

Si $n^2 + 3n - 4 = 0$ y $n \neq 1$ entonces $n = -4$. La expresión $n^2 + 3n - 4 = 0$ es equivalente a la disyunción $p \vee q$ donde la proposición p es p: $n = 1$; la proposición q es q: $n = -4$ y la premisa $\neg p$: $n \neq 1$. La forma del silogismo disyuntivo $((p \vee q) \wedge \neg p) \to q$ garantiza la conclusión dada.

1.9.5 Dilema constructivo

Esta regla presenta la siguiente forma:

p_1: $p \vee q$

p_2: $p \rightarrow r$

$\underline{p_3: q \rightarrow r}$

c: r

p_2 es una premisa verdadera equivalente a $\neg p \vee r$.

p_3 es una premisa verdadera equivalente a $\neg q \vee r$.

La conjunción $p_2 \wedge p_3$ es verdadera y equivalente a $(\neg p \wedge \neg q) \vee r$. Esta fbf es equivalente a $\neg(p \vee q) \vee r$. Como $p \vee q$ es verdadera, entonces $\neg(p \vee q)$ es falsa. Luego r es verdadera.

1.9.6 Regla de contradicción

Consideremos la proposición $(\neg p \rightarrow F_0) \rightarrow p$.

Si $\neg p$ es verdadera, $\neg p \rightarrow F_0$ es falsa y por lo tanto, toda la proposición es verdadera.

Si $\neg p$ es falsa, $\neg p \rightarrow F_0$ es verdadera. Además p verdadera (dado que $\neg p$ es falsa) y por tanto, toda la proposición es verdadera.

Concluimos que $(\neg p \rightarrow F_0) \rightarrow p$ es un razonamiento válido, puesto que es una tautología.

1.9.7 Dilema destructivo

Esta regla presenta la siguiente forma:

p_1: $p \rightarrow q$

p_2: $r \rightarrow s$

$\underline{p_3: \neg q \vee \neg s}$

c: $\neg p \vee \neg r$

En esta regla, la tabla de verdad necesitaría 16 renglones. Por consiguiente es útil en este punto describir el *proceso de derivación* o *prueba formal* mediante el cual se demuestra que una fórmula es consecuencia válida de un conjunto de premisas.

1.9.8 Prueba formal

Damos dos reglas que permiten manifestar qué implicaciones, reglas de inferencia o leyes lógicas se utilizan en cada etapa de una *prueba formal* o *deducción*. Estas reglas son indicadas con letras **P** (premisa) , **T** (tautología) y se enuncian así:

Regla **P**: Una premisa puede introducirse en cualquier línea de la deducción.

Regla **T**: Una fbf puede introducirse en una línea de una prueba formal si la fbf está tautológicamente implícita en una o más fbfs precedentes de la deducción.

Una *prueba formal* de que un razonamiento determinado es válido, es una sucesión de enunciados cada uno de los cuales es, o bien una premisa del razonamiento dado, o bien se deduce de los enunciados precedentes mediante un razonamiento válido elemental, y tal que el último enunciado de la sucesión es la conclusión del razonamiento cuya validez se quiere demostrar.

Hay una buena cantidad de razonamientos que no pueden probarse usando las reglas de inferencias dadas anteriormente. En algunos casos necesitaremos utilizar las equivalencias

lógicas que nos permitirán la sustitución de cualquier enunciado o parte del enunciado por otra expresión que le sea equivalente.

Así, en el razonamiento $(p \to q) \wedge p \to q$, podemos reemplazar el condicional por una expresión lógica equivalente, obteniendo: $(\neg p \vee q) \wedge p \to q$.

Ejemplo 55

Veamos cómo mostrar la validez del dilema destructivo:

1) $p \to q$ Regla **P**
2) $\neg q \vee \neg s$ Regla **P**
3) $s \to \neg q$ Regla **T** (la línea 3) es lógicamente equivalente a 2))
4) $\neg q \to \neg p$ Regla **T** (la línea 4) es lógicamente equivalente a 1))
5) $r \to s$ Regla **P**
6) $r \to \neg q$ Regla **T** (Silogismo Hipotético por 5) y 3))
7) $r \to \neg p$ Regla **T** (Silogismo Hipotético por 6) y 4))
8) $\neg r \vee \neg p$ Regla **T** (la línea 7) es lógicamente equivalente a 8))

Ejemplo 56

Veamos otro ejemplo que muestra la forma para presentar una prueba formal de:

p_1: $p \vee q$
p_2: $p \to r$
$\underline{p_3: \neg s \to \neg q}$
c: $s \vee r$

1) $p \vee q$ Regla **P**
2) $\neg q \to p$ Regla **T** (las líneas 1 y 2 son equivalentes)
3) $\neg s \to \neg q$ Regla **P**
4) $\neg s \to p$ Regla **T** (silogismo hipotético de 3) y 2))
5) $p \to r$ Regla **P**
6) $\neg s \to r$ Regla **T** (silogismo hipotético de 4) y 5))
7) $s \vee r$ Regla T (por equivalencia con 6))

Ejemplo 57

Mostraremos un método para comprobar que un razonamiento no es válido:

Sean las proposiciones p, q, r, s, t y el razonamiento

p_1: p
p_2: $p \vee q$
p_3: $q \to (r \to s)$
$\underline{p_4: t \to r}$
c: $\neg s \to \neg t$

Para mostrar que el razonamiento no es válido, necesitamos encontrar una valuación que haga falsa la conclusión y donde las premisas sean verdaderas. Como la conclusión es una

implicación, resulta falsa en el único caso en que el antecedente es verdadero y el consecuente es falso, luego el valor de verdad de s debe ser 0 y el de t debe ser 1. Analizando la premisa p_4 como t tiene el valor 1, para que sea verdadera r debe tener el valor 1. Como p es una premisa su valor de verdad debe ser 1 y en consecuencia para que p_2 sea verdadera, q puede tener los valores 0 o 1. Analizando la premisa p_3, para que sea verdadera, como $r \rightarrow s$ es falso, es decir tiene el valor 0, entonces q debe valer 0.

Luego con la valuación v(p) = 1, v(q) = 0, v(r) = 1, v(s) = 0 y v(t) = 1, resultan las cuatro premisas verdaderas y la conclusión falsa.

Ejemplo 58

Dado el enunciado: "*Si las leyes son buenas y su cumplimiento es estricto, disminuirá el delito. Si el cumplimiento estricto de la ley hace disminuir el delito, entonces nuestro problema es de carácter práctico. Las leyes son buenas, luego nuestro problema es de carácter práctico".*

A continuación procedemos a identificar las proposiciones, las simbolizamos y armamos el esquema de razonamiento.

p: las leyes son buenas

q: el cumplimiento de las leyes es estricto.

r: disminuirá el delito.

s: nuestro problema es de carácter práctico.

El razonamiento es:

p_1: $(p \wedge q) \rightarrow r$

p_2: $(q \rightarrow r) \rightarrow s$

p_3: p

c: s

Veamos que es válido:

1) $(p \wedge q) \rightarrow r$ Regla **P**

2) $p \rightarrow (q \rightarrow r)$ Regla **T** (aplicando a 1) el teorema de la deducción)

3) $(q \rightarrow r) \rightarrow s$ Regla **P**

4) $p \rightarrow s$ Regla **T** (silogismo hipotético a 2) y 3)

5) p Regla **P**

6) s Regla **T** (Modus Ponens 4) y 5))

Ejemplo 59

Consideremos el razonamiento:

p_1: $p \rightarrow r \wedge q$

p_2: $\neg q \vee \neg r$

p_3: $\neg q \rightarrow \neg t$

p_4 : $p \vee t$

c: q

El orden o la forma deductiva que se sigue para obtener la conclusión no es única. En este caso anotamos primero todas las premisas y luego seguimos trabajando con las reglas de deducción

1) $p \rightarrow r \wedge q$ Regla **P**
2) $\neg q \vee \neg r$ Regla **P**
3) $\neg q \rightarrow \neg t$ Regla **P**
4) $p \vee t$ Regla **P**
5) $\neg (r \wedge q)$ Regla **T** (de 2) al aplicar ley de De Morgan y conmutativa)
6) $\neg p$ Regla **T** (de 5) y 1) al aplicar Modus Tollens)
7) t Regla **T** (de 6) y 4) aplicando Silogismo Disyuntivo)
8) q Regla **T** (de 7) y 3) al aplicar Modus Tollens y doble negación)

Ejemplo 60

Veamos el siguiente texto donde encontramos un razonamiento realizado por Sherlock Holmes en el libro Estudio en Escarlata (Conan Doyle, A., 2005).

"*Y ahora llegamos a la gran pregunta del porqué. El robo no ha sido el objeto del asesinato, puesto que nada desapareció ¿Fue por motivos políticos o fue una mujer? Esta es la pregunta con que me enfrento. Desde el principio me he inclinado hacia esta última suposición. Los asesinatos políticos se complacen demasiado en hacer sólo su trabajo y huir. Este asesinato, por el contrario, había sido realizado muy deliberadamente, y quién lo perpetró ha dejado huellas por toda la habitación, mostrando que estuvo ahí todo el tiempo*".

¿Es correcta la conclusión a la que se arriba?

Definimos las proposiciones: p: Fue un robo, q: Algo desapareció, r: Fue político, s: Fue una mujer, t: El asesino huyó inmediatamente y u: El asesino dejó huellas por toda la habitación.

El esquema queda planteado de la siguiente forma:

p_1: $p \rightarrow q$
p_2: $\neg q$
p_3: $\neg p \rightarrow r \vee s$
p_4: $r \rightarrow t$
p_5: $u \rightarrow \neg t$
$\underline{p_6}$: $\underline{u}$
c: s

Una prueba o deducción formal es (sugerimos que el lector complete las argumentaciones:

1) $p \rightarrow q$ Regla **P**
2) $\neg q$ Regla **P**
3) $\neg p$ Regla **T** (de 1) y 2) aplicando Modus Tollens)
4) $\neg p \rightarrow r \vee s$ Regla **P**
5) $r \vee s$ Regla **T** (de 4) y 3) aplicando Modus Ponens)

6) $u \to \neg t$ Regla **P**
7) u Regla **P**
8) $\neg t$ Regla **T** (.....)
9) $r \to t$ Regla **T** (.....)
10) $\neg r$ Regla **T** (.....)
11) s Regla **T** (.....)
Finalmente, concluimos que el razonamiento es correcto.

Actividad 14

Problema Nº1: Decide si los siguientes condicionales son razonamientos válidos:

1.1) $((p \to q) \wedge \neg p) \to \neg q$
1.2) $((p \to q) \wedge \neg q) \to \neg p$
1.3) $((p \vee q) \wedge \neg p) \to q$
1.4) $((p \vee q) \wedge \neg q) \to p$
1.5) $((p \wedge q) \wedge (p \to (q \to r))) \to r$
1.6) $((q \to r) \wedge \neg q) \to r$

Problema Nº2: Dados los enunciados, forma las premisas y determina si los razonamientos son o no, válidos:

2.1) Si aumenta el ingreso per cápita del país entonces aumenta el consumo interno. Pero, el consumo interno no aumentó, por consiguiente no aumenta el ingreso per cápita del país.

2.2) Si Juan recibió el telegrama entonces tomó el avión. Juan no tomó el avión, entonces no recibió el telegrama.

2.3) La mayoría de los argentinos consume carne y la mayoría de los frigoríficos pagan impuestos por dicho producto, por tanto la mayoría de los argentinos paga impuesto.

2.4) Si 6 no es par, entonces 5 no es primo; pero 6 es par, por lo tanto 5 es primo.

Problema Nº3: ¿Cuáles de las siguientes proposiciones son verdaderas y cuáles son falsas? Justifica.

3.1) $n = 2$ sólo si $n^2 + 3n - 10 = 0$.
3.2) $n = 2$ si $n^2 + 3n - 10 = 0$.
3.3) $n = 2$ es suficiente para que $n^2 + 3n - 10 = 0$.
3.4) Si $n^2 + 3n - 10 = 0$ entonces $n = 2$.
3.5) Si $n^2 + 3n - 10 = 0$ entonces ($n = 2$ y $n = -5$).
3.6) Si $n^2 + 3n - 10 = 0$ entonces ($n = 2$ o $n = -5$).
3.7) $n^2 + 3n - 10 = 0$ si y solo si ($n = 2$ o $n = -5$).
3.8) $n^2 + 3n - 10 = 0$ si y solo si ($n = 2$ y $n = -5$).

Sugerencia: Asigna p: $n = 2$; q: $n = -5$ y r: $n^2 + 3n - 10 = 0$ o r: $(n - 2)(n + 5) = 0$.
Recuerda que $r \Leftrightarrow p \vee q$.

Problema Nº 4: Muestra que los siguientes razonamientos no son válidos.

4.1) $[\ (p\ \wedge \neg\ q) \wedge [p \rightarrow (q \rightarrow r)]\ ]\ \rightarrow \neg r$

4.2) $[(p\ \wedge q) \rightarrow r] \wedge (\neg\ q \vee r) \rightarrow p$

4.3) $\{p,\ p \rightarrow r,\ p \rightarrow\ (q \vee \neg\ r)\ ,\ \neg\ q \vee \neg s\ \} \models s$

Problema Nº5: Para cada esquema determina el razonamiento válido que lo identifica.

5.1)

p_1: $(p \vee r) \rightarrow \neg t$

$\underline{p_2: p \vee r}$

$\therefore \neg\ t$

5.2)

p_1: p

$\underline{p_2: r \rightarrow \neg p}$

$\therefore \neg r$

5.3)

p_1: $p \vee \neg r$

$\underline{p_2:\ r}$

$\therefore p$

5.4)

p_1: $p \rightarrow r$

$\underline{p_2:\ r \rightarrow q}$

$\therefore p \rightarrow q$

Problema Nº6: ¿Qué conclusión se puede obtener de cada uno de los siguientes conjuntos de premisas?

6.1) Si usted está en Buenos Aires, entonces su reloj señala la misma hora que en La Falda. Usted está en Buenos Aires.

6.2) Si no nos despedimos ahora, entonces no cumpliremos nuestro plan. No nos despedimos ahora.

6.3) Si son las cinco, entonces la oficina está cerrada. La oficina no está cerrada.

Problema Nº7: Simboliza cada uno de los siguientes razonamientos, determina la conclusión de modo que el razonamiento sea válido y justifica mediante un razonamiento válido elemental.

7.1) Juan vive en el norte de España o vive en el sur de Francia. Juan no vive en el norte de España.

7.2) Aumentan los ingresos familiares o aumenta el índice de pobreza. No aumentan los ingresos familiares.

7.3) Si Juan terminó de leer el libro, entonces el libro está en la biblioteca. El libro no está en la biblioteca.

Problema Nº8: Demuestra la validez de los siguientes razonamientos.

8.1)

p_1: $p\ \wedge q$

$\underline{p_2:\ p \rightarrow (q \rightarrow r)}$

$\therefore r$

8.3)

p_1: $p \vee\ q$

p_2: $q \rightarrow t$

$\underline{p_3: p \rightarrow r}$

$\therefore\ r \vee t$

8.5)

p_1: $p \rightarrow (r \wedge t)$

p_2: $s\ \rightarrow p$

$\underline{p_3: r\ \rightarrow \neg t}$

$\therefore\ \neg s$

8.2)
$p_1: \neg p \rightarrow q$
$p_2: q \rightarrow r$
$\underline{p_3: \neg r}$
$\therefore p$

8.4)
$p_1: \neg q \rightarrow \neg p$
$p_2: \neg p \rightarrow t$
$\underline{p_3: \neg t}$
$\therefore q$

8.6)
$p_1: p \vee (s \rightarrow \neg t)$
$\underline{p_2: s \wedge t}$
$\therefore p$

Problema Nº9: Demuestra que los siguientes enunciados son válidos. Utiliza el método directo.

9.1) Las computadoras tienen memoria finita y las instrucciones de los programas son finitas, las ciencias de la computación son finitas por naturaleza. Sin embargo, si las ciencias de la computación son finitas, las matemáticas discretas no le son de gran ayuda. Sabemos que las matemáticas discretas si son de gran ayuda. Por consiguiente no es verdad que, las computadoras tienen memoria finita y las instrucciones de los programas son finitas.

9.2) Para los matemáticos si la lógica es la que tiene la palabra final sobre lo cierto y lo errado entonces debe estudiarse en la escuela de enseñanza media. La lógica no se estudia en las escuelas medias. La lógica es la ciencia que tiene la palabra final sobre lo errado; en consecuencia la lógica tampoco tiene la palabra final sobre lo cierto.

9.3) Si el segundo nativo dijo la verdad, entonces solamente uno de los nativos es político. El segundo nativo dijo la verdad. Luego, solamente uno de los nativos es político.

9.4) Si obtienes la beca te has de ir a Francia y si te vas a Francia entonces no has de doctorarte en Rosario. Si te nombran profesor en la Universidad has de doctorarte en Rosario. Has de obtener la beca o te nombran profesor en la Universidad. Por lo tanto, te has de ir a Francia o has de doctorarte en Rosario.

9.5) Si mi vecino es arquitecto entonces no es italiano. Mi vecino es arquitecto. Si mi hermano es abogado, no es francés. Mi hermano es abogado. Que el arquitecto no sea italiano y que mi hermano no sea francés implica que tendrán que pagar la deuda contraída. Que mi hermano sea abogado y que tengan que pagar la deuda contraída implica que tendrán que aceptar el negocio propuesto. De todas estas consideraciones se desprende que tendrán que aceptar el negocio propuesto.

9.6) Si 6 es par, entonces 2 no divide a 7. 5 no es primo o 2 divide a 7.
Pero 5 es primo, por lo tanto 6 es impar.

1.10 Lectura Complementaria

La LPC en las demostraciones matemáticas

Para distinguir un razonamiento válido hemos ofrecido una *prueba formal* o *deducción lógica*. Una importante aplicación de los conceptos de consecuencias y equivalencias lógicas y de las reglas de inferencia se encuentra cuando en Matemática necesitamos demostrar teoremas. Un teorema es básicamente una implicación del tipo $H \Rightarrow T$, donde H se denomina hipótesis (premisas) y T es la tesis (conclusión). En todo teorema $H \Rightarrow T$ se re-

quiere que el condicional sea tautológico. Mostraremos varios métodos que pueden ser usados para justificar que $H \Rightarrow T$ es una tautología.

1.10.1 Método directo

La regla del método directo fundamenta el método de demostración más empleado en Matemática y que consiste en demostrar la verdad de una conclusión o tesis, dadas unas premisas o hipótesis, que son verdaderas.

De hecho, se basa en que, para que una implicación de antecedente verdadero sea verdadera, debe probarse que el consecuente también es verdadero.

Este método consiste, entonces, en deducir la conclusión de sus premisas mediante una sucesión de razonamientos elementales, de cada uno de los cuales se sabe que es válido.

Cuando queremos demostrar la implicación $H \Rightarrow C$ partimos de la suposición de que H es verdadero y utilizando las reglas de inferencia, leyes de la lógica, axiomas, definiciones o teoremas, concluimos que C es verdadera.

Ejemplo 61

Para trabajar con este método, definiremos un sistema axiomático simple, para que luego, a partir de dichos axiomas podamos probar algunos teoremas.

Sea B un subconjunto de los números reales ($B \subseteq \mathbb{R}$) donde se cumplen los siguientes axiomas:

A_1: $3 \in B$

A_2: $x \in B \Rightarrow 2x+1 \in B$

A_3: $x, y \in B \Rightarrow x+y \in B$

Probemos ahora:

Teorema A: $7 \in B \Rightarrow 18 \in B$

Hipótesis: $7 \in B$

Tesis: $18 \in B$

Demostración:

1) $7 \in B$ por hipótesis

2) $2 \cdot 7 + 1 = 15 \in B$ de 1) y A_2

3) $15 \in B \wedge 3 \in B$ conjunción de 2) y A_1

4) $15 + 3 = 18 \in B$ de 3) y A_3

Luego $7 \in B \Rightarrow 18 \in B$

Veamos ahora, cómo a partir de los tres axiomas y del Teorema A podemos deducir otras proposiciones:

Teorema B: $2 \in B \Rightarrow 23 \in B$

Hipótesis: $2 \in B$

Tesis: $23 \in B$

Demostración:

1) $2 \in B$ por hipótesis

2) $2 \cdot 2 + 1 = 5 \in B$ de 1) y A_2
3) $3 \in B$ por A_1
4) $2 \cdot 3 + 1 = 7 \in B$ de 3) y A_2
5) $18 \in B$ por 4) y Teorema 1
6) $5 \in B \wedge 18 \in B$ conjunción de 2) y 5)
7) $5 + 18 = 23 \in B$ de 6) y A_3
Luego $2 \in B \Rightarrow 23 \in B$

Ejemplo 62

Demostraremos que: Si m y n son enteros positivos, tales que m es un factor de n, y n es un factor de m, entonces m = n

Estamos nuevamente en presencia de una propiedad de la forma $p \rightarrow q$ donde p es: m y n son enteros positivos, tales que m es un factor de n y n es un factor de m y q es: m = n. Recurrimos a una prueba directa:

Demostración:

Sean m, n enteros positivos, tales que m es un factor de n y n es un factor de m.

Dado que m es un factor de n, se sigue que $m \leq n$.

Por ser n un factor de m, resulta $n \leq m$.

De donde m = n

1.10.2 Método por contraposición

Para probar la implicación $H \Rightarrow T$, probamos el contra recíproco $\neg T \Rightarrow \neg H$. Es decir, tomamos $\neg T$ como válida y debemos deducir $\neg H$.

Cuando pensamos la notación $p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_n \rightarrow c$, análogamente suponemos que la conclusión c es falsa.

Para que la implicación $p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_n \rightarrow c$ sea verdadera, como su consecuente es falso, deber ser también falso el antecedente. Para ello basta que con la suposición hecha, alguna de las premisas resulte falsa.

Este método consiste en:

1) Suponer que la conclusión es falsa.

2) Analizar los valores de verdad de las proposiciones que componen las premisas. En el análisis se debe trabajar bajo la suposición de que las premisas son verdaderas; hasta que resultan todas verdaderas o hasta que una de ellas (premisa) resulte forzosamente falsa.

Si resultan todas las premisas verdaderas el razonamiento no es válido mientras que si alguna premisa es falsa, el razonamiento es válido.

Ejemplo 63

Seguiremos trabajando con el sistema axiomático planteado en el Ejemplo 61.

Teorema C: $19 \notin B \Rightarrow 8 \notin B$

Hipótesis: $19 \notin B$

Tesis: $8 \notin B$

Demostración:

Suponemos, por Contradicción, que la tesis no se cumple:

1) $8 \in B$ por contradicción
2) $8 + 8 = 16 \in B$ de 1 y A_3
3) $16 \in B \wedge 3 \in B$ conjunción de 2 y A_1
4) $16 + 3 = 19 \in B$ de 3 y A_3. Esto contradice la hipótesis

Luego $19 \notin B \Rightarrow 8 \notin B$

Teorema D: $(2x - 4) \notin B \Rightarrow x \notin B \vee -8 \notin B$

Hipótesis: $(2x - 4) \notin B$

Tesis: $x \notin B \vee -8 \notin B$

Demostración:

Suponemos, por Contradicción, que la tesis no se cumple:

1) $x \in B \wedge -8 \in B$ por contradicción
2) $(2x + 1) \in B \wedge -8 \in B$ de 1 y A_2
3) $2x + 1 + (-8) = (2x - 7) \in B$ de 2 y A_3
4) $(2x - 7) \in B \wedge 3 \in B$ de 3 y A_1
5) $2x - 7 + 3 = (2x - 4) \in B$ de 4 y A_3. Esto contradice la hipótesis

Luego $(2x - 4) \notin B \Rightarrow x \notin B \vee -8 \notin B$

Ejemplo 64

Veamos cómo se aplica este método de contraposición en la prueba formal de un razonamiento:

a) Sea el razonamiento

p_1: $q \rightarrow r$

$\underline{p_2}$: $\underline{\neg q}$

c: r

Suponemos que r es falso y comenzamos el análisis por p_1. Como el consecuente, r, es falso, el antecedente, q, debe ser falso, así p_1 es verdadera. A continuación analizamos p_2; como q es falso, $\neg q$ es verdadera. Por tanto el razonamiento no es válido ya que la conclusión es falsa y todas las premisas son verdaderas.

b) Consideremos el siguiente razonamiento:

p_1: $\neg q \rightarrow r$

$\underline{p_2}$: $\underline{\neg r}$

c: q

Suponemos que la conclusión es falsa. O sea q es falsa.

En p_1, $\neg q$ es verdadera, entonces r es verdadera, y por tanto, p_2 ($\neg$ r) es falsa. Lo cual indica que este razonamiento es válido puesto que encontramos una premisa falsa.

c) Probaremos que si n^2 es par, entonces n es par.

La proposición es de la forma $p \rightarrow q$ donde p es: n^2 es par, y q es: n es par. Utilizando la equivalencia anterior, probaremos que "Si n no es par entonces n^2 no es par", es decir, usaremos el condicional equivalente $\neg q \rightarrow \neg p$

Si n no es par entonces n es impar. Es decir:

$n = 2m + 1$, para algún entero m

$n^2 = (2m + 1)^2$

$n^2 = 4m^2 + 4m + 1$

$n^2 = 2(2m^2 + 2m) + 1$

De donde n^2 es impar, es decir no es par. Lo que completa la prueba.

1.10.3 Método por Reducción al Absurdo

En símbolos, para probar que $p \rightarrow q$ probamos que $(p \wedge \neg q) \rightarrow F_0$ (La equivalencia lógica de las proposiciones $p \rightarrow q$ y $(p \wedge \neg q) \rightarrow F_0$ puede demostrarse realizando la tabla de verdad correspondiente).

Si queremos demostrar, por ejemplo, que si un triángulo T es equilátero entonces es isósceles, podemos considerar:

p: el triángulo T es equilátero

q: el triángulo T es isósceles

Luego, en lugar de probar que $p \rightarrow q$, probaremos que $(p \wedge \neg q) \rightarrow F_0$. Así, suponemos que la conclusión q es falsa, esto es, que T no es isósceles, y por lo tanto no tiene al menos dos lados iguales. Pero por hipótesis, T es equilátero y por lo tanto tiene tres lados iguales, y entonces tiene también al menos dos lados iguales; con lo cual hemos llegado a una contradicción: T no tiene al menos dos lados iguales y T tiene al menos dos lados iguales (tiene tres). Concluimos luego que T es equilátero entonces es isósceles.

En este ejemplo pueden percibirse los pasos característicos de este método de demostración:

1) Suponer que lo que queremos demostrar es falso.

2) Utilizar las hipótesis como premisas adicionales para producir una contradicción de la forma $(r \wedge \neg r)$, para alguna proposición r.

3) Concluir que lo que se quería demostrar es verdadero.

Ejemplo 65

Analizaremos la validez del siguiente razonamiento, utilizando el Método de Reducción al Absurdo:

p_1: $p \vee q$

p_2: $\neg p \vee r$

p_3: $\neg r$

$\therefore q$

p_4: $\neg q$ de suponer que la conclusión es falsa

p_5: p de p_1 y p_4 al aplicar Silogismo Disyuntivo

p_6: r de p_2 y p_5 aplicando Silogismo Disyuntivo
p_7: F_0 de p_3 y p_6 Conjunción
c: q

Es decir, cuando queremos demostrar la implicación $H \Rightarrow T$ utilizamos la proposición $H \wedge \neg T \rightarrow F_0$ para llegar a una contradicción:

Ejemplo 66

Probaremos la equivalencia: Sea a un número entero. a^2 es divisible por 3 si y solo si a es divisible por 3.

Tomando las siguientes proposiciones:

p: a^2 es divisible por 3

q: a es divisible por 3

La propiedad se expresa en la forma $p \Leftrightarrow q$, que es lógicamente equivalente a demostrar $p \Rightarrow q \ \wedge \ q \Rightarrow p$.

La prueba $q \Rightarrow p$ que se hace por el método directo, resulta sencilla: Si a es divisible por 3, entonces existe algún entero k tal que $a = 3\,k$, luego $a^2 = 9\,k^2 = 3\,(3\,k^2)$, por lo que a^2 es divisible por 3.

La prueba $p \Rightarrow q$ se hace por el método de reducción al absurdo usando la equivalencia $p \wedge \neg\, q \Rightarrow F_0$. Suponemos que a^2 es divisible por 3 y que a no es divisible por 3, entonces, al dividirlo entre 3, el resto es 1 o es 2, luego $a = 3k_1 + 1$ o $a = 3k_2 + 2$, donde k_1, k_2 enteros. Luego: $a^2 = (3k_1 + 1)^2 = 9k_1^2 + 6k_1 + 1 = 3\,(3k_1^2 + 2k_1) + 1$ o bien,

$a^2 = 9k_2^2 + 12\,k_2 + 4 = 3\,(3k_2^2 + 4k_2 + 1) + 1$.

En cualquiera de los casos el resto no es cero, por lo que a^2 no es divisible por 3. Hemos obtenido una contradicción $F_0 \equiv p \wedge \neg\, p$. Luego si a^2 es divisible por 3 entonces a es divisible por 3.

Ejemplo 67

Para finalizar, presentamos una demostración donde el uso de la verdad de la disyunción es muy importante en la prueba sobre la existencia de números irracionales a y b tales que a^b es racional.

En efecto, sea $b = \sqrt{2}$, que es irracional; el número $\left(\sqrt{2}\right)^{\sqrt{2}}$ puede ser racional o irracional.

Si $\left(\sqrt{2}\right)^{\sqrt{2}}$ es racional ya está probado tomando $a = \sqrt{2}$ y $b = \sqrt{2}$.

Si $\left(\sqrt{2}\right)^{\sqrt{2}}$ es irracional entonces $a = \left(\sqrt{2}\right)^{\sqrt{2}}$ y $b = \sqrt{2}$ y se verifica que

$a^b = \left(\left(\sqrt{2}\right)^{\sqrt{2}}\right)^{\sqrt{2}} = \left(\sqrt{2}\right)^{\sqrt{2}\sqrt{2}} = \left(\sqrt{2}\right)^{\left(\sqrt{2}\right)^2} = \left(\sqrt{2}\right)^2 = 2$ que es racional.

Actividad 15:

Problema Nº1: Demuestra la validez de los siguientes enunciados, por distintos métodos.

1.1)
p_1: $\neg p \rightarrow \neg q$
p_2: q
$\therefore p$

1.3)
p_1: $p \rightarrow q$
p_2: p
$\therefore q$

1.5)
p_1: $p \rightarrow q$
p_2: $r \rightarrow s$
p_3: $p \vee r$
$\therefore q \vee s$

1.2)
p_1: $(r \wedge t) \rightarrow q$
p_2: $q \rightarrow s$
p_3: $\neg s$
$\therefore \neg r \vee \neg t$

1.4)
p_1: $p \rightarrow r$
p_2: $\neg c \rightarrow \neg r$
$\therefore p \rightarrow c$

1.6)
p_1: $p \rightarrow \neg q$
p_2: $\neg p \rightarrow t$
p_3: q
$\therefore t$

Problema Nº2: Demuestra la validez o no de los razonamientos.

2.1) O el agua está fría o el día no es caluroso. El día es caluroso. Si el estanque se acaba de llenar, entonces el agua está fría. Por tanto el estanque se acaba de llenar.

2.2) Si la sustracción no es posible siempre en el sistema de números entonces el sistema incluye otros números. Pero, la sustracción es siempre posible. Por tanto, el sistema no incluye otros números.

2.3) O la lógica es difícil o no les gusta a muchos estudiantes. Si la matemática es fácil, entonces la lógica no es difícil. Por tanto, si a muchos estudiantes les gusta la lógica, la matemática no es fácil.

2.4) Si Juan tomó el tren especial, entonces estuvo en el accidente, y si estuvo en el accidente, entonces no asistió a la reunión. Juan asistió a la reunión. Luego, Juan no tomó el tren.

Problema Nº3: Demuestra que:

3.1) Sean a, b reales no negativos y sea $a^2 \geq b^2$. Entonces $a \geq b$. ¿Es cierto el teorema recíproco?

3.2) El cuadrado de un entero impar es impar.

1.11 Problemas complementarios

Para finalizar esta unidad de aprendizaje, en la que hemos tratado de resaltar la interdependencia entre la lógica y las ciencias de la computación, proponemos nuevos desafíos al quehacer matemático de nuestros lectores, dándoles la oportunidad para resolver estos problemas complementarios.

Problema Nº1: Los ítems 1.1) a 1.3) están referidos a: $(((r \wedge (p \rightarrow q)) \wedge (r \rightarrow \neg q)) \rightarrow \neg p)$

1.1) Escribe la proposición quitando los paréntesis de acuerdo a las reglas de precedencia.

1.2) Escribe la proposición usando sólo los conectivos negación, conjunción y disyunción.

1.3) Simplifica la proposición dada. Justifica.

1.4) Escribe la negación de la proposición: $((p \vee \neg q) \wedge (F_0 \wedge \neg q))$

1.5) Escribe el valor de verdad de cada una de las siguientes proposiciones, justificando la respuesta:

a) Si $3 + 10 = 12$ entonces $3 - 10 = 6$.

b) Si $3 + 10 = 12$ entonces $3 + 2 = 5$.

c) $3 + 10 = 12$ si y sólo si $3 - 10 = 6$.

Problema Nº2: A continuación presentamos definiciones de conceptos matemáticos y sus traducciones al lenguaje simbólico.

a) Sea A un subconjunto de los números reales, $\mathbf{R}$. Un número real *i, es cota inferior del conjunto* A si *i* es menor o igual que todos los elementos de A.

Por ejemplo, si A es el intervalo real, $A = [-2, 1)$. A tiene infinitas cotas inferiores. Algunas cotas inferiores son: -25, -13, $-7/2$, -2.

La traducción de la definición al lenguaje simbólico es:

Sea $A \subseteq \mathbf{R}$, $i \in \mathbf{R}$ *es cota inferior del conjunto* $A \Leftrightarrow \forall x\ (x \in A \rightarrow (i \leq x))$.

b) La traducción al lenguaje simbólico de que el número natural *z es el mínimo* del conjunto {x, y} es:

$z \in \mathbf{N}$, $z =$ mín {x, y} si y sólo si $(z = x \wedge z \leq y) \vee (z = y \wedge z \leq x)$

2.1) Traduce al lenguaje simbólico:

Sea A un subconjunto de los números reales, $\mathbf{R}$. Un número real s*, es cota superior del conjunto* A si s es mayor o igual que todos los elementos de A.

2.2) Caracteriza simbólicamente al elemento *w* como *máximo* del conjunto {x, y}.

Problema Nº3: Dibuja y simplifica la red de conmutación está dada por:

$(p \wedge q) \vee (\neg p \wedge q) \vee (q \wedge r) \vee ((\neg r \vee p) \wedge (p \vee q))$.

Problema Nº4: La lógica trivalente (o Lógica de Lukasiewicz).

Consideremos una lógica con tres valores de verdad, designados por 0 (Falso), 0.5 (Indiferente), 1 (Verdadero), con las operaciones $\neg$, $\vee$, $\wedge$,$\rightarrow$, $\leftrightarrow$, definidas por las tablas de verdad que se dan a continuación.

Una tautología, como en la lógica bivalente, es una proposición siempre verdadera. Muestra que en la *lógica trivalente no son tautologías*:

$\neg(\neg p) \leftrightarrow p$; $p \vee \neg p$; $(p \rightarrow q) \leftrightarrow (\neg p \vee q)$; $\neg\ (p \wedge q) \leftrightarrow (\neg p \vee \neg q)$;

pero que en cambio resulta *una tautología:* $p \vee \neg p \vee \neg\ (\neg p)$

p	¬p
0	1
0.5	0
1	0.5

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
0	0	0	0	1	1
0	0.5	0	0.5	1	0.5
0	1	0	1	1	0
0.5	0	0	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5
0.5	1	0.5	1	0.5	0.5
1	0	0	1	0	0
1	0.5	0.5	1	0.5	0.5
1	1	1	1	1	1

La negación definida anteriormente se conoce con el nombre de la negación "*cíclica*". Determina qué ocurre con las afirmaciones anteriores cuando se sustituye la negación cíclica por la negación "diametral" dada por la tabla:

p	¬p	p	¬p	p	¬p
0	1	0,5	0,5	1	0

Problema Nº5: Hay dos restaurantes, uno junto al otro. Uno tiene un letrero que dice: "La buena comida no es barata" y el otro tiene un letrero que dice "La comida barata no es buena". ¿Dicen lo mismo?

Problema Nº6: Determina la cantidad de veces que se ejecuta *Write(X)* en el siguiente programa, si se sabe que P(X): $X > 8$ y que Q(I, J): $(I > 5)$ and $(J < 10)$

```
BEGIN
   X := 10;
   FOR I := 1 TO 5 DO
         FOR J := 1 TO I + 2 DO
               IF ( P(X) or Q(I,J) ) THEN
                     Write(X)
               ENDIF
         ENDFOR
         X := X – 1;
   ENDFOR
END
```

Problema Nº7: Utilizando las expresiones relacionales que se dan a continuación se pide que se exhiban segmentos de programas de computación que resuelvan los problemas dados.

Sean las siguientes *expresiones relacionales*:

P(X): " $X \geq 4$" Q(X) : " $X \leq 15$" R(X): " $X \bmod 2 = 0$"

S(X): " X := X + 4" T(X) : " Imprimir X " U(X): " X := X + 1".

Dados los siguientes *problemas*:

7.1) Leer un número. Si el número ingresado está comprendido entre cuatro y quince, ambos incluidos, sumarle 4.

7.2) Leer un número. Si el número ingresado está comprendido entre cuatro y quince, ambos incluidos, sumarle 4. Sino, escribirlo.

7.3) Escribir un número previamente ingresado que esté comprendido entre cuatro y quince, ambos incluidos, y que sea múltiplo de dos.

7.4) Escribir los 15 primeros números naturales.

7.5) Generar los 16 primeros números naturales y escribir los múltiplos de dos.

Sugerencias:

Ejemplo 7.4)

```
X = 1;
WHILE  Q( X ) DO
   T ( X )
   U ( X );
ENDWHILE
```

Problema Nº8: Diseña un algoritmo que resuelva cada tarea que se detalla en cada ítems.

8.1) Dados un dígito y un número natural, debe informar si el dígito aparece en el número natural.

8.2) Dados un dígito y un número natural, debe informar cuantas veces aparece el dígito en el número natural.

8.3) Dado un número natural, debe determinar si es capicúa.

8.4) Dado un número entero, debe determinar la cantidad de cifras impares que lo componen.

Problema Nº9: Analiza el siguiente algoritmo, teniendo en cuenta que se utiliza una lista L para ingresar los N dígitos del número entero que deseamos tratar, y determina la tarea del algoritmo.

```
Algoritmo Determina_Mi_Tarea
BEGIN
   Write(´Ingresa la cantidad de dígitos que tiene el número a procesar´)
   Read(N)
   S:= 0; C:= 0;
   FOR I:= 1 TO   N DO
         Read(L(I))
         IF  ( L(I)  mod 2 = 0) THEN
               S:= S + L(I);
               C:= C + 1;
         ENDIF
   ENDFOR
   Write ('El número que ingresaste es: ')
```

```
   FOR I:= 1 TO   N DO
         Write(L(I))
   ENDFOR
   Write ('Este algoritmo te informa que obtuvo dos valores, siendo ellos: ')
   Write (S, C)
END
```

Problema Nº10: Justifica la validez del razonamiento "Si llueve, llevo paraguas. Si llevo paraguas, no uso sombrero. Uso sombrero. Luego no llueve."

Problema Nº11: Justifica la validez de las siguientes reglas de inferencia:

11.1)	11.2)
p_1: $p \vee q$	p_1: $p \vee q$
p_2: $p \rightarrow r$	p_2: $p \rightarrow r$
p_3: $q \rightarrow r$	p_3: $\neg s \rightarrow \neg q$
c: r	c: $s \vee r$

Problema Nº12: ¿Es $\neg p$ una consecuencia lógica de $S = \{p \vee q ; \neg p \vee \neg q\}$?

Problema Nº13: Se dan pares de afirmaciones. Justifica si tales afirmaciones son equivalentes:

13.1) Afirmación A: "Los autos se detienen si el semáforo tiene la señal en rojo"
Afirmación B: "Los autos se detienen sólo si el semáforo tiene la señal en rojo"

13.2) En el lenguaje comercial se utilizan frases como:
"Si el consumidor paga con tarjeta de crédito entonces el precio aumenta en un 2%".
"El consumidor paga con tarjeta de crédito sólo si el precio aumenta en un 2%".
¿Tienen los mismos valores de verdad?

Problema Nº14: ¿Contradices o demuestras la proposición?
La expresión $p(n) = n^2 - n + 41$ es número primo para todo n entero no negativo.

Problema Nº15: Critica la siguiente demostración en $\mathbf{Z}$.
TEOREMA: $\forall\, a \in \mathbf{Z}$, $a = 0$.

Demostración:

Se verifica que $a^2 = a^2$. Luego $a^2 - a^2 = a^2 - a^2$.

Es decir: $(a - a)(a + a) = a(a - a)$ y cancelando el factor $(a - a)$ a ambos lados de la igualdad, obtenemos $a + a = a$. Sumando a ambos lados de la igualdad, el opuesto de a, $(-a)$, obtenemos $a = 0$.

1.12 Ejercicios de opción múltiple

Ejercicio 1: La fórmula $(((x \wedge y) \vee (\neg x \wedge y)) \vee (x \wedge \neg y))$ es equivalente a:

a) $(x \wedge y)$
b) $\neg x$
c) $(x \vee y)$
d) Ninguna de las anteriores.

Ejercicio 2: Si la valuación de $(p \rightarrow q)$ es 0, entonces las valuaciones de las siguientes fórmulas $((\neg p \vee \neg q) \rightarrow q)$ y $(\neg q \rightarrow \neg p)$ son respectivamente:

a) 0, 1
b) 0, 0
c) 1, 1
d) Ninguna de las anteriores.

Ejercicio 3: Considerando las expresiones relacionales: P(x): "x es estudiante mayor de 25 años" y Q(x): "x es egresado con más de 30 años", la expresión simbólica de la negación de: "Algunos estudiantes son mayores de 25 años y todos los egresados tienen más de 30 años", es:

a) $\exists x\, P(x) \wedge \forall x\, Q(x)$
b) $\forall x \neg P(x) \wedge \exists x \neg Q(x)$
c) $\forall x \neg P(x) \vee \exists x \neg Q(x)$
d) Ninguna de las anteriores.

Ejercicio 4: Considerando que P(x): x es par; Q(x): x es mayor que 3 y siendo x un entero positivo, la expresión verdadera es:

a) $\forall x\, (P(x) \wedge Q(x))$.
b) $\exists x\, (\neg P(x) \vee \neg Q(x))$
c) $(\forall x\, P(x)) \wedge (\exists x \neg Q(x))$
d) Ninguna de las anteriores.

Ejercicio 5: Una fórmula de la forma $(\theta \rightarrow \varphi)$, donde "$\theta$" y "$\varphi$" son fórmulas arbitrarias, es una tautología cuando:

a) Es verdadera para alguna interpretación.
b) El antecedente es verdadero y el consecuente es falso.
c) Su negación es insatisfacible.
d) Ninguna de las anteriores.

Ejercicio 6: La negación de la expresión simbólica de la proposición "Existen números enteros cuyo cubo aumentado en uno es igual al cubo del siguiente" es:

a) $\forall n\, ((n \in \mathbb{Z}) \rightarrow (n^3 + 1 \neq (n + 1)^3))$
b) $\exists n\, ((n \in \mathbb{Z}) \wedge (n^3 + 1 \neq (n + 1)^3))$
c) $\forall n\, ((n \in \mathbb{Z}) \rightarrow (n^3 + 1 = (n + 1)^3))$
d) Ninguna de las anteriores.

Ejercicio 7: La fórmula $\forall x\, \exists y\, P(x, y)$ es lógicamente equivalente a:

a) $\exists x\, \forall y\, P(x, y)$
b) $\exists x\, \exists y\, P(x, y)$
c) $\forall x\, \forall y\, P(x, y)$
d) Ninguna de las anteriores.

Ejercicio 8: θ implica lógicamente a φ cuando:

a) existe alguna interpretación que haciendo verdadera a θ hace verdadera a φ.

b) toda interpretación que hace verdadera a θ, hace verdadera a φ.

c) θ es consecuencia lógica de φ.

d) Ninguna de las anteriores.

Ejercicio 9: La expresión simbólica de la proposición "Los números naturales múltiplos de 6 son múltiplos de 2 es:

a) $\forall x ((x \in \mathbf{N} \wedge (x = 6m,\ m \in N)) \rightarrow (x = 2n,\ n \in \mathbf{N}))$.

b) $\exists x\ (x \in \mathbf{N} \wedge ((x = 6m,\ m \in N) \rightarrow (x = 2n,\ n \in \mathbf{N})))$.

c) $\forall x\ (x \in \mathbf{N} \wedge ((x = 2m,\ m \in N) \rightarrow (x = 6n,\ n \in \mathbf{N})))$.

d) Ninguna de las anteriores.

Ejercicio 10: La fórmula $((p \wedge q) \vee ((\neg p \wedge \neg q) \vee q))$ es equivalente a:

a) $(p \wedge \neg q)$

b) $(\neg p \vee q)$

c) p

d) Ninguna de las anteriores.

Ejercicio 11: Una interpretación es un modelo para un conjunto S de fórmulas cuando:

a) satisface a alguna fórmula de S.

b) satisface a la conjunción de fórmulas de S.

c) satisface a alguna fórmula de S o a su negación.

d) Ninguna de las anteriores.

Ejercicio 12: Una inferencia se usa para:

a) verificar que las fórmulas están bien formadas.

b) expresar fórmulas complejas mediante implicaciones.

c) demostrar que una conjunción de fórmulas implican lógicamente una conclusión.

d) Ninguna de las anteriores.