LÓGICAS DE BLOQUEO

Una Clase de Lógicas Paraconsistentes, con Interpretaciones en Computación Teórica y Grupos de Discusión

Tesis
Entregada a la
Facultad de Ciencias
de la Universidad de Chile
en Cumplimiento Parcial de los Requisitos
Para Optar al Grado de

Doctor en Ciencias con Mención en Matemáticas

por

Juan Sebastián Castillo Sandoval

Mayo, 2007

Director de Tesis : Dr. Renato Lewin R.

Codirector de Tesis : Dr. Marcelo Arenas S.

FACULTAD DE CIENCIAS
UNIVERSIDAD DE CHILE

INFORME DE APROBACIÓN
TESIS DE DOCTORADO

Se informa a la Escuela de Postgrado de la Facultad de Ciencias que la Tesis de Doctorado presentada por el candidato

JUAN SEBASTIÁN CASTILLO SANDOVAL

ha sido aprobada por la Comisión de Evaluación de la Tesis como requisito para optar al grado de Doctor en Ciencias con mención en Matemáticas, en el examen de Defensa de Tesis rendido el día 14 de noviembre de 2007.

Director de Tesis

Dr. Renato Lewin R. _____

Codirector de Tesis

Dr. Marcelo Arenas S. _____

Comisión de Evaluación de la Tesis

Dra. Irene Mikenberg L. _____

Dra. Gloria Schwarze D. _____

Dr. Eduardo Friedman R. _____

LÓGICAS DE BLOQUEO

Una Clase de Lógicas Paraconsistentes, con Interpretaciones en Computación Teórica y Grupos de Discusión

Agradecimientos

Es imposible mencionar en sólo una página a todas las personas del mundo universitario a las que debo agradecimiento. A través de las nombradas, envío mi gratitud también a las que no podré mencionar individualmente, a pesar de estar en deuda, académica o humanamente:

Agradezco muy especialmente a mi director y a mi codirector de tesis, Renato Lewin y Marcelo Arenas. El profesor Lewin, quien me dio a conocer la Lógica Paraconsistente (dentro de la cual se inscribe este trabajo), aceptó dirigir esta tesis sin tener ninguna muestra concreta sobre mi capacidad para sacarla a flote, y dedicó mucho tiempo y esfuerzo a discutir ideas, revisar el trabajo, y darme invaluable orientación y ayuda. Y el profesor Arenas, sin siquiera conocerme previamente, aceptó explicarme los tópicos computacionales necesarios, labor en la que muchas veces invirtió amablemente su tiempo, con gran disposición. Junto a ellos, agradezco al profesor Rolando Pomareda, cuyo amable auspicio me permitió, siendo alumno de la Universidad de Chile, desarrollar esta tesis bajo la dirección y codirección de profesores de la Universidad Católica. Y, naturalmente, agradezco a MECESUP, sin cuya beca seguramente me hubiese sido imposible cursar el Doctorado y realizar la tesis.

Agradezco a todos los profesores de la Universidad de Chile de quienes fui alumno en la Licenciatura en Matemáticas y en el Doctorado. Pensé mencionarlos a todos, pero por razones de espacio ello es evidente y lamentablemente imposible. Los simbolizaré en la persona del profesor Nicolás Yus, en quien mis numerosas inquietudes respecto a la filosofía y la naturaleza de las matemáticas encontraron eco amplio y entusiasta. Agradezco también a las profesoras Irene Mikenberg, Gloria Schwarze y Victoria Marshall, de la Universidad Católica, cuyas clases, junto a las del profesor Lewin, me mostraron el universo de la Teoría de Conjuntos y la Lógica Matemática.

Agradezco a mi amigo y ex condiscípulo de toda la Licenciatura y parte del Doctorado, Manuel Arenas, porque en las numerosas ocasiones en que la cabal comprensión de algunos temas se me volvía peligrosamente difícil, siempre conté con el auxilio de su gran talento matemático, tan superior al que escasamente poseo. Junto a él, agradezco a mi otro ex compañero del Doctorado en la Universidad de Chile, Leonelo Iturriaga, cuya ayuda (junto a la de Manuel) en la preparación de los temibles Exámenes de Calificación, fue invaluable. Finalmente, también a Gabriela, Pilar, Tonino y Carla, de la Universidad Católica, que casi sin conocerme me recibieron en su grupo de estudio con tanta cordialidad y buena disposición.

Agradezco a mis actuales compañeros de posgrado en Matemáticas de la Universidad de Chile por su grata camaradería, que otorga calidez a nuestro poco numeroso mundo de matemáticos. Quiero agradecer en especial a Leslie Jiménez y a Romina Menares: a Romina, por lo grato que ha sido siempre compartir con ella; y a Leslie, porque en momentos particularmente difíciles su amistad fue un apoyo y una alegría.

Agradezco a todos mis alumnos y ex alumnos de ayudantías, a quienes debo la enriquecedora experiencia de haber procurado aportarles un poco en el difícil inicio de la vida matemática, y con muchos de los cuales tengo el agrado de seguir compartiendo habitualmente. Hay tantos y tantas que quisiera mencionar aquí.... pero no es posible.

Finalizo con cuatro personas, no pertenecientes al mundo académico, a las que me es indispensable agradecer: a mis padres, Estinelly y Miguel, a cuyo amor infinito debo más de lo que puede expresarse en palabras; a mi hermano Juan Cristián: no importa la edad, siempre será mi hermanito menor; y a Ximena, que por tanto tiempo me ha entregado mucho más de lo que merezco, con una autenticidad, una pureza y una nobleza sin límites.

A todos, y a los muchos que no pude mencionar, mi más sincero agradecimiento.

RESUMEN

En el presente trabajo se define una familia de sistemas deductivos proposicionales, de los cuales como mínimo una gran parte son paraconsistentes. Primero se define y estudia detenidamente uno de dichos sistemas paraconsistentes, y se muestra detalladamente cómo permite describir aspectos del comportamiento de bases de datos inconsistentes, tratadas con el método llamado "de reparaciones". A continuación, en base a las principales características de la lógica con la que se ha estado trabajando, se define la familia de sistemas deductivos mencionada. Se muestra también una interpretación para otra de estas lógicas, interpretación de carácter bastante diferente a la propuesta para el sistema anteriormente estudiado.

ABSTRACT

In this work a family of propositional deductive systems is defined, from which an important number of them, as a minimum, are paraconsistent. To start with, one of the above mentioned paraconsistent systems is defined and thoroughly studied. The way that system allows to describe aspects of inconsistent databases behaviour is shown, worked out with the method called "repairs". Next, based on the main characteristics of the logic we have been working with, the afore mentioned family of deductive systems is defined. An interpretation for another one from these logics is also shown. This interpretation shows a rather different character from the proposed one for the system studied before.

Índice

Introducción

La lógica matemática nació a mediados del siglo XIX, como un intento de utilizar los métodos y la formalidad de la matemática para estudiar los procedimientos "correctos" para razonar, que la lógica tradicional estudia de manera no matemática. En la primera mitad del siglo XX surgieron una serie de variantes, que se apartaban de los modos tradicionales de raciocinio, dirigiéndose en ocasiones en direcciones probablemente insospechables antes de la aplicación a la lógica del modo de pensar propio de la matemática. Las partes de la lógica matemática que se ocupan de los principios aceptados por la lógica tradicional son conocidas como "lógica (matemática) clásica", mientras que las que se ocupan de otro tipo de principios son las "lógicas no clásicas". Dentro de éstas, se denominan "paraconsistentes" aquellas lógicas no clásicas que "toleran ciertas contradicciones" (en el apéndice del segundo capítulo de este trabajo se explicará el sentido exacto de esta expresión). La lógica tradicional considera como un principio básico del razonamiento el que las contradicciones son inadmisibles, y por lo tanto la existencia de lógicas paraconsistentes, articuladas formalmente como sistemas deductivos estudiados con las herramientas rigurosas de la matemática, tiene interés filosófico para la lógica tradicional. Es así que se han planteado diversas opiniones al respecto, habiendo incluso estudiosos que sostienen que no puede siquiera existir una cosa tal como una lógica paraconsistente (en el apéndice del segundo capítulo apuntaremos los motivos que esgrimen).

En el presente trabajo, presentamos una clase de sistemas deductivos, muchos de cuyos integrantes son paraconsistentes; dada la mencionada variedad de opiniones sobre la paraconsistencia, en el apéndice del segundo capítulo puntualizaremos en qué sentido los calificamos de tales. Mostraremos también interpretaciones para dos integrantes de la clase, estudiando muy detalladamente una de ellas; al respecto, es importante apuntar lo siguiente: para la mayoría de los matemáticos, las contrapartes "filosóficas" de su trabajo no tienen mayor importancia, pues son ajenas a la labor matemática; pero quienes trabajan en el área de la lógica matemática son, en general, una excepción: dado el objeto de su estudio, suelen dar importancia a las interpretaciones intuitivas de los sistemas deductivos formales que estudian. Ciertamente esto no es indispensable, pues, como matemática que es, la lógica matemática define con precisión objetos y operaciones, cuyo funcionamiento no requiere de ninguna alusión filosófica o coloquial (de lo contrario, no se la podría incluir dentro de la matemática); pero, en la práctica, quienes trabajan en el área tienen en general interés en

los aspectos filosóficos de los sistemas formales que estudian, para que, por así decirlo, "merezcan llamarse lógicas". Esto es especialmente cierto en el caso de las lógicas paraconsistentes, debido a la mencionada trangresión de un principio tradicionalmente considerado básico; revisando los trabajos que se han publicado, observamos que muchos de los principales autores del área han declarado explícitamente que al articular sus sistemas deductivos han tenido como objetivo prioritario el que tengan una justificación intuitiva. Por consiguiente, al definir un sistema deductivo paraconsistente es muy valioso mostrar interpretaciones "naturales" para él. Consecuentemente, conferiremos importancia a este aspecto, dedicando un extenso capítulo a plantear una interpretación para el sistema deductivo paraconsistente que habremos definido, y a mostrar detalladamente en qué sentido formal este sistema y esta interpretación se corresponden.

El presente trabajo está estructurado de la siguiente manera:

En el primer capítulo se presentan algunas convenciones, referentes sobre todo a notación, y se recuerdan al lector algunos conceptos usuales de la lógica matemática, que utilizaremos en nuestro trabajo. Se establece también un criterio para distinguir los resultados originales de aquellos preexistentes, indicando que todo resultado numerado es original del presente trabajo.

En el segundo capítulo se define un sistema deductivo proposicional, y se desarrollan diversos resultados referentes a él, mostrándose, entre otras cosas, que es paraconsistente. Se obtienen resultados que se necesitarán para el trabajo posterior, y también algunos (pocos) que no se requerirán, pero que contribuyen a la comprensión del funcionamiento de esta lógica.

En el tercer capítulo se presenta una interpretación para el sistema deductivo definido en el capítulo anterior, interpretación que se inscribe en el área de la computación teórica (consiste en describir algunos aspectos del comportamiento de bases de datos inconsistentes, mediante la lógica que se ha definido). Para ello, primero se presentan al lector las nociones de computación necesarias para comprender el capítulo; a continuación se plantea el objetivo a lograr, que, como se ha mencionado, consiste en formalizar la idea de que el caso estudiado constituye una interpretación "legítima" para el sistema deductivo; esta formalización, que es desarrollada a continuación, consiste en mostrar que el sistema deductivo es correcto y completo, con respecto a la interpretación propuesta y en las condiciones especificadas; para terminar el capítulo, se esboza brevemente cómo podría operar nuestro sistema deductivo en otras situaciones del estilo de la ya estudiada.

En el cuarto capítulo, teniendo ya a nuestra disposición una idea del funcionamiento del sistema deductivo con el que se ha trabajado, se define en base a ciertas características de él una clase de sistemas deductivos, a cuyos integrantes se da el nombre de "lógicas de bloqueo"; se efectúan algunos comentarios sobre ellos, haciendo notar que la definición de la clase no implica que necesariamente todos sus miembros sean paraconsistentes.

En el quinto capítulo se define una nueva lógica proposicional de bloqueo paraconsistente, y se muestra brevemente una interpretación para ella, obteniendo de este modo una segunda interpretación "natural" para un sistema deductivo de bloqueo paraconsistente.

En el sexto y último capítulo se esboza someramente cómo se podría extender a lógica de primer orden (lógica predicativa) el sistema deductivo proposicional con que se trabajó en los capítulos dos y tres.

Finalmente, se presenta la bibliografía pertinente, donde figuran tanto los trabajos que han sido utilizados para el desarrollo del presente, como algunos otros que son útiles como referencia general respecto a los temas atinentes al presente trabajo.

Para concluir esta introducción, quizás sea buena idea sugerir al lector empezar por revisar la primera parte del apéndice del Capítulo 2, pues ahí se explica qué se entiende exactamente por "paraconsistencia", y se esboza brevemente un contexto para comprender el concepto.

CAPÍTULO 1

PRELIMINARES

Aunque el lector, por el hecho mismo de estar abordando el estudio del presente trabajo, ha de tener conocimientos previos sobre lógica matemática, será útil citar algunas definiciones elementales de este campo, principalmente porque a veces estas definiciones de conceptos básicos varían un poco de un autor a otro, de modo que es importante fijarlas desde el principio. Asimismo, será conveniente adoptar desde ya algunas convenciones con respecto a la notación, así como recordar otras que son de uso común, pero que a veces también varían un poco según el autor. Tenemos lo siguiente:

Un **lenguaje proposicional** consta de símbolos agrupados en tres categorías: letras proposicionales, conectivos lógicos y signos de puntuación. En el cálculo proposicional clásico los signos de puntuación son los paréntesis, las letras proposicionales son un conjunto infinito numerable $\{p_1, p_2, p_3, \dots\}$, y se utilizan habitualmente cuatro o cinco conectivos lógicos, de los cuales, en el presente trabajo utilizaremos solamente los cuatro conectivos "estándar", que son:

$\neg$ (negación)
$\vee$ (disyunción)
$\wedge$ (conjunción)
$\rightarrow$ (implicación)

A estos cuatro conectivos clásicos, agregaremos un conectivo unario que no pertenece al lenguaje del cálculo proposicional clásico.

Una vez que fijemos un lenguaje proposicional, definiremos el conjunto de las **fórmulas bien formadas** al estilo habitual. Aunque las fórmulas bien formadas están rigurosamente definidas, nos tomaremos ciertas libertades de escritura, habituales en trabajos de lógica matemática, sobre todo con los paréntesis, con objeto de hacer más cómoda la lectura; así, aunque hay un único tipo de paréntesis (los redondos), las fórmulas que tengan muchos paréntesis las escribiremos alternando paréntesis redondos y cuadrados, para que se distingan mejor; asimismo, muchas veces se omitirán algunos paréntesis en fórmulas que en rigor los llevan,

pero que sin ellos se leen sin ambigüedad, bajo las siguientes extendidas convenciones: los paréntesis exteriores frecuentemente se omiten; la negación "se aplica a tan poco como sea posible"; la conjunción y la disyunción también, salvo presencia de negaciones, ante la cual la regla anterior tiene preferencia; por ejemplo, escribiremos $P_2 \wedge P_3$ en vez de $(P_2 \wedge P_3)$, y escribiremos $\neg P_1 \wedge P_3 \rightarrow P_4$ en vez de $(((\neg P_1) \wedge P_3) \rightarrow P_4)$.

Un **sistema deductivo** en un lenguaje proposicional, es un conjunto de fórmulas destacadas, los **axiomas**, junto con un conjunto de reglas, las **reglas de inferencia**, en base a los y las cuales se establece una noción rigurosa de "deducción" o "demostración". En el caso del cálculo clásico, puede utilizarse cualquiera de entre varios conjuntos, diferentes pero equivalentes, de axiomas (más precisamente, de esquemas axiomáticos); la regla de inferencia de uso más habitual es *Modus Ponendo Ponens* (o simplemente *Modus Ponens*), que indica que si en una demostración se han obtenido las fórmulas A y $A \rightarrow B$, donde A y B son fórmulas cualesquiera, entonces puede agregarse a la demostración la fórmula B. En base a ello, en el cálculo clásico se define: dado un conjunto Σ de fórmulas (las **premisas**), una **deducción formal** (o **demostración formal**) de una fórmula ϕ a partir de Σ, es una sucesión finita de fórmulas $\left(\phi_1, \phi_2, \ldots, \phi_n\right)$, tal que $\phi_n = \phi$, y tal que para cada una de ellas se cumple que, o bien es un axioma, o bien es una premisa (está en Σ), o bien se ha obtenido aplicando *Modus Ponens* a fórmulas que hayan aparecido anteriormente en la deducción. En el presente trabajo introduciremos una definición de demostración formal levemente diferente, pero que para el cálculo proposicional clásico es equivalente a la mencionada.

Una **asignación veritativa** es una función σ que a cada letra proposicional le asigna un valor perteneciente a un conjunto con dos elementos $\{V\ ,\ F\}$, llamados así por "verdadero" y "falso". Una tal función σ se extiende de manera unívoca al conjunto de todas las fórmulas bien formadas del cálculo proposicional clásico, mediante la siguiente definición recursiva:

$$
\begin{array}{lcl}
\sigma(\neg\phi_1) = V & \Leftrightarrow & \sigma(\phi_1) = F \\
\sigma(\phi_1 \wedge \phi_2) = V & \Leftrightarrow & \sigma(\phi_1) = V \ \text{ y } \ \sigma(\phi_2) = V \\
\sigma(\phi_1 \vee \phi_2) = F & \Leftrightarrow & \sigma(\phi_1) = F \ \text{ y } \ \sigma(\phi_2) = F \\
\sigma(\phi_1 \rightarrow \phi_2) = F & \Leftrightarrow & \sigma(\phi_1) = V \ \text{ y } \ \sigma(\phi_2) = F
\end{array}
$$

A cada asignación veritativa le corresponde, con la recursión recién descrita, exactamente una extensión; debido a esto, frecuentemente se habla de asignación

veritativa al aludir a lo que en rigor es la extensión de la asignación. Siguiendo esta práctica, al utilizar este concepto, cuando demos un nombre a una asignación veritativa (por ejemplo σ), se usará tal nombre para referirse indistintamente tanto a la asignación propiamente tal como a su extensión.

Es muy usual utilizar letras griegas, como ϕ , ψ , para nombrar fórmulas (en el metalenguaje). También se utilizan a veces, aunque es menos frecuente, letras latinas, como A, B, C. Cuando se usan dentro de un texto redactado en lenguaje coloquial, las letras griegas son más cómodas a la vista (se distinguen fácilmente del resto del texto, al menos en castellano y demás idiomas que usan alfabeto latino); pero en fórmulas en que figuran varias subfórmulas, como por ejemplo en algunos axiomas, me parece más legible usar letras latinas (nuevamente, al menos en los idiomas en que usamos este alfabeto). Por lo tanto adoptaré indistintamente ambos usos: las fórmulas del lenguaje objeto serán nombradas a veces como ϕ , ψ , ξ , etc., y a veces como A, B, C, D, etc.

El conjunto de las fórmulas bien formadas de un lenguaje proposicional dado será denotado con la abreviatura $\mathcal{FOR}$. Si ϕ es una fórmula y $x_1, x_2, \ldots, x_k$ son letras proposicionales, la expresión $\phi(x_1, x_2, \ldots, x_k)$ significa que toda letra proposicional que aparece en ϕ está en el conjunto $\{x_1, x_2, \ldots, x_k\}$, aunque no necesariamente todas las letras $x_1, x_2, \ldots, x_k$ figuran en ϕ.

Para referirnos al Cálculo Proposicional Clásico utilizaremos la habitual abreviatura CPC.

Para expresar "ϕ se deduce de Σ" o "ϕ se demuestra a partir de Σ" (siendo Σ un conjunto de fórmulas) escribiremos, como es usual, $\Sigma \vdash \phi$. A veces, cuando en un trabajo de lógica matemática se usa una definición de deducción formal distinta de la clásica, no se denota con el mismo símbolo; pero como la que presentaremos es, según se dijo, equivalente a la clásica para CPC, usaremos el símbolo habitual. Cuando queramos decir específicamente que ϕ se deduce de Σ en CPC, **siempre** escribiremos $\Sigma \vdash_{cpc} \phi$.

Muchas veces se establece una diferencia entre el concepto de "sistema deductivo" (que es "algo puramente sintáctico", según la definición que hemos citado antes) y el concepto de "lógica" (que sería "algo con contenido semántico"). En el presente trabajo, aunque a veces usaremos indistintamente las dos expresiones, nos estaremos refiriendo siempre a sistemas deductivos, salvo expresa indicación contraria.

Con respecto a la numeración de los resultados: es costumbre habitual, en muchos escritos matemáticos, llevar numeraciones por separado para los distintos tipos de resultados; por ejemplo, el capítulo 2 de un determinado trabajo podría empezar con la definición [2.1], después continuar con la definición [2.2], y después con el teorema [2.1]; personalmente, me parece que es mejor el modo de numerar en el que se va siguiendo una numeración correlativa, independiente del tipo de resultado, ya que en la práctica, cuando se aborda el estudio de un texto matemático, uno debe estar revisando constantemente resultados que han aparecido previamente, y este proceso resulta mucho más ágil, en mi opinión, usando este segundo modo de numerar. Este será, entonces, el criterio que adoptaré; en el ejemplo recién descrito, la numeración sería así:

[2.1] **DEFINICION**: Sea A un .....

[2.2] **DEFINICION**: Sea B un .....

[2.3] **TEOREMA**: *Sea C un* .....

Al final de cada ítem numerado, se indicará el término del mismo mediante el símbolo □□ .

Para facilitar la distinción entre los resultados obtenidos en este trabajo y aquellos preexistentes, utilizaré el criterio siguiente: toda definición, lema, teorema, corolario u observación preexistente no será numerado, de modo que toda definición, lema, teorema, corolario u observación que esté numerado, es original de este trabajo (hasta donde es de mi conocimiento); sólo dejaré sin numerar aquellas observaciones no muy relevantes, que parezca más adecuado incluir en el cuerpo mismo del texto, a modo de simple comentario. Aunque (según estamos mencionando) no se numerarán las definiciones de uso común, ni las definiciones y teoremas pertenecientes a otros autores, ellas serán destacadas con letras negritas.

[NOTA: Son excepciones al criterio que acabamos de indicar, al menos en cierta medida, las definiciones [2.1] y [2.2] , pues el lenguaje proposicional que establece [2.1] ha sido utilizado previamente, y en [2.2] se definen las fórmulas bien formadas al estilo habitual. Sin embargo, estas definiciones han sido numeradas, por un lado para que el desarrollo de nuestro trabajo comience de manera sistemática y ordenada, y por otro lado para darle al lenguaje proposicional que utilizaremos un nombre adecuadamente relacionado con las lógicas que serán presentadas. Mencionemos también que las definiciones [3.2] , [3.4] y [3.16] se

basan directamente en conceptos preexistentes, pero la formulación exacta que aquí presentamos es inédita.]

Finalmente, indiquemos que al término de varios de los capítulos se incluye, como sección final, un apéndice de comentarios "filosóficos" o históricos, relacionados a veces con las interpretaciones intuitivas de las lógicas que estamos presentando, y otras veces con aspectos históricos o filosóficos de la lógica paraconsistente en general. Según se mencionó en la introducción del presente trabajo, si estas contrapartes intuitivas de los sistemas deductivos formales son de interés para la mayoría de quienes trabajan en lógica matemática, esto es especialmente cierto en el caso de las lógicas paraconsistentes.

CAPÍTULO 2

UNA PRIMERA LÓGICA DE BLOQUEO

En este capítulo definiremos un sistema deductivo proposicional paraconsistente, y estudiaremos algunos resultados referentes a él. En el capítulo siguiente lo aplicaremos a bases de datos inconsistentes. Una vez hecho esto, teniendo ya una cierta idea de cómo funciona esta lógica, nos basaremos en ella para definir (en el capítulo 4) la clase de los sistemas deductivos de bloqueo, clase de la cual el que ahora presentamos será un ejemplo en particular.

Para definir este sistema deductivo utilizaremos los símbolos del lenguaje del cálculo clásico CPC, más un símbolo adicional, el conectivo unario ^{o} , que llamaremos **"símbolo de confirmación"** (y que escribiremos como superíndice; por ejemplo, la expresión $((P_1 \wedge P_2)^o)$ será una fórmula bien formada). Por lo tanto, el lenguaje proposicional que usaremos queda conformado del modo siguiente :

[2.1] **DEFINICION:** Denominaremos **lenguaje proposicional de bloqueo (básico)** al lenguaje proposicional que consta de los siguientes símbolos:

Letras proposicionales (infinitas numerables): P_1 , P_2 , P_3 , P_4 , P_5 , ...
Conectivos: $\neg$, $\vee$, $\wedge$, $\rightarrow$, ^{o}
Paréntesis (signos de puntuación): (,)

□□

[NOTA: El símbolo que hemos llamado "de confirmación", el superíndice ^{o} , fue creado por Newton da Costa, pero con un uso diferente al del presente trabajo, como abreviatura o símbolo definido (no como símbolo primitivo); sin embargo, el sentido "filosófico" que este autor le daba tiene cierta relación con el que puede tener en la lógica que vamos a definir; es por eso que lo he adoptado. Al respecto, véase el apéndice histórico-filosófico de este primer capítulo.]

La definición de las fórmulas bien formadas será la usual, agregando el caso del símbolo nuevo:

[2.2] DEFINICION: El conjunto de las **fórmulas bien formadas** en el lenguaje proposicional de bloqueo (básico) se define recursivamente, del modo siguiente:

a) Las letras proposicionales P_i son fórmulas bien formadas.
b) Si ϕ y ψ son fórmulas bien formadas, también lo son las expresiones $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\neg\psi)$ y (ψ^o).
c) Las únicas fórmulas bien formadas son aquellas expresiones (sucesiones de símbolos) que se obtienen mediante el uso de (a) y/o (b).

□□

Según se indicó en los preliminares, nos tomaremos las libertades habituales referentes a los paréntesis. Respecto al símbolo de confirmación, lo trataremos igual que al otro conectivo unario (la negación), es decir "aplicándolo a tan poco como sea posible". Por ejemplo, la fórmula $\phi \wedge \phi^o$ debe ser interpretada como $(\phi \wedge (\phi^o))$, y no como $((\phi \wedge \phi)^o)$; y la fórmula $\neg\phi \wedge \psi^o$ es $((\neg\phi) \wedge (\psi^o))$.

Ahora definiremos nuestro primer sistema deductivo de bloqueo. El sentido del nombre que a continuación le daremos a este sistema, será explicado cuando definamos la clase de las lógicas de bloqueo. La definición de deducción formal es algo diferente de la clásica, por lo que la incluiremos aquí, en la definición del sistema deductivo. Junto a esta definición (de deducción formal), esta lógica constará de cinco reglas de inferencia y diez axiomas. Los nombres que daremos a las reglas de inferencia sugieren su papel en el sistema deductivo. En cuanto a los axiomas, obsérvese que ellos "son los mismos de CPC, pero en conjunción con su confirmación", en el sentido de que se obtienen tomando una de las axiomatizaciones conocidas del cálculo proposicional clásico, y escribiendo cada uno de tales axiomas ϕ en conjunción con la fórmula ϕ^o :

[2.3] DEFINICION: Denominaremos **BLO-OC** al sistema deductivo, del lenguaje proposicional de bloqueo, dado por la siguiente definición de deducción formal, las siguientes reglas de inferencia y los siguientes axiomas:

DEDUCCION FORMAL: Dado un conjunto Σ de fórmulas (las **premisas**), una **deducción formal** (o **demostración formal**) **de una fórmula** ϕ **a partir** de Σ es una sucesión finita de fórmulas $\left(\phi_1, \phi_2, \ldots, \phi_n\right)$, tal que $\phi_n = \phi$, y tal

que para cada ϕ_i de la demostración se cumple que, o bien ella es un axioma, o bien ella ha sido obtenida aplicando una de las reglas de inferencia a fórmulas pertenecientes a $\Sigma \cup \{\phi_1, \phi_2, ..., \phi_{i-1}\}$ (es decir, a fórmulas que hayan aparecido en la demostración con anterioridad a ϕ_i , y/o a premisas).

REGLAS DE INFERENCIA: Si A y B son fórmulas cualesquiera, las siguientes son reglas de inferencia:

[1] $\{A, A^o\} \vdash A \wedge A^o$ (Adjunción de Confirmación)

[2] $A \wedge A^o \vdash A$ (Desadjunción de Confirmación)

[3] $A \wedge A^o \vdash A^o$ (Desadjunción de Confirmación)

[4] $\{A, A^o, A \to B, (A \to B)^o\} \vdash B \wedge B^o$ (Pseudo - Modus Ponens)

[5] $\{A, \neg\neg A\} \vdash A^o$ (Obtención de Confirmación)

AXIOMAS: Si A , B y C son fórmulas cualesquiera, las siguientes fórmulas son axiomas:

[A1] $[A \to (B \to A)] \wedge [A \to (B \to A)]^o$

[A2] $[(A \to B) \to ((A \to [B \to C]) \to (A \to C))] \wedge [(A \to B) \to ((A \to [B \to C]) \to (A \to C))]^o$

[A3] $[A \to (B \to [A \wedge B])] \wedge [A \to (B \to [A \wedge B])]^o$

[A4] $[(A \wedge B) \to A] \wedge [(A \wedge B) \to A]^o$

[A5] $[(A \wedge B) \to B] \wedge [(A \wedge B) \to B]^o$

[A6] $[A \to (A \vee B)] \wedge [A \to (A \vee B)]^o$

[A7] $[B \to (A \vee B)] \wedge [B \to (A \vee B)]^o$

[A8] $[(A \to C) \to ((B \to C) \to ([A \vee B] \to C))] \wedge [(A \to C) \to ((B \to C) \to ([A \vee B] \to C))]^o$

[A9] $[(A \to B) \to ([A \to \neg B] \to \neg A)] \wedge [(A \to B) \to ([A \to \neg B] \to \neg A)]^o$

[A10] $[\neg\neg A \to A] \wedge [\neg\neg A \to A]^o$

□□

[2.4] EJEMPLOS: A partir del conjunto de premisas $\Sigma = \{P_1, \neg\neg P_1\}$, en CPC una deducción de la fórmula P_1 tendría longitud 1: sería simplemente (P_1). En **BLO-OC** no podemos incluir directamente la premisa; para demostrar P_1, podemos efectuar la deducción siguiente (bajo ella se justifica cada paso):

$$\left(\quad P_1{}^o \quad , \quad P_1 \wedge P_1{}^o \quad , \quad P_1 \quad \right)$$

(Regla [5] sobre premisas , regla [1] sobre paso 1 y premisa , regla [2] sobre paso 2)

En cambio, según demostraremos posteriormente, P_1 no puede ser deducida a partir de $\{P_1, \neg P_1\}$ en **BLO-OC**, a pesar de que $P_1 \in \{P_1, \neg P_1\}$. Obsérvese que en principio las premisas "tienen bloqueado el paso hacia las demostraciones" por la definición de deducción formal; en el ejemplo que acabamos de ver, la premisa P_1 "fue admitida" en la demostración sólo después de "ser desbloqueada" por la acción de las reglas de inferencia [1], [2] y [5]. Esta situación (que, nótese, da indicios del sentido del nombre "lógicas de bloqueo") resultará ser característica del sistema deductivo **BLO-OC**. Veamos ahora una demostración (en **BLO-OC**) sin premisas. Demostraremos la tautología clásica $A \rightarrow (A \wedge B \rightarrow B)$; listaremos los pasos verticalmente, y después indicaremos la justificación de cada uno:

(1) $\Big([A \wedge B \rightarrow B] \rightarrow [A \rightarrow (A \wedge B \rightarrow B)]\Big) \wedge \Big([A \wedge B \rightarrow B] \rightarrow [A \rightarrow (A \wedge B \rightarrow B)]\Big)^o$

(2) $[A \wedge B \rightarrow B] \rightarrow [A \rightarrow (A \wedge B \rightarrow B)]$

(3) $\Big([A \wedge B \rightarrow B] \rightarrow [A \rightarrow (A \wedge B \rightarrow B)]\Big)^o$

(4) $[A \wedge B \rightarrow B] \wedge [A \wedge B \rightarrow B]^o$

(5) $[A \wedge B \rightarrow B]$

(6) $[A \wedge B \rightarrow B]^o$

(7) $[A \rightarrow (A \wedge B \rightarrow B)] \wedge [A \rightarrow (A \wedge B \rightarrow B)]^o$

(8) $[A \rightarrow (A \wedge B \rightarrow B)]$

Justificación de los pasos: axioma [A1]; regla de inferencia [2]; regla de inferencia [3]; axioma [A5]; regla de inferencia [2]; regla de inferencia [3], *Pseudo - Modus Ponens* (regla de inferencia [4]) sobre pasos (5), (6), (2) y (3); y regla de inferencia [2].

Si utilizamos para CPC la axiomatización en la que nos hemos basado para definir los axiomas de **BLO-OC**, tenemos, para la fórmula $A \to (A \wedge B \to B)$, la siguiente deducción en CPC "correspondiente" a la recién escrita:

$$\Big(\quad [A \wedge B \to B] \to [A \to (A \wedge B \to B)] \quad , \quad [A \wedge B \to B] \quad , \quad [A \to (A \wedge B \to B)] \quad \Big)$$

(axioma, axioma, *Modus Ponens*). Si el lector examina estas dos demostraciones, la comparación probablemente le sugerirá un modo de demostrar en **BLO-OC** las otras tesis del cálculo proposicional clásico (las fórmulas que en CPC pueden demostrarse sin premisas). Esto resultará ser correcto: más adelante probaremos que en **BLO-OC** se puede demostrar (sin premisas) toda tautología clásica [ver comentarios en el apéndice histórico-filosófico].

□□

[2.5] OBSERVACION: La definición de demostración formal que hemos establecido para el sistema deductivo **BLO-OC** en [2.3], es en principio diferente a la de CPC, pues en éste se puede incluir cualquier premisa directamente en la demostración, mientras que aquí eso no se permite (sólo los axiomas pueden ser incluidos directamente), y además en el caso clásico las reglas de inferencia se deben aplicar sólo a fórmulas que hayan aparecido con anterioridad en la demostración, es decir a fórmulas en $\{\phi_1, ..., \phi_{i-1}\}$ (para el paso i de la demostración), y no en $\Sigma \cup \{\phi_1, ..., \phi_{i-1}\}$. Sin embargo, si aplicamos al cálculo clásico CPC la presente definición, su funcionamiento deductivo queda intacto: en efecto, supongamos que en CPC suprimimos la posibilidad de incluir directamente premisas en las deducciones, pero permitimos aplicar *Modus Ponens* a $\Sigma \cup \{\phi_1, ..., \phi_{i-1}\}$ en el paso i de la demostración (todo esto, sin alterar su regla de inferencia *Modus Ponens*, ni sus axiomas); entonces, dada cualquier premisa $A \in \Sigma$, podemos incluir en la deducción una demostración de la tautología $A \to A$ (que, como toda tautología, con esta nueva definición aún puede ser demostrada usando sólo axiomas, sin necesidad de premisas) y podemos aplicar *Modus Ponens* a esta fórmula y a la premisa A, con lo cual podemos ahora agregar a la demostración la fórmula A, de modo que "no se pierde nada suprimiendo la inclusión directa de premisas" (si se ha agregado la posibilidad de aplicar la regla de inferencia a $\Sigma \cup \{\phi_1, ..., \phi_{i-1}\}$). Asimismo, esta adición de la posiblidad de aplicar *Modus Ponens* a premisas que no hayan aparecido en la demostración no agrega nada, pues con la definición habitual de todos modos se podía llevar la premisa a la demostración, y ahí aplicarle *Modus Ponens*. En

resumen, para el cálculo proposicional clásico la presente definición de deducción formal es equivalente a la usual, de manera que podría ser usada para CPC en lugar de dicha definición habitual, quedando intacto este sistema deductivo [ver comentario en el apéndice histórico-filosófico del presente capítulo].

□□

Obsérvese, también, que nuestra definición de demostración formal conlleva una dificultad extra (en comparación con CPC) cuando se quiere probar alguna propiedad de la lógica **BLO-OC** mediante inducción sobre la longitud de las demostraciones de fórmulas, ya que al utilizar este procedimiento, la hipótesis inductiva será válida sólo para fórmulas que hayan aparecido en la deducción, pero al aplicar las reglas de inferencia podría estarse usando una premisa que no está en la deducción, y que por lo tanto no necesariamente cumple con la hipótesis inductiva. Dado que este procedimiento (inducción sobre la longitud de las demostraciones) se utilizará con bastante frecuencia (cosa que ocurre usualmente en las partes sintácticas de la lógica matemática), vale la pena considerar esta observación.

[2.6] **NOTACION:** En este capítulo y en el siguiente, $\Sigma \vdash \phi$ **siempre** significará que ϕ se deduce a partir de Σ **en BLO-OC**. Recordemos también que, para decir que ϕ se deduce de Σ en CPC, escribiremos $\Sigma \vdash_{cpc} \phi$.

□□

La mayoría de los resultados que obtendremos, concernientes al sistema deductivo **BLO-OC**, se refieren a conjuntos de premisas escritas en lenguaje clásico, es decir fórmulas que no contienen el símbolo de confirmación. Por lo tanto, convendremos la notación siguiente:

[2.7] **NOTACION:** En lo que resta del presente capítulo, y también en el siguiente (en el capítulo 3), el símbolo " Σ " representará **siempre** a un conjunto (arbitrario pero fijo) de fórmulas que no contienen el símbolo de confirmación ^{o}, es decir, un conjunto de fórmulas escritas en el lenguaje de CPC.

□□

El enunciado del primer teorema que demostraremos requiere las dos definiciones siguientes. La segunda viene precedida de una explicación referente a su sentido intuitivo.

[2.8] DEFINICION: Para cada fórmula ψ del lenguaje de **BLO-OC**, la **parte principal de** ψ, denotada $\mathbf{P}(\psi)$, se define del modo siguiente:

$$\mathbf{P}(\psi) = \begin{cases} \phi & \text{si } \psi = \phi^o \text{ o si } \psi = \phi \wedge \phi^o \\ \psi & \text{en los otros casos} \end{cases}$$

□□

Queremos ahora definir una función $\mathcal{FOR} \to \mathcal{FOR}$ que haga lo siguiente: dada una fórmula cualquiera del lenguaje de **BLO-OC**, nuestra función, bajo condiciones adecuadas, deberá buscar las subfórmulas de forma ϕ^o "maximales", es decir que no son a su vez subfórmulas de otra subfórmula de la misma forma, y sustituir cada una de ellas por una letra proposicional que no aparezca en las premisas, cambiando todas las apariciones de una subfórmula dada (de forma ϕ^o) por una misma letra proposicional, y usando letras distintas para subfórmulas ϕ^o distintas (y, además, sin alterar el resto de la fórmula). Así, por ejemplo, la fórmula

$$(P_1)^o \wedge (P_2 \vee [P_3 \vee (P_5)^o]^o)$$

tiene dos subfórmulas de forma ϕ^o "maximales", $(P_1)^o$ y $[P_3 \vee (P_5)^o]^o$, de manera que debería ser transformada por nuestra función en una fórmula como ésta:

$$P_8 \wedge (P_2 \vee P_{11});$$

y la fórmula siguiente:

$$\neg\Big(\big[P_1 \vee ([P_2]^o)\big]^o\Big) \vee \Big(P_3 \rightarrow \big[P_1 \vee ([P_2]^o)\big]^o\Big)$$

se debería convertir en algo como

$$(\neg P_7) \vee \big(P_3 \rightarrow P_7\big) \ .$$

Procedamos ahora a definir formalmente una función que efectúe un procedimiento como el descrito:

[2.9] DEFINICION: Llamemos $\mathcal{FOR}^o$ al conjunto de todas las fórmulas bien formadas de forma (ψ^o). Tal conjunto es infinito numerable, de modo que hay biyecciones entre él y el conjunto de los números naturales; por lo tanto, dado cualquier natural K, existen biyecciones entre $\mathcal{FOR}^o$ y los naturales mayores que K. Llamemos BK a una tal biyección (arbitraria pero fija) entre $\mathcal{FOR}^o$ y los naturales mayores que K. Dada esta BK, definimos recursivamente la función $\mathbf{C_{BK}} : \mathcal{FOR} \rightarrow \mathcal{FOR}$ del modo siguiente:

a) Para cada letra proposicional P_i , $\mathbf{C_{BK}}(P_i) = P_i$.

b) Para cualesquiera fórmulas ψ y ϕ , definimos $\mathbf{C_{BK}}(\neg\psi) = \neg(\mathbf{C_{BK}}(\psi))$, $\mathbf{C_{BK}}(\psi \wedge \phi) = \mathbf{C_{BK}}(\psi) \wedge \mathbf{C_{BK}}(\phi)$, $\mathbf{C_{BK}}(\psi \vee \phi) = \mathbf{C_{BK}}(\psi) \vee \mathbf{C_{BK}}(\phi)$, $\mathbf{C_{BK}}(\psi \rightarrow \phi) = \mathbf{C_{BK}}(\psi) \rightarrow \mathbf{C_{BK}}(\phi)$.

c) Para cada $\phi \in \mathcal{FOR}$, $\mathbf{C_{BK}}(\phi^o) = P_{BK(\phi^o)}$, es decir, a cada fórmula de forma (ϕ^o) la función $\mathbf{C_{BK}}$ le asigna la letra proposicional cuyo subíndice es el número natural que BK le asigna a esa misma fórmula (ϕ^o) .

□□

[NOTA: Dada la manera en la que vamos a utilizar esta función, en principio resultará perjudicial el hecho de que, hablando informalmente, ella puede "alterar la forma" de las fórmulas a las que se aplica de un modo que no se quiere, en el siguiente sentido: por ejemplo, si tomando K=1 se escoge una biyección B1 tal que $\mathrm{B1}([P_2]^o) = 4$, entonces se tiene que $\mathbf{C_{B1}}([P_2]^o) = P_4$, y con ello tenemos que $\mathbf{C_{B1}}(P_4 \rightarrow [P_2]^o) = \mathbf{C_{B1}}(P_4) \rightarrow \mathbf{C_{B1}}([P_2]^o) = P_4 \rightarrow P_4$: obtenemos una tautología clásica, a partir de una fórmula que no tiene esa forma tautológica; esto

resultará inadecuado para el uso que pretenderemos darle a esta función $\mathbf{C_{BK}}$ (en cambio, obtener $P_4 \rightarrow P_4$ a partir de por ejemplo $[P_2]^o \rightarrow [P_2]^o$ estaría bien, pues la fórmula original tiene la misma forma tautológica que la resultante) (repetimos que estos comentarios son sólo intuitivos, no formales). Esto lo corregiremos escogiendo, en el teorema [2.11], un natural K adecuado, relacionado con el conjunto de las letras proposicionales que aparecen en las premisas que se estén considerando; en realidad, esta necesidad de limitar el cambio en la forma de las fórmulas, en relación a las letras proposicionales que figuran en las premisas, es precisamente la razón por la que se incorporó, en la definición de la función $\mathbf{C_{BK}}$, el número natural K del que depende la biyección BK.]

[2.10] OBSERVACION: Nótese que por la propia definición de la función $\mathbf{C_{BK}}$, dada cualquier fórmula ψ del lenguaje proposicional de bloqueo, el símbolo de confirmación no aparece en la fórmula $\mathbf{C_{BK}}(\psi)$; por lo tanto, si ϕ es una fórmula perteneciente a la imagen de $\mathbf{C_{BK}}$, entonces ϕ está escrita en el lenguaje de CPC.

Asimismo, obsérvese que las partes (a) y (b) de la definición nos dicen que las fórmulas del cálculo proposicional clásico son invariantes bajo la acción de esta función, es decir que si ϕ es una fórmula escrita en el lenguaje de CPC (en ϕ no aparece ^{o}), entonces $\mathbf{C_{BK}}(\phi) = \phi$.

□□

Pasamos ahora al primer teorema referente a **BLO-OC**. Este nos dará una restricción para el conjunto de fórmulas que pueden ser deducidas formalmente en **BLO-OC** a partir de premisas que no contienen el símbolo ^{o}. Recuérdese que hemos convenido (en [2.6] y [2.7]) que el símbolo Σ denotará siempre a un conjunto de premisas que no contienen el símbolo de confirmación ^{o}, y que el símbolo $\vdash$ se refiere a deducción en **BLO-OC**.

[2.11] TEOREMA: *Sea K un número natural tal que* $\phi = \phi(p_1, p_2, \ldots, p_K)$ *para todo* $\phi \in \Sigma$. *Si* $\Sigma \vdash \psi$, *entonces* $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$. (**P** *y* $\mathbf{C_{BK}}$ *según definiciones* [2.8] *y* [2.9], $\mathbf{C_{BK}}$ *cualquiera de las posibles para el* K *dado*).

Demostración: Antes de empezar, nótese que la Observación [2.10], y el hecho de que las fórmulas de Σ están (según la notación convenida) escritas sin el símbolo ^{o}, garantizan que la expresión $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$ está bien definida [si en $\mathbf{C_{BK}}(\mathbf{P}(\psi))$ o en Σ apareciese el símbolo de confirmación, la expresión podría carecer de sentido].

Probaremos el teorema por inducción sobre la longitud de las demostraciones de fórmulas a partir del conjunto (arbitrario pero fijo) Σ :

Demostraciones de longitud 1: Si ψ es una fórmula que ha sido demostrada a partir de Σ en un sólo paso, entonces ψ es un axioma de **BLO-OC** o ha sido obtenida con la regla de inferencia [5] (*Obtención de Confirmación*), porque para aplicar cualquiera de las otras cuatro reglas de inferencia se requeriría que las premisas contuvieran el símbolo de confirmación ^{o}.

Si ψ se obtuvo con la regla de inferencia [5], entonces ψ es de forma ϕ^o siendo las fórmulas ϕ y $\neg\neg\phi$ premisas (pertenecen al conjunto Σ) ; lo primero nos dice que $\mathbf{P}(\psi) = \phi$; lo segundo nos dice que en ϕ no aparece el símbolo de confirmación (por pertenecer ϕ a Σ), de manera (por observación [2.10]) que $\mathbf{C_{BK}}(\phi) = \phi$; en resumen, tenemos que. $\mathbf{C_{BK}}(\mathbf{P}(\psi)) = \phi$ con ϕ y $\neg\neg\phi$ pertenecientes a Σ . Por lo tanto $\mathbf{C_{BK}}(\mathbf{P}(\psi)) \in \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$, luego $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$.

Y si ψ es axioma, veamos que $\mathbf{C_{BK}}(\mathbf{P}(\psi))$ es una tautología clásica: tenemos que cada axioma ψ de **BLO-OC** es de forma $\phi \wedge \phi^o$, donde ϕ , que es $\mathbf{P}(\psi)$, es una instancia de tautología clásica (aunque posiblemente conteniendo el símbolo de confirmación). Si ψ era el axioma [A1] , tenemos $\phi = A \to [B \to A]$, con lo que $\mathbf{C_{BK}}(\mathbf{P}(\psi)) = \mathbf{C_{BK}}(A \to [B \to A]) = \mathbf{C_{BK}}(A) \to [\mathbf{C_{BK}}(B) \to \mathbf{C_{BK}}(A)]$; pero en la observación [2.10] vimos que las fórmulas pertenecientes a la imagen de la función $\mathbf{C_{BK}}$ no contienen el símbolo de confirmación; por lo tanto, $\mathbf{C_{BK}}(A) \to [\mathbf{C_{BK}}(B) \to \mathbf{C_{BK}}(A)]$ es una fórmula de la forma $C \to [D \to C]$ escrita en el lenguaje de CPC, de manera que es una tautología clásica (de hecho, si tomamos para CPC la axiomatización en la que hemos basado **BLO-OC**, esta fórmula $C \to [D \to C]$ es un axioma de CPC). Analizando los demás axiomas de **BLO-OC**, vemos que ocurre algo similar (esto es debido a que en la parte

principal de cada axioma de **BLO-OC**, los conectivos principales son los clásicos, de modo que al aplicarle a cada axioma la función $\mathbf{C_{BK}}$, ésta "preserva la forma antes de eliminar el símbolo de confirmación"). Por ejemplo, para el axioma [A6] se tiene $\mathbf{C_{BK}}(\phi) = \mathbf{C_{BK}}(A \to [A \vee B]) = \mathbf{C_{BK}}A \to [\mathbf{C_{BK}}A \vee \mathbf{C_{BK}}B]$, que es fórmula en el lenguaje de CPC de forma $C \to [C \vee D]$, y por lo tanto tautología clásica (probablemente axioma, según cuál axiomatización se tome para CPC). En resumen, para cada axioma ψ de **BLO-OC** se tiene que $\mathbf{C_{BK}}(\mathbf{P}(\psi))$ es una tautología de CPC (probablemente incluso un axioma de CPC), y por lo tanto $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$.

Hipótesis inductiva: Suponemos que para toda fórmula ψ que puede ser demostrada en **BLO-OC** a partir de Σ en menos de N pasos, se cumple que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$.

Paso inductivo: Sea ψ fórmula demostrada a partir de Σ en N pasos. Son posibles los siguientes casos:

(a) Si ψ no es de forma ϕ^o ni de forma $\phi \wedge \phi^o$, entonces $\mathbf{P}(\psi) = \psi$, y la fórmula ψ se obtuvo necesariamente con la regla de inferencia [2] (ψ no es axioma, porque no es de forma $\phi \wedge \phi^o$, y las otras reglas de inferencia sólo producen fórmulas de forma ϕ^o o de forma $\phi \wedge \phi^o$); ψ fue entonces deducida a partir de la fórmula $\psi \wedge \psi^o$, que no es premisa ya que en ella aparece el símbolo de confirmación; por lo tanto, $\psi \wedge \psi^o$ figura en la demostración (antes de ψ), de modo que $\psi \wedge \psi^o$ se demuestra a partir de Σ en menos de N pasos, luego podemos aplicarle la hipótesis inductiva; tenemos entonces que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi \wedge \psi^o))$, y como $\mathbf{P}(\psi \wedge \psi^o) = \psi = \mathbf{P}(\psi)$, lo que hemos obtenido es $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$, que es lo que se quería demostrar.

(b) Si ψ es de forma $\phi \wedge \phi^o$, entonces ψ se obtuvo con la regla de inferencia [1], o con la [2], o con la [4], o es un axioma (pues las otras dos reglas de inferencia producen fórmulas cuyo conectivo principal es ^{o}, y no $\wedge$). El caso de ψ axioma fue tratado en la primera etapa de la inducción (al estudiar las demostraciones de longitud 1). Si ψ se obtuvo con la regla de inferencia [1], entonces en la deducción había aparecido la fórmula ϕ^o (pues no puede ser premisa); por lo tanto, por hipótesis inductiva $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\phi^o))$, y como $\mathbf{P}(\phi^o) = \phi = \mathbf{P}(\phi \wedge \phi^o) = \mathbf{P}(\psi)$, lo que hemos obtenido es que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$. Si ψ fue obtenida aplicando la regla de inferencia [2], entonces en la deducción había aparecido la fórmula $(\phi \wedge \phi^o) \wedge (\phi \wedge \phi^o)^o$ [no puede ser premisa], de manera que le aplicamos la hipótesis inductiva y

tenemos $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}[(\phi \wedge \phi^o) \wedge (\phi \wedge \phi^o)^o])$; como $\mathbf{P}[(\phi \wedge \phi^o) \wedge (\phi \wedge \phi^o)^o] = (\phi \wedge \phi^o)$, y como $\mathbf{C_{BK}}(\phi \wedge \phi^o) = [\mathbf{C_{BK}}(\phi)] \wedge P_i$ (para alguna letra proposicional P_i), tenemos $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} [\mathbf{C_{BK}}(\phi)] \wedge P_i$; y como $\mathbf{C_{BK}}(\phi) \wedge P_i \vdash_{cpc} \mathbf{C_{BK}}(\phi)$, concluimos que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\phi)$, es decir $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$. Finalmente, si ψ fue obtenida aplicando *Pseudo-Modus Ponens*, entonces en la deducción de ψ habían aparecido fórmulas $(A \to \psi)^o$ y A^o (no pueden ser premisas); por hipótesis inductiva $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(A^o)) = \mathbf{C_{BK}}(A)$ y $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}([A \to \psi]^o)) = \mathbf{C_{BK}}(A \to \psi) = \mathbf{C_{BK}}(A) \to \mathbf{C_{BK}}(\psi)$; por *Modus Ponens* en CPC, esto dice que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\psi) = \mathbf{C_{BK}}(\phi \wedge \phi^o)$, y en el caso recién analizado (el caso de $\psi = \phi \wedge \phi^o$ obtenida con la regla [2]) vimos que esto implica que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$.

(c) Si ψ es de forma ϕ^o, sólo pudo ser obtenida aplicando la regla [2], o la regla [3], o la regla [5]. Sin embargo, notamos que no es posible que haya sido obtenida aplicando la regla [2]: en efecto, si suponemos que sí, entonces se había deducido la fórmula $[\phi^o] \wedge [(\phi^o)^o]$ (no puede ser premisa), luego, por hipótesis inductiva, $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}([\phi^o] \wedge [(\phi^o)^o])) = \mathbf{C_{BK}}(\phi^o) = P_i$ siendo P_i una letra proposicional que no figura en Σ (recordar que K es un número natural tal que $\phi = \phi(p_1, p_2, \ldots, p_K)$ para todo $\phi \in \Sigma$, y que la imagen de BK consta sólo de números mayores que K); esto , según sabemos, es imposible en CPC, por lo tanto la suposición de que se había deducido $\phi^o \wedge (\phi^o)^o$ es falsa, y por lo tanto no se pudo obtener ψ usando la regla [2]. Si ψ se obtuvo con la regla [3], se había deducido la fórmula $\phi \wedge \phi^o$ [no es premisa]; por hipótesis inductiva, $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\phi \wedge \phi^o)) = \mathbf{C_{BK}}(\phi) = \mathbf{C_{BK}}(\mathbf{P}(\psi))$. Finalmente, si ψ fue obtenida mediante la regla [5], dicha regla se aplicó a las fórmulas ϕ y $\neg\neg\phi$. Hay ahora tres casos posibles: uno, las fórmulas ϕ y $\neg\neg\phi$ son ambas premisas, en cuyo caso $\phi \in \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$, y por lo tanto $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \phi = \mathbf{C_{BK}}(\phi) = \mathbf{C_{BK}}(\mathbf{P}(\psi))$ [notando que $\phi = \mathbf{C_{BK}}(\phi)$ porque ϕ es premisa]; dos, $\neg\neg\phi$ es premisa pero ϕ no lo es, en cuyo caso en ϕ no aparece el símbolo de confirmación (ya que $\neg\neg\phi$ es premisa), con lo que $\mathbf{P}(\phi) = \phi$, de modo que al aplicar la hipótesis inductiva a ϕ tenemos que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\phi)) = \mathbf{C_{BK}}(\phi) = \mathbf{C_{BK}}(\mathbf{P}(\psi))$; y tres, la fórmula $\neg\neg\phi$ no es premisa, en cuyo caso al aplicarle la hipótesis inductiva tenemos $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\neg\neg\phi)) = \mathbf{C_{BK}}(\neg\neg\phi) = \neg\neg\mathbf{C_{BK}}(\phi)$, y como $\neg\neg\mathbf{C_{BK}}(\phi) \vdash_{cpc} \mathbf{C_{BK}}(\phi)$, tenemos $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\phi)$, es decir $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$.

□□

[2.12] **COROLARIO :** *Si se utiliza un conjunto de premisas construido a partir de un número finito de letras proposicionales y sin usar el símbolo °, no es posible demostrar en* **BLO-OC** *fórmulas de forma* $(\phi^o)^o$, *ni de forma* $(\phi^o) \wedge (\phi^o)^o$.

Demostración: Sea K un natural tal que todas las letras proposicionales que figuran en el conjunto de premisas Σ pertenecen a $\{p_1, p_2, \ldots, p_K\}$, y fijemos una función $\mathbf{C_{BK}}$. Tenemos entonces que $\mathbf{P}([\phi^o]^o) = \phi^o = \mathbf{P}([\phi^o] \wedge [\phi^o]^o)$, y con ello $\mathbf{C_{BK}}(\mathbf{P}([\phi^o]^o)) = \mathbf{C_{BK}}(\mathbf{P}([\phi^o] \wedge [\phi^o]^o)) = \mathbf{C_{BK}}(\phi^o) = P_j$ con P_j letra proposicional que no figura en Σ . El teorema nos dice, por lo tanto, que para que ocurriese que $\Sigma \vdash (\phi^o)^o$ o que $\Sigma \vdash (\phi^o) \wedge (\phi^o)^o$, tendría que darse que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} P_j$ con P_j letra proposicional tal que ninguna fórmula de Σ la contiene, y sabemos que en CPC esto no puede ocurrir.

(Alternativamente, el lector podría inferir este corolario más o menos directamente del principio de la parte (c) del paso inductivo, en la demostración del Teorema [2.11])

□□

[2.13] **COROLARIO :** *Si se utiliza un conjunto de premisas construido sin usar el símbolo °, no es posible demostrar en* **BLO-OC** *fórmulas de forma* $(\phi^o)^o$ *ni de forma* $(\phi^o) \wedge (\phi^o)^o$.

Demostración: Supongamos que $\Sigma \vdash (\phi^o)^o$, probablemente figurando en Σ infinitas letras proposicionales (esto último sería posible sólo si Σ contiene infinitas fórmulas). Consideremos una demostración formal cualquiera (fija) de la fórmula $(\phi^o)^o$ a partir de Σ . La definición de deducción formal en **BLO-OC**, establecida en [2.3] , indica que en esta demostración de $(\phi^o)^o$ se han usado las reglas de inferencia una cantidad finita de veces. Cada una de las reglas de inferencia se aplica sobre un número finito de fórmulas (más específicamente, como máximo sobre cuatro), de manera que la cantidad de premisas que se han utilizado para efectuar la demostración es finita. En otras palabras, existe un suconjunto finito de Σ , llamémoslo Δ , tal que $\Delta \vdash (\phi^o)^o$. Como Δ es finito, contiene una cantidad finita de letras proposicionales, y como es subconjunto de Σ , en Δ figuran solamente fórmulas en las que no aparece el símbolo de confirmación: entonces, el haber concluido que $\Delta \vdash (\phi^o)^o$ contradice el lema anterior [2.12] . Por consiguiente, la suposición inicial de que $\Sigma \vdash (\phi^o)^o$ es falsa. Idéntico argumento nos dice que $\Sigma \nvdash (\phi^o) \wedge (\phi^o)^o$.

□□

Aparte de los que acabamos de ver, el Teorema [2.11] nos proporciona otros dos corolarios, [2.17] y [2.18] , que serán utilizados en nuestra aplicación de **BLO-OC** a bases de datos; pero para abordar estos corolarios (así como otros resultados que obtendremos) necesitaremos las siguientes definiciones:

[2.14] DEFINICION: Sea ϕ una fórmula cualquiera. Entonces $\overline{\phi}$ denotará a la fórmula ϕ sin el símbolo ^{o}, es decir a la fórmula resultante de eliminar de ϕ cada aparición del símbolo de confirmación ^{o}. Lo definimos recursivamente:

a) Para cada letra proposicional P_i definimos $\overline{P_i} = P_i$
b) Para A y B fórmulas arbitrarias, definimos $\overline{(A^o)} = \overline{A}$, $\overline{(\neg A)} = \neg(\overline{A})$, $\overline{(A \wedge B)} = \overline{A} \wedge \overline{B}$, $\overline{(A \vee B)} = \overline{A} \vee \overline{B}$, y $\overline{(A \rightarrow B)} = \overline{A} \rightarrow \overline{B}$.

□□

[2.15] DEFINICION: Definimos la función **S:** $\mathcal{FOR} \rightarrow \mathcal{FOR}$ de la siguiente manera:

$$\mathbf{S}(\psi) = \begin{cases} \overline{\phi} \wedge (\overline{\phi})^o & \text{si } \psi = \phi \wedge \phi^o \\ (\overline{\phi})^o & \text{si } \psi = \phi^o \\ \overline{\psi} & \text{en los otros casos} \end{cases}$$

A esta función la llamaremos **función simplificación**, y las fórmulas pertenecientes a la imagen de **S** serán denominadas **fórmulas simplificadas**. Además, usaremos el prefijo "S-" para indicar que nos referimos a fórmulas simplificadas; en particular, con frecuencia usaremos la siguiente nomenclatura:
"ϕ **es una S-fórmula**" significará que ϕ es una fórmula simplificada.
"ϕ **es S-demostrable**" o "**hay una S-demostración de** ϕ" significarán que existe una demostración de ϕ que consta solamente de fórmulas simplificadas.

□□

[2.16] **OBSERVACIONES:** Si una fórmula ψ no es de forma ϕ^o ni de forma $\phi \wedge \phi^o$, entonces la fórmula $\mathbf{S}(\psi)$ se construye suprimiendo el símbolo ^{o} de cada subfórmula de ψ ; por ejemplo, si $\psi = A \to B$ (con A y B fórmulas cualesquiera), entonces $\mathbf{S}(\psi) = \overline{\psi} = \overline{A} \to \overline{B}$ $[\neq \mathbf{S}(A) \to \mathbf{S}(B)]$.

Nótese también que la función $\mathbf{S}$ es idempotente, es decir que $\mathbf{S}(\mathbf{S}(\psi)) = \mathbf{S}(\psi)$.

Finalmente, obsérvese que en virtud de la forma de los axiomas de CPC y de **BLO-OC** (tomando para CPC la axiomatización "correspondiente" a **BLO-OC**), y del primer párrafo de esta observación, se tiene lo siguiente: si ψ es un axioma de **BLO-OC**, es de forma $\phi \wedge [\phi^o]$, siendo ϕ una instancia de esquema axiomático de **CPC** (aunque probablemente conteniendo el símbolo de confirmación); por lo tanto $\overline{\phi}$ [es decir la parte principal de la simplificación $\overline{\phi} \wedge (\overline{\phi})^o$ de $\phi \wedge (\phi)^o$], es un axioma de CPC, y por consiguiente la simplificación $\mathbf{S}(\psi) = \mathbf{S}(\phi \wedge [\phi^o]) = \overline{\phi} \wedge (\overline{\phi})^o$ es un axioma de **BLO-OC**. Por ejemplo, si ϕ es de forma $A \to (B \to A)$ (probablemente conteniendo el símbolo de confirmación ^{o}), entonces la fórmula $\overline{\phi} = \overline{A} \to (\overline{B} \to \overline{A})$ es un axioma de CPC, y por lo tanto la fórmula $\psi = \overline{\phi} \wedge (\overline{\phi})^o$ es un axioma de **BLO-OC**.

□□

[2.17] **COROLARIO:** *Si* Σ *es, según hemos convenido, un conjunto de premisas construidas sin utilizar el símbolo* ^{o}*, y si* ψ *es una* **S***-fórmula tal que* $\Sigma \vdash \psi$ *, entonces* $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{P}(\psi)$.

Demostración: Si $\Sigma \vdash \psi$, entonces, según lo explicado en la demostración de [2.13] en relación a la definición de deducción formal en **BLO-OC** , existe un suconjunto finito de Σ , llamémoslo Δ , tal que $\Delta \vdash \psi$; por ser subconjunto de Σ , en Δ figuran solamente fórmulas en las que no aparece el símbolo de confirmación, y como Δ es finito, contiene una cantidad finita de letras proposicionales, de modo que existe un número natural K tal que $\phi = \phi(p_1, p_2, \ldots, p_K)$ para todo $\phi \in \Delta$. Podemos entonces aplicarle el Teorema [2.11] a Δ : escogiendo para el número K cualquier función $\mathbf{C_{BK}}$, tenemos que $\{\xi \in \Delta : \neg\neg\xi \in \Delta\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$. Del hecho de que $\Delta \subseteq \Sigma$, se deduce la inclusión $\{\xi \in \Delta : \neg\neg\xi \in \Delta\} \subseteq \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$; y como en CPC se verifica

la propiedad de monotonía, se sigue que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$. Ahora bien, si ψ es una S-fórmula, en su parte principal $\mathbf{P}(\psi)$ no aparece el símbolo de confirmación, de manera que, según la observación [2.10], tenemos $\mathbf{C_{BK}}(\mathbf{P}(\psi)) = \mathbf{P}(\psi)$. Luego la afirmación $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{C_{BK}}(\mathbf{P}(\psi))$ es equivalente, para S-fórmulas, a la afirmación $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \mathbf{P}(\psi)$.

□□

[2.18] COROLARIO: *Si* $\psi = \overline{\psi}$ *(es decir, en* ψ *no aparece* $^\circ$*)* *y* $\Sigma \vdash \psi$, *entonces* $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \vdash_{cpc} \psi$.

Demostración: Si $\psi = \overline{\psi}$, entonces $\mathbf{P}(\psi) = \psi$. Se aplica entonces el corolario anterior [2.17] , sustituyendo en su conclusión $\mathbf{P}(\psi)$ por ψ .

□□

[2.19] OBSERVACION: En el ejemplo [2.4] se mencionó que en **BLO-OC** no es posible deducir P_1 a partir de $\{P_1, \neg P_1\}$. El corolario anterior demuestra esta afirmación. A su vez, esto muestra que el sistema deductivo **BLO-OC** es paraconsistente.

□□

El resultado recíproco del Corolario [2.18] es también válido. Para establecerlo, utilizaremos lo siguiente:

[2.20] OBSERVACION (Monotonía): El sistema deductivo **BLO-OC** verifica la propiedad de monotonía: en efecto, si tenemos conjuntos cualesquiera de fórmulas K_1 y K_2 tales que $K_1 \subseteq K_2$, entonces para cualquier fórmula ϕ tenemos que $K_1 \vdash \phi \Longrightarrow K_2 \vdash \phi$, ya que a partir de K_2 se puede efectuar la misma demostración de ϕ que se efectuó a partir de K_1

□□

[2.21] **NOTACION:** Si Δ es un conjunto de fórmulas, la expresión $\neg\neg\Delta$ representará al conjunto $\{\neg\neg\delta \in \mathcal{FOR} : \delta \in \Delta\}$. En particular, utilizaremos frecuentemente el símbolo $\neg\neg\Sigma$, donde Σ representa, según hemos convenido, un conjunto de fórmulas en el lenguaje de CPC.

□□

[2.22] **LEMA:** *Si* $\Sigma \vdash_{cpc} \psi$, *entonces* $\Sigma \cup \neg\neg\Sigma \vdash \psi$ *y* $\Sigma \cup \neg\neg\Sigma \vdash \psi^o$.

Demostración: Antes de empezar la demostración, obsérvese que para que la expresión " $\Sigma \vdash_{cpc} \psi$ " tenga sentido, la fórmula ψ debe pertenecer, al igual que las fórmulas en Σ , al lenguaje de CPC.

Procedemos por inducción sobre la longitud de la demostración de ψ en CPC:

Demostraciones de longitud 1: Si en CPC se deduce ψ desde Σ en un sólo paso, entonces ψ es un axioma de CPC o bien $\psi \in \Sigma$. Tomando para CPC la axiomatización correspondiente a la de **BLO-OC**, en el primer caso tendríamos que $\psi \wedge \psi^o$ es axioma de **BLO-OC**, al que le podemos aplicar la regla de inferencia [2] para obtener ψ, o la regla [3] para obtener ψ^o. En el segundo caso, $\psi \in \Sigma$ nos dice que en $\Sigma \cup \neg\neg\Sigma$ están ψ y $\neg\neg\psi$; aplicando la regla de inferencia [5] obtenemos la fórmula ψ^o ; a partir de ella y de la premisa ψ, con la regla [1] obtenemos $\psi \wedge \psi^o$; y aplicándole a esta fórmula la regla de inferencia [2], obtenemos ψ ; luego, en este caso también se deducen ψ y ψ^o a partir de $\Sigma \cup \neg\neg\Sigma$.

Hipótesis inductiva: Suponemos que para toda fórmula ψ que puede ser demostrada en CPC a partir de Σ en menos de N pasos, se cumple que en **BLO-OC** es posible demostrar las fórmulas ψ y ψ^o a partir de $\Sigma \cup \neg\neg\Sigma$.

Paso inductivo: Sea ψ una fórmula que ha sido demostrada en CPC a partir de Σ en N pasos. Tenemos tres casos posibles: que ψ sea un axioma de CPC, que ψ pertenezca a Σ , o que ψ haya sido obtenida con *Modus Ponens*

a partir de fórmulas A y $A \to \psi$ que hayan aparecido con anterioridad en la deducción de ψ . Los casos primero y segundo fueron tratados en la primera etapa de la inducción (al estudiar las demostraciones de longitud 1). Y en el tercer caso, la hipótesis inductiva se aplica a las fórmulas A y $A \to \psi$, pues ellas fueron demostradas en CPC a partir de Σ en menos de N pasos; tenemos entonces que en **BLO-OC** pueden deducirse, a partir de $\Sigma \cup \neg\neg\Sigma$, las fórmulas A , A^o , $(A \to \psi)$ y $(A \to \psi)^o$; aplicando la regla de inferencia de *Pseudo - Modus Ponens*, obtenemos $\psi \wedge \psi^o$; y a partir de esta fórmula obtenemos, mediante las reglas de inferencia [2] y [3] , las fórmulas ψ y ψ^o .

□□

El siguiente es el resultado recíproco de [2.18] que habíamos anunciado:

[2.23] **LEMA:** *Si* $\psi = \overline{\psi}$ *(es decir, si en* ψ *no aparece* ^{o}*) y si* $\Sigma \nvdash \psi$, *entonces* $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash_{cpc} \psi$.

Demostración: El lema [2.22] nos indica que dado cualquier conjunto K integrado solamente por fórmulas escritas en lenguaje clásico (sin utilizar el símbolo de confirmación), tenemos $K \vdash_{cpc} \psi \implies K \cup \neg\neg K \vdash \psi$, y por lo tanto $K \cup \neg\neg K \nvdash \psi \implies K \nvdash_{cpc} \psi$. Tomando $K = \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$, se tiene $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \cup \neg\neg\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash \psi \implies \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash_{cpc} \psi$. Por otro lado, la monotonía de **BLO-OC** (Observación [2.20]) implica que para cualesquiera conjuntos K_1 y K_2 , si $K_1 \subseteq K_2$ entonces $K_1 \vdash \psi \implies K_2 \vdash \psi$, y por lo tanto si $K_1 \subseteq K_2$ entonces $K_2 \nvdash \psi \implies K_1 \nvdash \psi$. Aplicando esto al caso presente, como $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \subseteq \Sigma$ y $\neg\neg\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \subseteq \Sigma$, tenemos $\Sigma \nvdash \psi \implies \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \cup \neg\neg\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash \psi$. Reuniendo estos dos hechos que hemos indicado, obtenemos $\Sigma \nvdash \psi \implies \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \cup \neg\neg\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash \psi \implies \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash_{cpc} \psi$.

□□

Del Lema [2.22] se deduce también lo siguiente:

[2.24] **COROLARIO:** *En* **BLO-OC** *se pueden demostrar (sin premisas) todas las tautologías de CPC.*

Demostración: En CPC, si ψ es una tautología, entonces $\varnothing \vdash_{cpc} \psi$. Por el Lema [2.22] , en **BLO-OC** tenemos $\varnothing \cup \neg\neg\varnothing = \varnothing \vdash \psi$.

□□

El Corolario [2.24] demuestra la afirmación que se formuló al final de [2.4] (ver comentarios en el apéndice histórico-filosófico).

Los resultados anteriores serán utilizados para aplicar **BLO-OC** a bases de datos inconsistentes. En lo que resta del presente capítulo, veremos otros resultados referentes a **BLO-OC**, que no participarán directamente en la antedicha aplicación a bases de datos, pero que nos muestran diversos aspectos del funcionamiento del sistema deductivo **BLO-OC**.

Si se examinan las reglas de inferencia de Obtención, Adjunción y Desadjunción de Confirmación (reglas [1], [2], [3] y [5]), y si se desarrollan algunas demostraciones formales en **BLO-OC** (ver ejemplos [2.4]), uno puede formarse la impresión de que estas cuatro reglas de inferencia parecen tener que ver principalmente con el "desbloqueo" de las premisas, en el sentido de que una premisa no puede ser incluida directamente en una demostración, sino que debe ser "desbloqueada" mediante las mencionadas reglas, para poder ser "admitida" en la demostración. Si ese fuera el principal papel de dichas reglas de inferencia, el poder deductivo "efectivo" de esta lógica parecería residir esencialmente en la regla de *Pseudo-Modus Ponens*. Estas observaciones resultan ser correctas, si las interpretamos formalmente en el sentido del lema siguiente:

[2.25] **LEMA:** *A partir de un conjunto de premisas en lenguaje clásico (sin el símbolo °), si no se usa Pseudo-Modus Ponens, en* **BLO-OC** *sólo es posible*

deducir fórmulas cuya parte principal es, o bien la parte principal de un axioma de **BLO-OC**, *o bien una premisa cuya doble negación es también premisa.*

Demostración: Probaremos el lema por inducción sobre la longitud de las demostraciones de fórmulas a partir del conjunto (arbitrario pero fijo) Σ :

Demostraciones de longitud 1: Si ψ es una fórmula que ha sido demostrada a partir de Σ en un sólo paso, entonces ψ es un axioma de **BLO-OC** o ha sido obtenida con la regla de inferencia [5] (*Obtención de Confirmación*), porque para aplicar cualquiera de las otras cuatro reglas de inferencia se requeriría que las premisas contuvieran el símbolo de confirmación ^{o}. En el primer caso se cumple directamente que su parte principal es la de un axioma de **BLO-OC** ; y en el segundo caso, ψ es de forma ϕ^o con ϕ y $\neg\neg\phi$ en Σ , de modo que su parte principal es una premisa cuya doble negación es también premisa.

Hipótesis inductiva: Supongemos que para toda fórmula que puede ser demostrada en **BLO-OC** a partir de Σ sin usar *Pseudo-Modus Ponens* en menos de N pasos, se cumple que su parte principal es, o bien la parte principal de un axioma de **BLO-OC**, o bien una premisa cuya doble negación es también premisa.

Paso inductivo: Sea ψ fórmula demostrada a partir de Σ en N pasos sin usar *Pseudo-Modus Ponens*. Son posibles los siguientes casos:

(a) Si ψ es un axioma de **BLO-OC**, se cumple directamente que su parte principal es la parte principal de un axioma de **BLO-OC**.

(b) Si ψ fue obtenida utilizando la regla de inferencia [1] , entonces ψ es de forma $\phi \wedge \phi^o$ y fue obtenida a partir de las fórmulas ϕ y ϕ^o. Como ésta última no puede ser premisa (por contener el símbolo de confirmación), había aparecido con anterioridad en la deducción, y por lo tanto había sido demostrada en menos de N pasos. Se le aplica entonces la hipótesis inductiva, con lo cual su parte principal, que es también la parte principal de ψ , cumple con lo que se quiere (es decir, o bien es la parte principal de un axioma de **BLO-OC**, o bien es una premisa cuya doble negación es también premisa).

(c) Si ψ fue obtenida mediante la regla de inferencia [2] , entonces la fórmula $\psi \wedge \psi^o$ había aparecido en la deducción (no puede ser una premisa), de modo que se le aplica la hipótesis inductiva, con lo que su parte principal, que es ψ , o bien es la parte principal de un axioma de **BLO-OC**, o bien es una premisa

cuya doble negación es también premisa. Ahora bien, esto significa que ψ no es de forma ϕ^o ni de forma $\phi \wedge \phi^o$, porque la parte principal de cada axioma de **BLO-OC** tiene como conectivo principal el de implicación (luego no es de forma ϕ^o ni de forma $\phi \wedge \phi^o$), y en las premisas no aparece el símbolo de confirmación ^{o} . Por lo tanto, al no ser ψ de ninguna de esas dos formas, se tiene que $\mathbf{P}(\psi) = \psi$; y con ello tenemos que $\mathbf{P}(\psi)$ cumple con lo que se quiere (es decir, o bien es la parte principal de un axioma de **BLO-OC**, o bien es una premisa cuya doble negación es también premisa).

(d) Si ψ fue obtenida mediante la regla [3] , entonces ψ es de forma ϕ^o y la fórmula $\phi \wedge \phi^o$ había aparecido en la deducción (no puede ser una premisa), luego por hipótesis inductiva se tiene que la parte principal de $\phi \wedge \phi^o$, que es también la parte principal de ψ , cumple con lo que se quiere.

(e) Si ψ fue obtenida mediante la regla [5] , entonces ψ es de forma ϕ^o y fue obtenida a partir de las fórmulas ϕ y $\neg\neg\phi$. Casos posibles: Primero, si ϕ y $\neg\neg\phi$ son premisas (si están ambas en Σ), entonces la parte principal de ψ , que es ϕ , es una premisa cuya doble negación es también premisa. Segundo, si $\neg\neg\phi$ es una premisa pero ϕ no lo es, a ϕ se le aplica la hipótesis inductiva, con lo que su parte principal o bien es la parte principal de un axioma de **BLO-OC**, o bien es una premisa cuya doble negación es también premisa; pero ϕ no puede contener el símbolo de confirmación, porque $\neg\neg\phi \in \Sigma$, por lo que $\mathbf{P}(\phi) = \phi = \mathbf{P}(\phi^o) = \mathbf{P}(\psi)$, es decir que la parte principal de ψ cumple con lo que se quiere. Y tercero, veamos que no puede ocurrir que $\neg\neg\phi$ no sea una premisa: si suponemos que no es premisa, se le aplica la hipótesis inductiva, con lo cual su parte principal, que es ella misma, o bien es la parte principal de un axioma de **BLO-OC**, o bien es una premisa cuya doble negación es también premisa; pero esto último contradice la suposición de que $\neg\neg\phi$ no es una premisa, y lo primero no puede ser, porque ningún axioma de **BLO-OC** tiene a la negación como conectivo principal de su parte principal. Luego no puede ocurrir que $\neg\neg\phi$ no sea una premisa. Con esto se han cubierto todos los casos posibles para ψ obtenida mediante la regla [5].

□□

[Vemos, entonces, que sin usar *Pseudo - Modus Ponens* no se pueden deducir ni siquiera las tautologías clásicas que no son parte principal de alguno de los axiomas de **BLO-OC**.]

Veremos a continuación los tres últimos resultados de este capítulo. El lema siguiente se utilizará para demostrar el Teorema [2.27], el cual, según se comentará a continuación del Corolario [2.28], entrega un resultado significativo referente a las demostraciones en **BLO-OC** de las fórmulas simplificadas (definición [2.15]), dentro de las cuales están las escritas en lenguaje clásico (sin ^{o}):

[2.26] LEMA: (a) *Si* $\Sigma \vdash \phi^o$, *entonces* $\Sigma \vdash \phi$. *Además, para cada demostración* $\mathcal{D}$ *de la fórmula* ϕ^o *a partir de* Σ , *si* $\phi \notin \Sigma$ *o si en* $\mathcal{D}$ *no se usó la regla de inferencia* [5] *(Obtención de Confirmación), entonces existe una demostración de* ϕ *a partir de* Σ *de longitud menor o igual a la de* $\mathcal{D}$, *y en la que sólo aparecen fórmulas que figuran en* $\mathcal{D}$ *(salvo, probablemente, la conclusión* ϕ *).*

(b) *Si* $\vdash \phi^o$, *entonces* $\vdash \phi$. *Además, para cada demostración* $\mathcal{D}$ *de* ϕ^o, *existe una demostración de* ϕ *de longitud menor o igual a la de* $\mathcal{D}$, *y en la que sólo aparecen fórmulas que figuran en* $\mathcal{D}$ *(salvo, probablemente, la conclusión* ϕ*).*

Demostración: (a) La fórmula ϕ^o ha sido deducida (en $\mathcal{D}$) usando la regla de inferencia [3] o la regla [5] : en efecto, la fórmula ϕ^o no se obtuvo aplicando la regla de inferencia [1] ni la regla [4] , pues estas dos reglas producen fórmulas de forma $\xi \wedge \xi^o$, cuyo conectivo principal es $\wedge$ y no ^{o} , mientras que el de ϕ^o es ^{o} ; por igual motivo ϕ^o no es un axioma; y el corolario [2.13] garantiza que ϕ^o no se obtuvo con la regla de inferencia [2].

Supongamos que ϕ^o fue obtenida utilizando la regla [3]; esto significa que se había deducido la fórmula $\phi \wedge \phi^o$ (en $\mathcal{D}$, luego con longitud menor a la de $\mathcal{D}$), con lo que podemos aplicar la regla de inferencia [2] a $\phi \wedge \phi^o$, obteniendo una demostración de ϕ de longitud no mayor que la de $\mathcal{D}$ y en la que sólo figuran fórmulas que aparecen en $\mathcal{D}$ (salvo, probablemente, la conclusión ϕ).

Supongamos ahora que ϕ^o se obtuvo mediante la regla [5] y que ϕ no es una premisa ($\phi \notin \Sigma$). Entonces ϕ ya había sido deducida en $\mathcal{D}$, y por lo tanto, con longitud menor que la de $\mathcal{D}$ y usando sólo fórmulas que figuran en $\mathcal{D}$.

Finalmente, supongamos que ϕ^o se obtuvo mediante la regla [5] y que $\phi \in \Sigma$. Entonces con la premisa ϕ y la fórmula ϕ^o que se ha deducido con $\mathcal{D}$, podemos usar la regla [1] para agregar a $\mathcal{D}$ la fórmula $\phi \wedge \phi^o$, y aplicando ahora la regla [2] se deduce ϕ. Obsérvese que esta demostración de ϕ tiene dos pasos más que $\mathcal{D}$, y las dos fórmulas nuevas que en ella aparecen, $\phi \wedge \phi^o$ y la conclusión ϕ, probablemente no figuraban en $\mathcal{D}$.

(b) En este caso no existe conjunto Σ de premisas, de modo que la demostración de la parte (a) se aplica de nuevo, pero ahora sin el tercer caso.

□□

[2.27] **TEOREMA:** *Para cualquier fórmula ψ del lenguaje proposicional de bloqueo (básico), si $\Sigma \vdash \psi$ entonces $\Sigma \vdash \mathrm{S}(\psi)$. Más aún, en tal caso existe una demostración de $\mathrm{S}(\psi)$ en la que solamente aparecen* S*-fórmulas (fórmulas simplificadas).*

Demostración: Procederemos por inducción sobre la longitud de las demostraciones de fórmulas a partir del conjunto Σ:

Demostraciones de longitud 1: Si ψ ha sido demostrada a partir de Σ en un sólo paso, entonces ψ es un axioma o ha sido obtenida mediante la regla de inferencia [5], pues las otras cuatro reglas de inferencia no pueden aplicarse sobre premisas que no contienen el símbolo de confirmación.

Si ψ es un axioma de **BLO-OC**, hemos visto en la Observación [2.16] que su simplificación también lo es; por consiguiente $\mathrm{S}(\psi)$, como todo axioma, puede ser deducida a partir de Σ en un único paso, en el que en este caso figura una fórmula simplificada (ella misma), es decir que existe una S-deducción de $\mathrm{S}(\psi)$ a partir de Σ. Y si ψ fue obtenida mediante la regla [5], entonces $\psi = \phi^o$ con $\phi \in \Sigma$ y $(\neg\neg\phi) \in \Sigma$; como las premisas no contienen el símbolo de confirmación, esto significa que $\phi = \overline{\phi}$, y por lo tanto $\mathrm{S}(\psi) = (\overline{\phi})^o = \phi^o = \psi$, de manera que la demostración de ψ en un sólo paso, a partir de Σ, es una demostración de $\mathrm{S}(\psi)$ en un sólo paso, en el que figura una S-fórmula (ella misma), de modo que es una S-deducción de $\mathrm{S}(\psi)$ a partir de Σ.

Hipótesis inductiva: Suponemos que si una fórmula puede ser demostrada a partir de Σ en menos de N pasos, entonces existe una S-demostración de su simplificación a partir de Σ .

Paso inductivo: Sea ψ una fórmula que ha sido demostrada a partir de Σ en N pasos. Son posibles los siguientes casos:

(a) Si ψ no es de forma ϕ^o ni de forma $\phi \wedge \phi^o$, entonces $\mathrm{S}(\psi) = \overline{\psi}$, y además ψ ha sido obtenida mediante la regla de inferencia [2], porque todas las otras reglas producen fórmulas de forma ϕ^o o de forma $\phi \wedge \phi^o$, esta última correspondiente también a los axiomas de **BLO-OC**, por lo que ψ tampoco es axioma. En consecuencia, en la demostración de ψ había aparecido la fórmula $\psi \wedge \psi^o$, la que por lo tanto puede deducirse a partir de Σ en menos de N pasos. Por hipótesis inductiva, existe una S-deducción a partir de Σ de la fórmula $\mathrm{S}(\psi \wedge \psi^o) = \overline{\psi} \wedge (\overline{\psi})^o$; si prolongamos esta S-demostración aplicando a $\overline{\psi} \wedge (\overline{\psi})^o$ la regla [2], obtenemos una deducción de $\overline{\psi} = \mathrm{S}(\psi)$ en la que figuran solamente S-fórmulas.

(b) Si ψ es de forma $\phi \wedge \phi^o$, entonces $\mathrm{S}(\psi) = \overline{\phi} \wedge (\overline{\phi})^o$, y la fórmula $\psi = \phi \wedge \phi^o$ ha sido obtenida con la regla [1], o con la regla [2], o con la regla [4], o es un axioma (las otras dos reglas de inferencia producen fórmulas cuyo conectivo principal es ^{o}). Veamos estos cuatro casos posibles:

(b.1) El caso de ψ axioma de **BLO-OC** ya fue cubierto en el primer paso de esta inducción, al estudiar las deducciones de longitud 1.

(b.2) Si la fórmula $\psi = \phi \wedge \phi^o$ fue obtenida mediante la regla [2], entonces en la deducción de ψ había aparecido la fórmula $\psi \wedge \psi^o = (\phi \wedge \phi^o) \wedge (\phi \wedge \phi^o)^o$; por hipótesis inductiva tenemos una S-demostración, a partir de Σ , de la fórmula $\mathrm{S}(\psi \wedge \psi^o) = \mathrm{S}([\phi \wedge \phi^o] \wedge [\phi \wedge \phi^o]^o) = (\overline{\phi \wedge \phi^o}) \wedge (\overline{\phi \wedge \phi^o})^o = (\overline{\phi} \wedge \overline{\phi}) \wedge (\overline{\phi} \wedge \overline{\phi})^o$. Agreguemos a dicha S-deducción las siguientes S-fórmulas, que primero transcribiremos y luego justificaremos:

(1) $\quad (\overline{\phi} \wedge \overline{\phi}) \wedge (\overline{\phi} \wedge \overline{\phi})^o$

(2) $\quad (\overline{\phi} \wedge \overline{\phi})$

(3) $\quad (\overline{\phi} \wedge \overline{\phi})^o$

(4) $\quad [\,(\overline{\phi} \wedge \overline{\phi}) \rightarrow \overline{\phi}\,] \wedge [\,(\overline{\phi} \wedge \overline{\phi}) \rightarrow \overline{\phi}\,]^o$

(5) $\quad [\,(\overline{\phi} \wedge \overline{\phi}) \rightarrow \overline{\phi}\,]$

(6) $\quad [\,(\overline{\phi} \wedge \overline{\phi}) \rightarrow \overline{\phi}\,]^o$

(7) $\quad \overline{\phi} \wedge (\overline{\phi})^o$

[Justificación: fórmula S-demostrable por hipótesis inductiva; regla de inferencia[2]; regla [3]; axioma [A4]; regla[2]; regla [3]; y regla 4 (*Pseudo - Modus Ponens*) aplicada sobre pasos (2), (3), (5) y (6).]

Como hemos tomado una S-deducción y le hemos agregado sólo S-fórmulas, lo que hemos obtenido es una S-demostración, a partir de Σ , de la fórmula $\overline{\phi} \wedge (\overline{\phi})^o = \mathrm{S}(\psi)$.

(b.3) Si la fórmula $\psi = \phi \wedge \phi^o$ fue obtenida mediante la regla [1], en la deducción habían aparecido las fórmulas ϕ y ϕ^o , las que por lo tanto pueden ser demostradas a partir de Σ en menos de N pasos. Por hipótesis inductiva, tenemos S-deducciones de las fórmulas $\mathrm{S}(\phi)$ y $\mathrm{S}(\phi^o)$. $\mathrm{S}(\phi^o)$ es $(\overline{\phi})^o$; y dependiendo de si ϕ es de forma ξ^o, de forma $\xi \wedge \xi^o$, o de otra forma, la fórmula $\mathrm{S}(\phi)$ puede ser respectivamente $(\overline{\xi})^o$, $\overline{\xi} \wedge (\overline{\xi})^o$ ó $\overline{\phi}$:

Si $\mathrm{S}(\phi) = \overline{\phi}$, lo que dice la hipótesis inductiva es que se tienen S-deducciones de las fórmulas $\overline{\phi}$ y $(\overline{\phi})^o$ a partir de Σ , de modo que uniendo tales demostraciones y aplicando a continuación la regla [1] a las fórmulas $\overline{\phi}$ y $(\overline{\phi})^o$, se obtiene una S-demostración de la fórmula $\overline{\phi} \wedge (\overline{\phi})^o = \mathrm{S}(\psi)$ a partir de Σ .

El caso $\mathrm{S}(\phi) = (\overline{\xi})^o$ no puede ocurrir si ψ fue obtenida mediante la regla de inferencia [1], porque correspondería a $\phi = \xi^o$, con lo cual $\phi^o = (\xi^o)^o$, teniendo que haber aparecido esta fórmula en la deducción para que se haya obtenido $\phi \wedge \phi^o$ mediante la regla [1] (la fórmula no puede ser una premisa, pues contiene el símbolo de confirmación); pero el Corolario [2.13] nos asegura que no es posible deducir $(\xi^o)^o$ a partir de Σ .

Y si $\mathrm{S}(\phi) = \overline{\xi} \wedge (\overline{\xi})^o$, entonces se tiene que $\phi = \xi \wedge \xi^o$, y por consiguiente la fórmula $\mathrm{S}(\phi^o) = (\overline{\phi})^o$, que por hipótesis inductiva ha sido S-demostrada a partir de Σ, es $(\overline{\phi})^o = (\overline{\xi} \wedge \overline{\xi})^o$. Consideremos la fórmula $\overline{\xi} \wedge \overline{\xi}$; ella puede ser o no ser una premisa. Si lo es, tomamos la S-deducción que tenemos para $(\overline{\xi} \wedge \overline{\xi})^o$ a partir de Σ , y aplicamos la regla [1] a la premisa $(\overline{\xi} \wedge \overline{\xi})$ y a la

fórmula S-deducida $(\overline{\xi} \wedge \overline{\xi})^o$, con lo que obtenemos una S-demostración de $(\overline{\xi} \wedge \overline{\xi}) \wedge (\overline{\xi} \wedge \overline{\xi})^o = \overline{\phi} \wedge (\overline{\phi})^o = \mathrm{S}(\psi)$ a partir de Σ . Y si $\overline{\xi} \wedge \overline{\xi}$ no es una premisa, es decir si $(\overline{\xi} \wedge \overline{\xi}) \notin \Sigma$, aplicamos el Lema [2.26], con lo que obtenemos una demostración de $(\overline{\xi} \wedge \overline{\xi})$ en la que solamente aparecen fórmulas que pertenecían a la demostración que se tenía de $(\overline{\xi} \wedge \overline{\xi})^o$, salvo, probablemente, la conclusión $\overline{\xi} \wedge \overline{\xi}$; la demostración de $(\overline{\xi} \wedge \overline{\xi})^o$ por hipótesis inductiva contiene solamente S-fórmulas, y $\overline{\xi} \wedge \overline{\xi}$ es una S-fórmula, así que tenemos una S-deducción de $\overline{\xi} \wedge \overline{\xi}$; aplicando la regla de inferencia [1] a las fórmulas S-demostradas $\overline{\xi} \wedge \overline{\xi}$ y $(\overline{\xi} \wedge \overline{\xi})^o$, obtenemos una S-deducción de $(\overline{\xi} \wedge \overline{\xi}) \wedge (\overline{\xi} \wedge \overline{\xi})^o = \overline{\phi} \wedge (\overline{\phi})^o = \mathrm{S}(\psi)$ a partir de Σ .

(b.4) Si la fórmula $\psi = \phi \wedge \phi^o$ fue obtenida mediante la regla [4], esta regla se aplicó a fórmulas $A \to \phi$, $(A \to \phi)^o$, A y A^o. Las fórmulas $(A \to \phi)^o$ y A^o no pueden ser premisas, de modo que necesariamente habían aparecido en la demostración de ψ , por lo que podemos aplicarles la hipótesis inductiva, y con ello obtenemos S-demostraciones de las fórmulas $\mathrm{S}([A \to \phi]^o) = (\overline{A} \to \overline{\phi})^o$ y $\mathrm{S}(A^o) = (\overline{A})^o$, a partir de Σ. Las fórmulas $A \to \phi$ y A pueden ser premisas o pueden haber sido deducidas. Esto nos conduce a las siguientes posibilidades:

Si $A \to \phi$ es una premisa, entonces no contiene el símbolo de confirmación, de modo que $A \to \phi = \overline{A} \to \overline{\phi}$, y por lo tanto la fórmula $\overline{A} \to \overline{\phi}$ es una premisa; si en cambio $A \to \phi$ no es una premisa, entonces ella ha aparecido en la demostración de ψ , y por lo tanto podemos aplicarle la hipótesis inductiva, con lo que tenemos una S-demostración de la fórmula $\mathrm{S}(A \to \phi) = \overline{A} \to \overline{\phi}$; en resumen, o bien la fórmula $\overline{A} \to \overline{\phi}$ es una premisa, o bien ella puede ser S-demostrada a partir de Σ .

Con ello, si pudiésemos contar con una S-demostración a partir de Σ de la fórmula $\overline{A}$, o si ella fuese una premisa, podríamos aplicarle la regla de inferencia [4] (*Pseudo - Modus Ponens*) a ella, a $\overline{A} \to \overline{\phi}$ (que, como acabamos de ver, o bien es una premisa, o bien puede ser S-demostrada a partir de Σ), y a las dos fórmulas que ya tenemos S-deducidas en virtud de la hipótesis inductiva, obteniéndose de esta manera una S-demostración de $\overline{\phi} \wedge (\overline{\phi})^o = \mathrm{S}(\psi)$. Veamos entonces que, efectivamente, o bien $\overline{A}$ es un premisa, o bien puede ser S-deducida a partir de Σ :

Si la fórmula $\overline{A}$ es una premisa (independientemente de si A lo es o no), entonces, según acabamos de explicar, tenemos una S-demostración de $\mathrm{S}(\psi)$ a partir de Σ. Y si $\overline{A}$ no es una premisa, es decir si $\overline{A} \notin \Sigma$, entonces podemos aplicar el Lema [2.26] a la S-deducción que tenemos de la fórmula $(\overline{A})^o$, con lo que obtenemos una demostración de $\overline{A}$ en la que solamente aparecen

fórmulas que pertenecían a la S-demostración que se tenía de $(\overline{A})^o$, salvo probablemente la conclusión $\overline{A}$, que es una S-fórmula, de modo que tenemos una S-demostración de $\overline{A}$ a partir de Σ ; según hemos explicado, esto nos provee de una S-demostración de la fórmula $\mathrm{S}(\psi)$ a partir de Σ .

(c) Finalmente, si ψ es de forma ϕ^o , entonces $\mathrm{S}(\psi) = (\overline{\phi})^o$, y la fórmula $\psi = \phi^o$ ha sido obtenida aplicando la regla de inferencia [2], o la regla [3], o la regla [5] (las otras dos reglas de inferencia producen fórmulas cuyo conectivo principal es $\wedge$ y no ^{o} ; y, también por conectivo principal, $(\overline{\phi})^o$ no es axioma). Veamos los tres casos posibles:

(c.1) La fórmula $\psi = \phi^o$ no pudo ser obtenida mediante la regla [2], porque de ser así, en la deducción habría aparecido la fórmula $(\phi^o) \wedge (\phi^o)^o$, y el Corolario [2.13] garantiza que esto no es posible.

(c.2) Si la fórmula $\psi = \phi^o$ fue obtenida mediante la regla [3], entonces en la deducción había aparecido la fórmula $\phi \wedge \phi^o$, y por lo tanto podemos aplicar a esta última la hipótesis inductiva, con lo que tenemos una S-demostración de $\mathrm{S}(\phi \wedge \phi^o) = \overline{\phi} \wedge (\overline{\phi})^o$ a partir de Σ ; aplicando ahora la regla de inferencia [3], obtenemos una S-demostración de la fórmula $(\overline{\phi})^o = \mathrm{S}(\psi)$ a partir de Σ .

(c.3) Para terminar, veamos el caso en que $\psi = \phi^o$ fue obtenida mediante aplicación de la regla de inferencia [5]. En este caso $\psi = \phi^o$ se obtuvo a partir de las fórmulas ϕ y $\neg\neg\phi$. Son entonces posibles los dos casos siguientes:

Primero, supongamos que la fórmula ϕ es una premisa, es decir $\phi \in \Sigma$. Esto quiere decir que ϕ no contiene el símbolo de confirmación, de modo que $\phi = \overline{\phi}$, y por consiguiente $\neg\neg\phi = \neg\neg(\overline{\phi})$. Si $\neg\neg(\overline{\phi}) \in \Sigma$, se aplica la regla de inferencia [5] a las premisas $(\overline{\phi})$ y $\neg\neg(\overline{\phi})$, obteniéndose una S-deducción (que consta de un sólo paso) de la fórmula $(\overline{\phi})^o = \mathrm{S}(\psi)$. Y si $\neg\neg(\overline{\phi}) \notin \Sigma$, es decir si $\neg\neg\phi \notin \Sigma$, entonces $\neg\neg\phi$ tuvo que aparecer en la demostración de $\psi = \phi^o$ (pues se usó para aplicar la regla [5]), de modo que podemos aplicarle la hipótesis inductiva, con lo que tenemos una S-demostración, a partir de Σ , de la fórmula $\mathrm{S}(\neg\neg\phi) = \neg\neg(\overline{\phi})$. A esta fórmula y a la premisa $\phi = \overline{\phi}$ les aplicamos la regla de inferencia [5], y de este modo obtenemos una S-demostración de la fórmula $(\overline{\phi})^o = \mathrm{S}(\psi)$ a partir de Σ .

Y segundo, supongamos la fórmula ϕ no es una premisa, es decir $\phi \notin \Sigma$. En este caso, como la fórmula $\psi = \phi^o$ se obtuvo a partir de ϕ y $\neg\neg\phi$, tenemos que ϕ había aparecido en la demostración de ψ , y por lo tanto pode-

mos aplicarle la hipótesis inductiva. Existe entonces una S-deducción de $\mathbf{S}(\phi)$. Dependiendo de si la fórmula ϕ es de forma ξ^o , de forma $\xi \wedge \xi^o$, o de otra forma, su simplificación $\mathbf{S}(\phi)$ puede ser de forma $(\overline{\xi})^o$, de forma $\overline{\xi} \wedge (\overline{\xi})^o$, o de forma $\overline{\phi}$. Veamos estos tres casos posibles:

Si $\mathbf{S}(\phi)$ es $\overline{\phi}$, notemos lo siguiente: si la fórmula $\neg\neg\phi$ es una premisa, entonces $\neg\neg\phi = \neg\neg(\overline{\phi})$, y si no es una premisa, tuvo que aparecer en la deducción (pues fue utilizada para aplicar la regla [5]), de modo que se le aplica la hipótesis inductiva y con ello se tiene una S-demostración de $\mathbf{S}(\neg\neg\phi) = \neg\neg(\overline{\phi})$; en resumen, disponemos de la fórmula $\neg\neg(\overline{\phi})$, o como premisa, o S-demostrada. Por lo tanto podemos aplicar la regla [5] a las fórmulas $\mathbf{S}(\phi) = \overline{\phi}$ (que ha sido S-deducida) y $\neg\neg(\overline{\phi})$ (que es premisa o ha sido S-deducida) , y de este modo obtenemos una S-demostración de $(\overline{\phi})^o = \mathbf{S}(\dot{\psi})$ a partir de Σ .

Si $\mathbf{S}(\phi)$ es $(\overline{\xi})^o$, entonces $\phi = \xi^o$, luego $\overline{\phi} = \overline{(\xi^o)} = \overline{\xi}$, lo que a su vez significa que $(\overline{\phi})^o = (\overline{\xi})^o$; por lo tanto, la S-demostración que tenemos para la fórmula $\mathbf{S}(\phi)$ es una S-demostración de la fórmula $(\overline{\xi})^o = (\overline{\phi})^o = \mathbf{S}(\psi)$.

Y si $\mathbf{S}(\phi)$ es $\overline{\xi} \wedge (\overline{\xi})^o$, entonces la fórmula ϕ es de forma $\xi \wedge \xi^o$, y hay que probar que existe una S-demostración de $\mathbf{S}(\psi) = (\overline{\phi})^o = (\overline{\xi \wedge \xi^o})^o = (\overline{\xi} \wedge \overline{\xi})^o$. Tomemos la S-deducción que tenemos para la fórmula $\mathbf{S}(\phi) = \overline{\xi} \wedge (\overline{\xi})^o$, y prolonguémosla del siguiente modo (los pasos serán justificados después):

(1) $\quad \overline{\xi} \wedge (\overline{\xi})^o$

(2) $\quad \overline{\xi}$

(3) $\quad (\overline{\xi})^o$

(4) $\quad [\ \overline{\xi} \to (\overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}])\ ] \ \wedge\ [\ \overline{\xi} \to (\overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}])\ ]^o$

(5) $\quad \overline{\xi} \to (\overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}])$

(6) $\quad [\ \overline{\xi} \to (\overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}])\ ]^o$

(7) $\quad (\ \overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}]\ ) \ \wedge\ (\ \overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}]\ )^o$

(8) $\quad \overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}]$

(9) $\quad (\ \overline{\xi} \to [\overline{\xi} \wedge \overline{\xi}]\ )^o$

(10) $(\overline{\xi} \wedge \overline{\xi}) \wedge (\overline{\xi} \wedge \overline{\xi})^o$

(11) $(\overline{\xi} \wedge \overline{\xi})^o$

[Justificación: fórmula S-demostrable por hipótesis inductiva; regla [2]; regla [3]; axioma [A3]; regla[2]; regla [3]; regla 4 (*Pseudo - Modus Ponens*) sobre pasos (2), (3), (5) y (6); regla [2]; regla[3]; regla 4 (*Pseudo - Modus Ponens*) sobre pasos (2), (3), (8) y (9); y regla [3] .]

Todas las fórmulas que hemos agregado a la S-deducción que teníamos para $\overline{\xi} \wedge (\overline{\xi})^o$ son fórmulas simplificadas; por lo tanto, hemos probado que existe una S-demostración para la fórmula $(\overline{\xi} \wedge \overline{\xi})^o = (\overline{\phi})^o = \mathbf{S}(\psi)$.

□□

[2.28] COROLARIO: *Si ψ es una* S*-fórmula (con lo que $\psi = \mathbf{S}(\psi)$, según Observación* [2.16]), *entonces en el sistema deductivo* **BLO-OC** *la fórmula ψ puede ser demostrada a partir de Σ si y sólo si puede ser* S*-demostrada a partir de Σ . En particular, si $\phi = \overline{\phi}$, es decir si ϕ es una fórmula escrita en el lenguaje de CPC, entonces en* **BLO-OC** *la fórmula ϕ puede demostrarse a partir de Σ si y sólo si puede* S*-demostrarse a partir de Σ .*

□□

El Teorema [2.27] nos revela, con respecto al funcionamiento deductivo de **BLO-OC**, un aspecto conceptualmente significativo: aunque la simplificación de una fórmula puede ser demostrada utilizando (en el transcurso de la deducción formal) fórmulas no simplificadas, el Teorema [2.27] (o, más precisamente, la demostración del teorema) nos dice que cada deducción tal puede ser reemplazada por otra en que sólo figuran S-fórmulas, de modo que para demostrar una fórmula

simplificada es innecesario utilizar fórmulas no simplificadas (esto es lo que nos dice el Corolario [2.28]). El conjunto de las S-fórmulas, entonces, es, en cierto sentido, deductivamente cerrado.

Además, el corolario anterior [2.28] constituye una herramienta para demostrar propiedades de fórmulas escritas en el lenguaje de CPC, que sean deducibles en **BLO-OC** a partir de premisas también escritas en lenguaje clásico, porque nos permite limitar nuestro análisis al conjunto de las S-demostraciones, el cual es muchísimo más manejable que el conjunto de todas las deducciones, cuando se intenta demostrar una propiedad de **BLO-OC** mediante inducción sobre la longitud de las demostraciones de fórmulas.

CAPÍTULO 2 - APÉNDICE

La noción de que no es posible que una afirmación sea verdadera y falsa al mismo tiempo es conocida como "principio de no contradicción" (aunque también, paradójicamente, como "principio de contradicción"). A través de la historia, este principio ha sido defendido a veces como un hecho que no requiere de una demostración por ser evidente por sí mismo, y otras veces se lo ha intentado demostrar. Por otro lado, diversos pensadores han desconfiado de él, en diversos sentidos y grados.

Un argumento destacable que se ha esgrimido en su defensa es conocido como "principio del Pseudo-Escoto"; consiste en indicar que a partir de una contradicción es posible deducir cualquier cosa, y por lo tanto, si admitimos una contradicción como "verdadera", el raciocinio se vuelve inútil, pues ya no es capaz de garantizar que si algo es demostrable ha de ser verdad, ya que se podría deducir tanto lo cierto como lo falso; es decir, la racionalidad pierde su capacidad de distinguir lo verdadero de lo erróneo. Así, por ejemplo, si una teoría científica contiene una contradicción, esta teoría predecirá, para cualquier experimento dado, que él arrojará como resultado cada valor posible, con lo que efectuar experimentos no refutará ni apoyará la teoría, lo que la vuelve inútil. Este argumento, entonces, plantea algo así como: "No puedo demostrar directamente que es imposible que existan contradicciones, pero si aceptamos una, nuestra capacidad de razonar se inutiliza; por este motivo no podemos aceptar contradicciones como verdaderas". (Nota: Este argumento se conoce también por su descripción en Latín , "ex contradictione sequitur quodlibet", "de una contradicción se sigue cualquier cosa").

A principios del siglo XX algunos pensadores tomaron conciencia de que este argumento asume, tácitamente, las leyes lógicas que permiten deducir cualquier cosa a partir de una contradicción; si se objetan estas leyes, el argumento ya no será válido. Esta observación inicia el recorrido hacia la creación de las lógicas paraconsistentes, las que se definen del modo siguiente: un sistema deductivo se dice **paraconsistente** si existe al menos una contradicción de la cual no se puede deducir formalmente, dentro del sistema, toda afirmación posible. Hay que destacar que esta definición no es tan clara como puede parecer a primera vista, pues la noción de contradicción presupone la de negación, respecto de la cual han existido intensas discusiones. Hay incluso autores que niegan que pueda existir tal cosa como una lógica paraconsistente, pues sostienen que un conectivo lógico

unario que no verifica determinadas propiedades, como la del Pseudo-Escoto, no merece ser llamado "negación". Sin enfrascarnos en tan interesante discusión, indicaremos que el calificativo de "paraconsistente" que hemos aplicado a la lógica **BLO-OC** aspira a ser aceptado en el siguiente sentido: el conectivo unario $\neg$ de este sistema deductivo no verifica el principio del Pseudo-Escoto, hay un amplio conjunto de fórmulas respecto de las que sintácticamente se comporta al estilo de la negación clásica, y, semánticamente, presentaremos en este trabajo interpretaciones en las que este conectivo refleja a la negación clásica, al menos tomada en cierto sentido o interpretación (sobre todo, en el capítulo 5).

* * * * *

En el sistema deductivo **BLO-OC** que hemos definido, hemos denominado "símbolo de confirmación" al conectivo unario $^\circ$, escrito como superíndice. Según se comentó en la nota siguiente a la definición [2.1], este símbolo fue ideado por Newton da Costa, aunque él lo utilizó como abreviatura, no como símbolo primitivo (no era parte del lenguaje proposicional); lo presentó en el contexto de un conjunto de sistemas deductivos, conocido como "jerarquía de sistemas deductivos C_n" (para cada número natural n, un sistema C_n, más C_ω); da Costa creó estos sistemas a principios de la década de 1960, y los presentó en 1963, en su tesis de promoción de la Universidad de Paraná (Brasil), bajo el título "Sistemas Formais Inconsistentes" [ref. 5]. Se considera que estos sistemas son el primer ejemplo de lógica paraconsistente plenamente desarrollada que se haya publicado. En ellos, si A es una fórmula, da Costa escribía A° para abreviar $\neg(A \wedge (\neg A))$; obsérvese que esta fórmula dice "no ocurre que A es cierto y falso a la vez"; en otras palabras, dentro de los sistemas C_n , que "admitían la posibilidad de contradicciones", A° significaba que la fórmula A no era contradictoria, o que era "consistente" ; algunos de los axiomas que da Costa incorporó a sus sistemas C_n se referían a este tipo de fórmulas, y lo que dichos axiomas indicaban era, vagamente hablando, que las fórmulas consistentes se comportan al estilo clásico. En este sentido, hay una cierta relación entre el uso que hace da Costa del símbolo $^\circ$ y el que tiene en **BLO-OC**, al menos en ciertos casos, como el que nos ocupará en el capítulo 3, y sobre todo en el capítulo 5, donde este símbolo entrará en juego para distinguir las afirmaciones "sobre las que hay acuerdo" de aquellas "sobre las que hay opiniones contradictorias". [Nota: diversos autores han adoptado el símbolo $^\circ$ de da Costa, algunos nuevamente como abreviatura, y otros como símbolo primitivo del lenguaje proposicional.]

Como dijimos, se considera que los sistemas deductivos C_n son el primer ejemplo de lógica paraconsistente plenamente desarrollada que se haya publicado. Con anterioridad, habían sido presentados otros sistemas deductivos que buscaban evitar que la presencia de una contradicción fuera automáticamente suficiente para que cualquier cosa fuese deducible, pero ellos no fueron completamente exitosos, en distintos sentidos. Los nombres de Jan Lukasiewicz, Nicolaj Vasiliev, Andrei Kolmogórov, Ingebrigt Johansson y Stanislav Jaśkowski, entre otros, aparecen en la ruta que había de desembocar en la obra de Newton da Costa. Una completísima semblanza de la historia y la filosofía de la lógica paraconsistente es expuesta en el magnífico trabajo de Andrés Bobenrieth "Inconsistencias ¿Por qué no?" [referencia 3] (el cual, me permito agregar, es en mi opinión altamente recomendable para el lector interesado en el tema).

* * * * *

Al final de [2.4] se afirma que en **BLO-OC** se puede demostrar (sin premisas) toda tautología clásica. Esta afirmación se demuestra en [2.24]. Un sistema deductivo con tal característica es llamado una "lógica paraconsistente clásica" en [referencia 2]; el hecho de que **BLO-OC** sea un sistema deductivo paraconsistente clásico será fundamental para la aplicación que de él efectuaremos en el próximo capítulo; en realidad, es casi seguro que esta aplicación sería imposible si **BLO-OC** no fuese paraconsistente clásico; explicaremos este punto en el apéndice histórico-filosófico del mencionado próximo capítulo (el tercero).

* * * * *

En [2.5] se hace notar que la definición de deducción formal que hemos asignado a **BLO-OC**, si se aplica a CPC, es equivalente a la habitual. En mi opinión, este hecho es muy importante filosóficamente (¡matemáticamente no!): en efecto, si uno utiliza una definición de deducción formal muy diferente de la clásica, no es difícil que se obtengan lógicas con características singulares o

curiosas, por ejemplo lógicas paraconsistentes; el mérito, por así decirlo, consiste en lograr resultados nuevos usando la definición clásica de demostración formal, o alguna sumamente similar, pues la clásica parece ser la definición "correcta" de deducción; de ahí la importancia de que para CPC nuestra definición sea equivalente a la usual.

Para terminar, mencionemos el siguiente detalle referente a [2.5] : en dicho resultado se afirma que toda tautología clásica seguiría siendo deducible en CPC si se usara la definición de deducción formal de **BLO-OC** en lugar de la usual; aunque esta afirmación no se demostró, debería ser clara, ya que este cambio de definición consistiría en suprimir en CPC la posibilidad de incluir directamente premisas en las deducciones, pero permitiendo ahora aplicar *Modus Ponens* no sólo a fórmulas que hayan aparecido en la demostración con anterioridad sino también directamente a premisas (todo esto, sin alterar la regla de inferencia *Modus Ponens*, ni los axiomas); dado que ambos cambios se refieren al uso de las premisas que se estén considerando, claramente ellos no afectan a las deducciones formales que se efectúan sin utilizar premisas, y por lo tanto las tautologías siguen siendo deducibles como tesis de la lógica.

CAPÍTULO 3

BASES DE DATOS INCONSISTENTES

a) Nociones sobre bases de datos

En este capítulo utilizaremos **BLO-OC** para describir ciertos aspectos del comportamiento de las bases de datos inconsistentes, lo que a su vez nos proveerá de una interpretación "natural" para **BLO-OC** ; según se mencionó en la introducción del presente trabajo, uno de los aspectos que nos interesan aquí, es proponer interpretaciones naturales para sistemas deductivos paraconsistentes.

Daremos a continuación una noción de los elementos de computación teórica necesarios para comprender este capítulo. Dado que nuestro trabajo no se centra en el aspecto computacional, sino en el lógico-matemático, las ideas computacionales serán expuestas de manera intuitiva, formalizando solamente lo que es indispensable. El lector puede encontrar detalles formales en [referencia 9].

Lo que coloquialmente se suele denominar base de datos, en computación teórica se llama **instancia de base de datos** (la base de datos es, intuitivamente hablando, el "molde abstracto", que se materializa en una instancia de base de datos determinada). En este trabajo nos interesaremos en particular en las bases de datos denominadas "relacionales": dada una coleción de objetos $c_1, c_2, c_3, \ldots$ (probablemente infinita), una **instancia de base de datos relacional** es una colección finita de afirmaciones de forma $R(c_{\alpha_1}, c_{\alpha_2}, \ldots, c_{\alpha_t})$, las que interpretamos como "los objetos $c_{\alpha_1}, c_{\alpha_2}, \ldots, c_{\alpha_t}$ están en relación R entre sí". En la práctica, gran parte de las bases de datos son relacionales.

Por ejemplo, si en una instancia de base de datos relacional se quiere guardar la información de que el número de cédula A_1 corresponde a la persona de nombre K_1, y el número de cédula A_2 corresponde a la persona K_2, lo que hacemos es dar un nombre, digamos nuevamente R, a la relación de corresponder un número de cédula a determinada persona, e incluimos en la instancia relacional las afirmaciones $R(A_1, K_1)$ y $R(A_2, K_2)$, que codifican la información "A_1 es la cédula de K_1" y "A_2 es la cédula de K_2". Si una instancia de bases de datos DB contiene solamente esta información (si consta solamente de estas dos afirmaciones), escribimos $DB = \{ R(A_1, K_1) , R(A_2, K_2) \}$.

Obsérvese que aunque la colección incial de objetos $c_1, c_2, c_3, \ldots$. sea infinita, en una instancia de base se datos sólo figura una cantidad finita de ellos.

En una base de datos relacional habitualmente se ponen restricciones, denominadas **restricciones de integridad**, que suelen ser presentadas mediante fórmulas de lógica de primer orden (lógica predicativa). Por ejemplo, a un número de cédula no se le debe asignar más de un nombre, lo que se puede expresar mediante la fórmula de primer orden $\forall x \forall y \forall z(\neg R(x,y) \vee \neg R(x,z) \vee y = z)$. Las preguntas formuladas a la base de datos también pueden expresarse como fórmulas de primer orden. Por ejemplo, para preguntar quién tiene el número de cédula A, escribimos $R(A,x)$, siendo la respuesta aquella constante K del lenguaje de primer orden que al tomar el lugar de la variable "x" produce una fórmula $R(A,K)$ que nombra a una afirmación que pertenece a la instancia de base de datos. También se pueden formular preguntas de tipo "verdadero o falso" a la instancia, utilizando oraciones (fórmulas de primer orden sin variables libres); en este caso, la instancia de base de datos "asume que todo lo que no sabe, es falso"; por ejemplo, si una determinada afirmación $R(A,K)$ no pertenece a una instancia DB, ésta responde, si se le pregunta, que tal afirmación es falsa. Para formalizar estas ideas, así como para dar sentido a la pregunta de si determinada instancia de base de datos satisface cierta fórmula de primer orden, se utiliza el concepto de satisfacción de una oración en una instancia de base de datos. Nosotros no utilizaremos este concepto (sino una variación de él), pero mencionaremos que su definición, recursiva, es "la clásica", es decir $DB \models R(d_1, \ldots, d_t)$ significa que $R(d_1, \ldots, d_t) \in DB$; $DB \models \neg\phi$ significa que no es cierto que $DB \models \phi$; $DB \models \forall x_1 \ldots \forall x_t(\phi(x_1, \ldots, x_t))$ significa que para toda tupla $d_1, \ldots, d_t$ se cumple $DB \models \phi(d_1, \ldots, d_t)$; etc. Obsérvese que hemos hablado de definir satisfacción de una oración, es decir con todas las variables cuantificadas, y no de una fórmula cualquiera.

Se esperaría entonces que una instancia de base de datos DB "cumpla por construcción" con las fórmulas que describen sus restricciones de integridad. Sin embargo, puede ocurrir que DB cumpla con la(s) negación(es) de alguna(s) de tales fórmulas (utilizando la definición de satisfacción recién mencionada). Por ejemplo, al reunir información de bases de datos preexistentes, puede que a un número de cédula se le terminen asignando dos nombres, y en ese caso se cumplirá la fórmula $\exists x \exists y \exists z(R(x,y) \wedge R(x,z) \wedge y \neq z)$, y por lo tanto también su equivalente (en lógica clásica de primer orden) $\neg\Big[\forall x \forall y \forall z(\neg R(x,y) \vee \neg R(x,z) \vee y = z)\Big]$, que es la negación de la restricción antes anotada. Para referirnos a una situación como ésta, hablamos de bases de datos inconsistentes: una instancia de base de

datos DB se dirá **inconsistente con un conjunto dado de restricciones de integridad** si DB no satisface dicho conjunto de fórmulas (lo que es equivalente a decir que satisface la negación de alguna de las restricciones, en virtud de la ya aludida definición, de tipo clásico, de la expresión $DB \models \neg\phi$).

La palabra "inconsistente" que aparece en la definición anterior no es solamente un nombre: los métodos computacionales habituales de tratamiento de bases de datos utilizan lógica clásica de primer orden, e incorporan las restricciones de integridad a la descripción, de manera que en el ejemplo del párrafo anterior las oraciones contradictorias $\neg\left[\forall x\forall y\forall z(\neg R(x,y) \vee \neg R(x,z) \vee y = z)\right]$ y $\forall x\forall y\forall z(\neg R(x,y) \vee \neg R(x,z) \vee y = z)$ surgen ambas en la descripción, y por lo tanto, como toda contradicción, trivializan el sistema estudiado con lógica clásica. Esto hace que los métodos habitualmente utilizados en computación teórica no puedan usarse directamente al tratar instancias que sean inconsistentes con determinadas restricciones de integridad; el único modo de insistir en usar lógica clásica es eliminar toda la información que contradice a las restricciones. Algunos autores se han interesado en encontrar métodos alternativos, que utilicen otras lógicas, para así intentar evitar la eliminación de información. Veremos que **BLO-OC** es capaz de describir instancias inconsistentes sin eliminar la información contradictoria.

Para desarrollar nuestro trabajo, usaremos la noción de **conjunto de reparaciones de una base de datos con respecto a un conjunto de restricciones de integridad**. Este concepto fue introducido por Marcelo Arenas, Leopoldo Bertossi y Jan Chomicki, en [ref. 10]. De dicho trabajo tomamos las siguientes definiciones, que primero transcribiremos y posteriormente comentaremos:

DEFINICIONES (Arenas, Bertossi y Chomicki; 1999): Dadas dos instancias de base de datos D_1 y D_2, la **distancia** $\Delta(D_1, D_2)$ entre ellas, se define como su diferencia simétrica: $\Delta(D_1, D_2) = (D_1 - D_2) \cup (D_2 - D_1)$. Dada una instancia fija D, definimos el orden parcial $\leq_D$ de este modo: $D_1 \leq_D D_2$ significa $\Delta(D, D_1) \subseteq \Delta(D, D_2)$. Finalmente, dada una instancia DB, decimos que una instancia de base de datos DB' es una **reparación de DB con respecto a un conjunto I de restricciones de integridad** si DB' satisface las fórmulas del conjunto I y DB' es $\leq_{DB}$-minimal en la clase de las instancias que satisfacen I.

La idea del concepto de reparación es la siguiente: Consideremos la instancia $DB = \{R(A, K_1), R(A, K_2), R(A, K_3), R(B, K_2)\}$, con la restricción de la que ya hemos hablado, es decir $\forall x\forall y\forall z(\neg R(x,y) \vee \neg R(x,z) \vee y = z)$; DB

es inconsistente con esta restricción. Si nos interesan solamente las instancias que son consistentes con ella, podríamos eliminar de DB el segundo, tercer y cuarto elemento, obteniendo la instancia $DB_1 = \{R(A, K_1)\}$, que es consistente con la restricción; otra posibilidad es eliminar sólo el segundo y el tercer elemento, obteniendo $DB_2 = \{R(A, K_1), R(B, K_2)\}$; otra opción es $DB_3 = \{R(A, K_2), R(B, K_2), R(C, K_2)\}$. Estas tres instancias respetan la restricción, pero hay una importante diferencia entre DB_2 y las otras dos: para obtener la instancia DB_1, le hemos sacado a DB más elementos de lo necesario para obtener una instancia que satisfaga la restricción, pues si no hubiésemos eliminado a $R(B, K_2)$, sino sólo a $R(A, K_2)$ y a $R(A, K_3)$, la instancia resultante satisfaría la restricción; en DB_3 no hemos eliminado "en exceso", pero en cambio hemos agregado un elemento, $R(C, K_2)$, que era innecesario agregar. En cambio, DB_2 se ha obtenido efectuando un mínimo de cambios, modificando sólo lo indispensable para cumplir con la restricción. DB_2 resulta ser una reparación de DB (según la definición que hemos dado antes), mientras que DB_1 y DB_3 no lo son. Esta instancia DB resulta tener exactamente tres reparaciones, que son la mencionada DB_2, la instancia $\{R(A, K_2), R(B, K_2)\}$, y la instancia $\{R(A, K_3), R(B, K_2)\}$. Cualquier otra instancia de base de datos que se "fabrique" a partir de DB y que cumpla con la restricción funcional, "tendrá más cambios de los necesarios". La contraparte formal de estas observaciones está dada por la definición de reparación: así, por ejemplo, las tres instancias que hemos afirmado son las únicas reparaciones de DB, resultan ser las únicas instancias de base de datos que son $\leq_{DB}$-minimales en la clase de las instancias que satisfacen esta restricción. [NOTA: Concordantemente con los comentarios anteriores, cuando una instancia de base de datos DB respeta un determinado conjunto de restricciones de integridad, su única reparación resulta ser ella misma. Mencionaremos también que si un conjunto de restricciones no es contradictorio, que es el caso que normalmente interesa, toda instancia de base de datos tiene reparaciones.]

Una vez definido el concepto de reparación de una instancia de base de datos, los autores de [referencia 10] proponen que las afirmaciones que se acepten como "correctas" en relación a una instancia de base de datos DB inconsistente con determinadas restricciones de integridad, sean exactamente aquellas que se satisfacen en cada reparación de DB. Los comentarios efectuados en el párrafo anterior darán al lector, seguramente, una idea de por qué esta propuesta tiene mucho sentido. Nosotros adoptaremos este criterio para plantear con rigor el problema que abordaremos en el presente capítulo. A continuación efectuaremos un primer planteamiento, de manera intuitiva, y posteriormente, tras introducir algunos conceptos necesarios, lo formalizaremos.

b) Un primer planteamiento del problema

De todas las posibles instancias de base de datos relacionales y todas las posibles restricciones de integridad, escogeremos algunas en específico, y estudiaremos detenidamente estos casos en particular, mediante la aplicación del sistema deductivo **BLO-OC**. Posteriormente, en la sección (g) del presente capítulo, esbozaremos algunas posibles generalizaciones de nuestros resultados.

Para comenzar, consideraremos el caso de las instancias de base de datos relacionales que se construyen con una sola relación, la que supondremos binaria. Por lo tanto, estas instancias de base de datos constarán solamente de afirmaciones de tipo $R(a, b)$.

En cuanto a las restricciones de integridad, consideraremos la denominada "**Restricción Funcional**", que, como su nombre indica, exige que la relación R sea una función. Se trata de la restricción que hemos mencionado en las secciones anteriores: $\forall x \forall y \forall z(\neg R(x, y) \vee \neg R(x, z) \vee y = z)$. Así, en el ejemplo que hemos comentado, dado un número de cédula, la base de datos no debería asignarle más de un nombre.

Con ello, un primer planteamiento del problema que abordaremos, enunciado informalmente, es el siguiente: dada cualquier instancia de base de datos DB formada solamente por afirmaciones de tipo $R(a, b)$ a la que se le asigna la restricción funcional, pero que probablemente es inconsistente con tal restricción, mostraremos que **BLO-OC** es capaz de reflejar su funcionamiento, en el sentido de que una fórmula escrita en lenguaje clásico se podrá deducir en **BLO-OC**, a partir de un conjunto de premisas que se corresponderá de determinada manera con DB, si y sólo si es satisfecha en cada reparación de la instancia de base de datos.

En realidad, este problema será abordado en dos "versiones". Pero dejaremos para más adelante las explicaciones al respecto. Por el momento, debemos presentar algunos elementos necesarios para enunciar con formalidad el problema; por ejemplo, debemos definir el sentido preciso de afirmar que las reparaciones satisfacen determinada fórmula proposicional: veremos que esta noción necesariamente deberá diferir de la correspondiente para fórmulas de primer orden, que es la que se usa habitualmente en computación teórica, según ya se ha mencionado.

c) Simbolizar la base de datos

Para hablar de una instancia de base de datos relacional en el lenguaje proposicional de bloqueo, queremos representar en él a las afirmaciones de tipo $R(c_{\alpha_1}, \ldots, c_{\alpha_t})$ que integran la instancia. Todo lo explicado o comentado en la sección (a) de este capítulo, por ejemplo la definición habitual de $DB \models \phi$, se refería a lógica de primer orden, la que cuenta con símbolos para relaciones y constantes, de modo que una afirmación de tipo $R(c_{\alpha_1}, \ldots, c_{\alpha_t})$ puede ser construida directamente. En cambio, en un lenguaje proposicional es imposible construir afirmaciones relacionales: cada letra proposicional P_i debe nombrar a una afirmación relacional completa; visto de otro modo, la lógica proposicional presupone que se dispone de afirmaciones preexistentes, pues no cuenta con las herramientas necesarias para construirlas sin partir de algunas dadas. Es entonces necesario buscar un modo de representar las afirmaciones $R(c_{\alpha_1}, \ldots, c_{\alpha_t})$ de que se dispone, mediante letras proposicionales. Procederemos del modo siguiente:

Una instancia de base de datos relacional DB está constituida, como hemos dicho, por un conjunto finito de afirmaciones de forma $R(c_{\alpha_1}, c_{\alpha_2}, \ldots, c_{\alpha_t})$, donde los c_{α_i} se han tomado de un conjunto $\{c_1, c_2, c_3, \ldots\}$. Existe por lo tanto un conjunto finito de objetos $\{a_1, a_2, \ldots, a_n\}$, subconjunto de $\{c_1, c_2, \ldots\}$, tal que para cada $R(x_1, x_2, \ldots, x_t)$ perteneciente a DB, los objetos $x_1, x_2, \ldots, x_t$ están todos en $\{a_1, a_2, \ldots, a_n\}$, y tal que todo a_i del conjunto figura en alguna relación de la instancia de base de datos. Este conjunto $\{a_1, a_2, \ldots, a_n\}$ se denomina el **dominio activo de DB**. En otras palabras, el dominio activo de DB consta de exactamente aquellos objetos del conjunto $\{c_1, c_2, c_3, \ldots\}$ que figuran en alguna afirmación $R(x_1, x_2, \ldots, x_t)$ perteneciente a DB.

[3.1] **DEFINICION:** Sea DB una instancia de base de datos relacional cuyo dominio activo consta de n elementos, y sea R una relación k-aria que se utiliza en la instancia DB; llamemos $A(R)$ al conjunto de las n^k afirmaciones de tipo $R(c_{\alpha_1}, c_{\alpha_2}, \ldots, c_{\alpha_k})$ que se pueden construir con los elementos de tal dominio activo para la relación R. Una función que vaya desde el conjunto de letras proposicionales hacia $A(R)$ y que sea epiyectiva, se puede restringir para obtener una biyección $\mathrm{N} : \mathrm{L} \longrightarrow A(R)$ [esta L es, entonces, un conjunto finito de letras proposicionales]. Diremos que una tal biyección N es una **denominación** para DB, o que **da nombre** a los elementos de $A(R)$; y la letra N (de "nombrar") siempre denotará biyecciones de denominación.

□□

[NOTA: consideré llamar a esta función N "nombramiento" o "nominación", pero en castellano usamos estas palabras predominantemente en otros sentidos.]

En el caso de una instancia DB que contiene sólo afirmaciones de tipo $R(a,b)$ para cierta R relación binaria (que, como dijimos, es el caso que por el momento nos interesa), si el dominio activo de DB consta de n objetos, entonces es posible formar n^2 afirmaciones de forma $R(a,b)$ con los elementos de tal dominio activo. Teniendo una función de denominación, habremos dado a cada una de dichas n^2 afirmaciones $R(a,b)$, un nombre P_i en el conjunto de letras proposicionales. Podemos ahora representar instancias de base de datos usando el lenguaje proposicional, de manera natural. Si DB es una instancia que tiene dominio activo $\{a_1, a_2, ..., a_n\}$, ordenemos matricialmente (sólo para visualizar mejor) las n^2 relaciones binarias que con dicho dominio activo podemos formar:

$$\begin{pmatrix} R(a_1,a_1) & R(a_1,a_2) & \dots & R(a_1,a_n) \\ R(a_2,a_1) & R(a_2,a_2) & \dots & R(a_2,a_n) \\ \vdots & \vdots & & \vdots \\ R(a_n,a_1) & R(a_n,a_2) & \dots & R(a_n,a_n) \end{pmatrix}$$

Algunas de estas afirmaciones están en DB, y otras no. Por ejemplo, consideremos la instancia $DB = \{R(a_1,a_1), R(a_2,a_1), R(a_2,a_3), R(a_3,a_2), R(a_3,a_3)\}$; en este caso, el dominio activo es $\{a_1, a_2, a_3\}$, y las relaciones que con él se pueden formar son

$$\begin{pmatrix} R(a_1,a_1) & R(a_1,a_2) & R(a_1,a_3) \\ R(a_2,a_1) & R(a_2,a_2) & R(a_2,a_3) \\ R(a_3,a_1) & R(a_3,a_2) & R(a_3,a_3) \end{pmatrix}$$

de modo que la instancia DB se podría visualizar así:

$$\begin{pmatrix} R(a_1,a_1) & & \\ R(a_2,a_1) & & R(a_2,a_3) \\ & R(a_3,a_2) & R(a_3,a_3) \end{pmatrix}$$

Tomemos una función denominación $\mathbf{N}$, renombrando su dominio para recordar mejor, del modo siguiente: si $\mathbf{N}(P_i) = R(a_b, a_c)$, a la letra proposicional P_i rebauticémosla como P_{bc} , de manera que $\mathbf{N}(P_{bc}) = R(a_b, a_c)$. Entonces las relaciones que pueden formarse con los elementos del dominio activo podemos representarlas así:

$$\begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix}$$

Y un modo natural de representar a DB sería

$$\begin{pmatrix} P_{11} & \neg P_{12} & \neg P_{13} \\ P_{21} & \neg P_{22} & P_{23} \\ \neg P_{31} & P_{32} & P_{33} \end{pmatrix}$$

que podemos interpretar como: según DB, la afirmación P_{11} [que representa a $R(a_1, a_1)$] es verdadera, la afirmación P_{12} [que representa a $R(a_1, a_2)$] es falsa, etc. Recuérdese que hemos ordenado matricialmente sólo para visualizar mejor: la teoría que estamos proponiendo para representar a la instancia de base de datos del ejemplo, es simplemente $\left\{P_{11}\,,\neg P_{12}\,,\neg P_{13}\,,P_{21}\,,\neg P_{22}\,,P_{23}\,,\neg P_{31}\,,P_{32}\,,P_{33}\right\}$.

En resumen, las funciones de denominación nos permiten hablar de las afirmaciones pertenecientes a instancias de base de datos, utilizando un lenguaje proposicional (en lugar de uno de primer orden).

OBSERVACION: Para "hablar acerca de una instancia de base de datos" en un lenguaje proposicional, es indispensable partir con una función denominación dada, u otra herramienta que cumpla una labor similar, pues las letras proposicionales "son indistinguibles entre sí" para el sistema deductivo (Si se usara un sistema deductivo que de algún modo las distingue unas de otras, ese mecanismo cumpliría, a su manera, el papel de la función denominación).

OBSERVACION: Partiendo de la colección infinita numerable de objetos $c_1, c_2, c_3, \ldots$ pueden construirse numerables afirmaciones $R(c_{\alpha_1}, c_{\alpha_2}, \ldots, c_{\alpha_t})$, y por lo tanto con las infinitas numerables letras proposicionales podríamos nombrarlas a todas. Sin embargo, definir las funciones de denominación del modo en que se ha hecho, dando nombre con ellas solamente a las relaciones formadas con elementos del dominio activo, tendrá gran utilidad. Entre varias otras cosas, permite por ejemplo evitar, bajo ciertas condiciones, conjuntos infinitos de premisas.

Definiremos ahora la noción de satisfacción, en una instancia de base de datos, de fórmulas del lenguaje proposicional de bloqueo construidas sin utilizar el símbolo de confirmación (fórmulas, por lo tanto, escritas en el lenguaje de CPC). Nos basaremos completamente en la definición ("clásica") para fórmulas de primer orden, aludida en la sección (a) del presente capítulo, pero con la importante diferencia de que en el lenguaje proposicional sólo tiene sentido hacerlo fijando previamente una función denominación, pues, como hemos observado, sólo entonces pueden identificarse afirmaciones pertenecientes a una instancia de base de datos con letras proposicionales:

[3.2] DEFINICION (Satisfacción de fórmulas clásicas): Dada una instancia de base de datos cualquiera D y una función denominación N para D, la afirmación "D satisface una fórmula ξ dada (clásica) con la denominación N" será denotada $D \models \xi[\mathrm{N}]$, y la definiremos recursivamente del modo siguiente:

$D \models P_i[\mathrm{N}]$ es cierto si y sólo si P_i está en el dominio de N y además $\mathrm{N}(P_i) \in D$
$D \models (\neg\phi)[\mathrm{N}]$ es cierto si y sólo si no se cumple $D \models \phi[\mathrm{N}]$
$D \models (\phi \wedge \psi)[\mathrm{N}]$ es cierto si y sólo si $D \models \phi[\mathrm{N}]$ y $D \models \psi[\mathrm{N}]$
$D \models (\phi \vee \psi)[\mathrm{N}]$ es cierto si y sólo si $D \models \phi[\mathrm{N}]$ y/o $D \models \psi[\mathrm{N}]$
$D \models (\phi \rightarrow \psi)[\mathrm{N}]$ es cierto si y sólo si $D \models \psi[\mathrm{N}]$ y/o no se cumple $D \models \phi[\mathrm{N}]$

Obsérvese que no se asigna significado a la expresión $D \models (\phi^o)[\mathrm{N}]$, de modo que el concepto de satisfacción en una instancia de base de datos sólo está definido para fórmulas escritas en el lenguaje de CPC (sin el símbolo ^{o}).

□□

Notar que según la definición anterior, si P_t es una letra proposicional que no pertenece al dominio de determinada denominación N para DB, entonces $DB \models (\neg P_t)[\mathrm{N}]$. La idea es que una tal P_t , al no pertenecer al dominio de N,

debería representar a una afirmación $R(k,l)$ que "se sale" del dominio activo, y de ser así, como en ninguna reparación respecto a la restricción funcional figura $R(k,l)$ por no pertenecer "k" y/o "l" al dominio activo, tiene sentido tener $DB \models (\neg P_i)[\mathrm{N}]$.

[3.3] NOTACION: Dada una instancia de base de datos DB , y dado un conjunto $\mathbf{I}$ de restricciones de integridad, el conjunto de las reparaciones de DB con respecto a $\mathbf{I}$ será denotado $\mathbf{K}(DB)$.

□□

En el último párrafo de la sección (a) del presente capítulo se mencionó que los autores de [ref. 10] proponen que las afirmaciones que se acepten como "correctas" en relación a una instancia de base de datos DB inconsistente con determinadas restricciones de integridad, sean exactamente aquellas que se satisfacen en cada reparación de DB. Establecen esta idea definiéndola formalmente, utilizando lógica de primer orden, como es habitual en computación teórica. Basándonos directamente en esta idea, definiremos su equivalente para lógica proposicional, utilizando los conceptos y la simbología que hemos propuesto:

[3.4] DEFINICION: Siendo $\mathbf{K}(DB)$ el conjunto de las reparaciones de la instancia de base de datos DB respecto a un conjunto dado de restricciones de integridad, sea N una función denominación para DB , y sea ϕ una fórmula en que no figura el símbolo ^{o} . Entonces, la afirmación "$\mathbf{K}(DB)$ satisface ϕ con N" será simbolizada $\mathbf{K}(DB) \models \phi[\mathrm{N}]$, y significará que para cada $D \in \mathbf{K}(DB)$ se cumple $D \models \phi[\mathrm{N}]$.

□□

Así, la afirmación $\mathbf{K}(DB) \models P_i[\mathrm{N}]$ significa que $\mathrm{N}(P_i)$ pertenece a cada elemento de $\mathbf{K}(DB)$; la afirmación $\mathbf{K}(DB) \models (\neg\phi)[\mathrm{N}]$ significa que en cada instancia perteneciente a $\mathbf{K}(DB)$ la fórmula ϕ es falsa con N ; etc. Nótese que afirmar que se cumple $\mathbf{K}(DB) \models (\neg\phi)[\mathrm{N}]$, no es lo mismo que negar que se cumple $\mathbf{K}(DB) \models \phi[\mathrm{N}]$.

A continuación asignaremos un conjunto de fórmulas a cada instancia de base de datos del tipo que nos ocupa, tras lo cual veremos un ejemplo que ilustre esta construcción:

[3.5] DEFINICION (Conjunto $\Sigma_{\mathrm{N}}(DB)$): Sea DB una instancia de base de datos, formada sólo por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R ; y sea N una función denominación para DB. Denotaremos $\Sigma_{\mathrm{N}}(DB)$ al conjunto formado por las siguientes fórmulas:

(a) [*Por inmersión de DB*] Para cada $\mathrm{N}(P_i) \in DB$, incluimos la letra proposicional P_i ; y para cada $\mathrm{N}(P_i) \notin DB$, incluimos $\neg P_i$.

(b) [*Por restricción funcional*] Para cada "a" elemento del dominio activo "que no respeta la restricción", es decir si $R(a,b_j) \in DB$, $R(a,b_k) \in DB$ y $b_j \neq b_k$, incluimos la negación del nombre (bajo N) de cada afirmación $R(a,b_t) \in DB$, es decir que para cada $R(a,b_t) \in DB$ incluimos la fórmula $(\neg P_i)$ tal que $\mathrm{N}(P_i) = R(a,b_t)$. Si "$a$" es un elemento del dominio activo "que sí respeta la restricción", es decir si $R(a,c) \in DB$ y $c_k \neq c \Rightarrow R(a,c_k) \notin DB$, incluimos la doble negación $(\neg\neg P_i)$ de la letra proposicional tal que $\mathrm{N}(P_i) = R(a,c)$. Finalmente, para cada $R(a,d)$ constructible con elementos del dominio activo que no figure en DB, incluimos la triple negación $(\neg\neg\neg P_i)$ para la letra proposicional tal que $P_i = R(a,d)$.

(c) [*Por "minimalidad" de las reparaciones*] Para cada "a" elemento del dominio activo "que no respeta la restricción", es decir si $R(a,b_j) \in DB$, $R(a,b_k) \in DB$ y $b_j \neq b_k$, llamemos $R(a,g_1), R(a,g_2), \ldots, R(a,g_r)$ a **todas** las afirmaciones que figuran en DB que tienen a "a" como primera componente; sea P_{γ_i} el elemento del dominio de la denominación tal que $\mathrm{N}(P_{\gamma_i}) = R(a,g_i)$; entonces incluimos en $\Sigma_{\mathrm{N}}(DB)$ la fórmula $(P_{\gamma_1} \vee P_{\gamma_2} \vee \cdots \vee P_{\gamma_r})$ y su doble negación $\neg\neg(P_{\gamma_1} \vee P_{\gamma_2} \vee \cdots \vee P_{\gamma_r})$.

(b') [*Por restricción funcional*] Para cada "a" elemento del dominio activo "que no respeta la restricción", es decir si $R(a,b_j) \in DB$, $R(a,b_k) \in DB$ y $b_j \neq b_k$, llamemos $R(a,g_1), R(a,g_2), \ldots, R(a,g_r)$ a **todas** las afirmaciones que figuran en DB que tienen a "a" como primera componente; sea P_{γ_i} el elemento del dominio de la denominación tal que $\mathrm{N}(P_{\gamma_i}) = R(a,g_i)$; con los r elementos g_i que estamos considerando, pueden formarse r grupos integrados por $(r-1)$ de ellos; estos grupos son los siguientes:

$$\{\ g_2\ ,\ g_3\ ,\ g_4\ ,\ \cdots\ ,\ g_{r-2}\ ,\ g_{r-1}\ ,\ g_r\}$$

$$\{g_1\ ,\quad g_3\ ,\ g_4\ ,\ \cdots\ ,\ g_{r-2}\ ,\ g_{r-1}\ ,\ g_r\}$$

$$\{g_1\ ,\ g_2\ ,\quad g_4\ ,\ \cdots\ ,\ g_{r-2}\ ,\ g_{r-1}\ ,\ g_r\}$$

$$\vdots$$

$$\{g_1\ ,\ g_2\ ,\ g_3\ ,\ g_4\ ,\ \cdots\ ,\ g_{r-2}\ ,\quad g_r\}$$

$$\{g_1\ ,\ g_2\ ,\ g_3\ ,\ g_4\ ,\ \cdots\ ,\ g_{r-2}\ ,\ g_{r-1}\ \}$$

Para cada uno de estos r grupos, formamos la conjunción de las negaciones de los nombres de las correspondientes afirmaciones $R(a, g_i)$; e incluimos en $\mathbf{\Sigma_N}(DB)$ la disyunción de tales conjunciones, es decir la fórmula:

$$\left(\neg P_{\gamma_2} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \cdots$$
$$\cdots \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_2} \wedge \cdots \wedge \neg P_{\gamma_{r-2}} \wedge \neg P_{\gamma_{r-1}}\right)$$

Incluimos también la doble negación de dicha fórmula:

$$\neg\neg\Big[\left(\neg P_{\gamma_2} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \cdots$$
$$\cdots \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_2} \wedge \cdots \wedge \neg P_{\gamma_{r-2}} \wedge \neg P_{\gamma_{r-1}}\right)\Big]$$

□□

[3.6] EJEMPLO: Sea DB la siguiente instancia de base de datos : $DB = \{\ R(a,a)\ ,\ R(a,b)\ ,\ R(a,c)\ ,\ R(b,b)\ ,\ R(c,a)\ ,\ R(c,b)\ \}$; su dominio activo es $\{\ a\ ,\ b\ ,\ c\ \}$; tomemos para DB la siguiente denominación $\mathbf{N}$:

$$\begin{array}{lll} \mathbf{N}(P_1) = R(a,a) & , \quad \mathbf{N}(P_2) = R(a,b) & , \quad \mathbf{N}(P_3) = R(a,c) \ , \\ \mathbf{N}(P_4) = R(b,a) & , \quad \mathbf{N}(P_5) = R(b,b) & , \quad \mathbf{N}(P_6) = R(b,c) \ , \\ \mathbf{N}(P_7) = R(c,a) & , \quad \mathbf{N}(P_8) = R(c,b) & , \quad \mathbf{N}(P_9) = R(c,c) \ . \end{array}$$

Entonces el conjunto $\mathbf{\Sigma_N}(DB)$ es la unión de los cuatro conjuntos de fórmulas siguientes, correspondientes a las respectivas partes de [3.5] :

(a) $\{\ P_1\ ,\ P_2\ ,\ P_3\ ,\ \neg P_4\ ,\ P_5\ ,\ \neg P_6\ ,\ P_7\ ,\ P_8\ ,\ \neg P_9\ \}$

(b) $\{\ \neg P_1\ ,\ \neg P_2\ ,\ \neg P_3\ ,\ \neg\neg\neg P_4\ ,\ \neg\neg P_5\ ,\ \neg\neg\neg P_6\ ,\ \neg P_7\ ,\ \neg P_8\ ,\ \neg\neg\neg P_9\ \}$

(c) $\{\ (P_1 \vee P_2 \vee P_3)\ \ ,\ \ \neg\neg(P_1 \vee P_2 \vee P_3)\ \ ,\ \ (P_7 \vee P_8)\ \ ,\ \ \neg\neg(P_7 \vee P_8)\ \}$

(b') $\{\ (\neg P_2 \wedge \neg P_3) \vee (\neg P_1 \wedge \neg P_3) \vee (\neg P_1 \wedge \neg P_2)\ \ ,\ \ (\neg P_8 \vee \neg P_7)\ \ ,$

$\neg\neg\Big[(\neg P_2 \wedge \neg P_3) \vee (\neg P_1 \wedge \neg P_3) \vee (\neg P_1 \wedge \neg P_2)\Big]\ \ ,\ \ \neg\neg\Big[(\neg P_8 \vee \neg P_7)\Big]\ \ \}$

□□

Además del conjunto $\Sigma_{\mathbf{N}}(DB)$ que hemos definido, utilizaremos la siguiente extensión de él:

[3.7] DEFINICION (Conjunto $\Sigma^{\infty}_{\mathbf{N}}(DB)$): Sea DB una instancia de base de datos, formada sólo por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R ; y sea $\mathbf{N}$ una función denominación para DB. Denotaremos $\Sigma^{\infty}_{\mathbf{N}}(DB)$ al conjunto resultante de agregar a $\Sigma_{\mathbf{N}}(DB)$, definido en [3.5], las negaciones de todas las letras proposicionales que no pertenecen al dominio de la denominación $\mathbf{N}$, junto a sus triples negaciones; es decir:

$$\Sigma^{\infty}_{\mathbf{N}}(DB) \ = \ \Sigma_{\mathbf{N}}(DB) \cup \{\ \neg P_i\ :\ P_i \notin \mathit{Dom}(\mathbf{N})\ \} \cup \{\ \neg\neg\neg P_i\ :\ P_i \notin \mathit{Dom}(\mathbf{N})\ \}$$

□□

[3.8] EJEMPLO: Tomemos las mismas DB y $\mathbf{N}$ del ejemplo [3.6] . El dominio de esta denominación es $\{\ P_1\ ,\ P_2\ ,\ P_3\ ,\ P_4\ ,\ P_5\ ,\ P_6\ ,\ P_7\ ,\ P_8\ ,\ P_9\ \}$. Entonces el conjunto $\Sigma^{\infty}_{\mathbf{N}}(DB)$ está formado por las fórmulas de $\Sigma_{\mathbf{N}}(DB)$ mostradas en [3.6], más las fórmulas pertenecientes al conjunto infinito siguiente:

$$\{\ \neg P_i\ ,\ \neg\neg\neg P_i\ \}_{i>9} \ = \ \{\ \neg P_{10}\ ,\ \neg\neg\neg P_{10}\ ,\ \neg P_{11}\ ,\ \neg\neg\neg P_{11}\ ,\ \neg P_{12}\ ,\ \neg\neg\neg P_{12}\ ,\ \ldots\ \}$$

□□

d) Planteamiento del problema

Disponemos ahora de los conceptos adecuados para enunciar con precisión el objetivo principal del presente capítulo, que en la sección (b) hemos esbozado intuitivamente. Dicho objetivo, como se mencionó someramente, tiene dos "versiones". La primera es la siguiente:

Sea DB una instancia de base de datos, a la que se asigna la restricción funcional, formada sólo por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R ; sea N una función de denominación para DB ; y sea ψ cualquier fórmula del lenguaje proposicional de bloqueo básico que haya sido construida sin utilizar el símbolo de confirmación ^{o} , y que cumpla con cierta restricción, que especificaremos más adelante (esta restricción, hablando intuitivamente, dirá que ψ "se refiere a la instancia DB"). Bajo tales condiciones, demostraremos que se cumple que

$$\Sigma_{\mathrm{N}}(DB) \vdash \psi \quad \Longleftrightarrow \quad \mathbf{K}(DB) \models \psi[\mathrm{N}]$$

De la doble implicación anterior, la de izquierda a derecha corresponde a la noción usual de corrección, en el sentido de que indica que toda fórmula "referente a DB" que pueda deducirse de $\Sigma_{\mathrm{N}}(DB)$ en **BLO-OC**, se satisface en todas la reparaciones de DB ; y la de derecha a izquierda corresponde a la completud, en el sentido de que toda fórmula "referente a DB" que se satisfaga en cada reparación de DB , es demostrable en **BLO-OC** a partir de $\Sigma_{\mathrm{N}}(DB)$ [en realidad, esta restricción de "referirse a DB" será necesaria solamente para esta segunda implicación].

La segunda versión es la siguiente:

Sea DB una instancia de base de datos, a la que se asigna la restricción funcional, formada sólo por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R ; sea N una función de denominación para DB ; y sea ψ cualquier fórmula del lenguaje proposicional de bloqueo básico que haya sido construida sin utilizar el símbolo de confirmación ^{o} . Bajo tales condiciones, demostraremos que se cumple que

$$\Sigma^{\infty}_{\mathrm{N}}(DB) \vdash \psi \quad \Longleftrightarrow \quad \mathbf{K}(DB) \models \psi[\mathrm{N}]$$

Obsérvese que en este caso se abarcan todas las fórmulas escritas en el lenguaje de CPC, mientras que en el primero la restricción mencionada dejará afuera a algunas de ellas. Pero para lograr esto, en el segundo caso tenemos que usar el conjunto $\Sigma^{\infty}_{\mathbf{N}}(DB)$, que es infinito, mientras que en el primer caso basta con $\Sigma_{\mathbf{N}}(DB)$, que es finito.

Comenzaremos con la dirección correspondiente a la corrección; en ambas versiones lograremos establecer rápidamente esta implicación, gracias al trabajo que hemos realizado en el capítulo 2.

e) Corrección

Para demostrar que $\Sigma_{\mathbf{N}}(DB) \vdash \psi \;\Rightarrow\; \mathbf{K}(DB) \models \psi[\mathbf{N}]$, empezaremos con el resultado siguiente, que es en realidad un corolario de [2.18] :

[3.9] LEMA: *Sea DB una instancia de base de datos construida con una única relación binaria, sea* $\mathbf{N}$ *una denominación para DB, y sea ψ una fórmula en la que no aparece el símbolo de confirmación. Si* $\Sigma_{\mathbf{N}}(DB) \vdash \psi$ *, entonces* $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\} \vdash_{cpc} \psi$.

Demostración: Este Lema se deduce directamente del Corolario [2.18] , ya que por definición el conjunto $\Sigma_{\mathbf{N}}(DB)$ ha sido construido sin el símbolo de confirmación, de manera que cumple con la hipótesis que en [2.18] se le pide al conjunto Σ (según lo convenido en [2.7]).

□□

[3.10] LEMA: *Sea DB una instancia de base de datos construida con una única relación binaria, sea* $\mathbf{N}$ *una denominación para DB, y sea ψ una fórmula en la que no aparece el símbolo* $^{\circ}$. *Si* $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\} \vdash_{cpc} \psi$ *entonces* $\mathbf{K}(DB) \models \psi[\mathbf{N}]$.

Demostración: Antes de empezar, y aunque no se requiere hacerlo, indicaremos que la expresión $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\} \vdash_{cpc} \psi$ está siempre bien definida, ya que el símbolo $^\circ$ no figura en ψ [por hipótesis] ni en las fórmulas del conjunto $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ [por construcción de $\Sigma_{\mathbf{N}}(DB)$].

Revisando la construcción de $\Sigma_{\mathbf{N}}(DB)$ que hemos establecido en [3.5], vemos que el conjunto $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ consta de las fórmulas siguientes:

- Fórmula $\neg P_i$ cada vez que la afirmación $\mathbf{N}(P_i)$ no está en DB , según partes (a) y (b) de [3.5] .

- Fórmula P_i cada vez que la afirmación $\mathbf{N}(P_i)$ pertenece a DB y "respeta la restricción", según partes (a) y (b) de [3.5] .

- Toda fórmula descrita en la parte (c) de [3.5] que no es doble negación [es decir, todas las fórmulas que, cuando DB contiene varias afirmaciones con igual primera componente $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$, dicen "por lo menos una de ellas $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$ es cierta"].

- Toda fórmula descrita en la parte (b') de [3.5] que no es doble negación [es decir, todas las fórmulas que, cuando DB contiene varias afirmaciones con igual primera componente $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$, dicen "como máximo una de ellas $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$ es cierta"].

Vemos directamente que todas estas fórmulas son satisfechas con $\mathbf{N}$ en cada reparación de DB : en efecto, si una afirmación $R(a, b)$ no aparece en DB, tampoco aparece en las reparaciones; si una afirmación aparece en DB y "respeta la restricción", entonces aparece en cada reparación; y si una afirmación aparece en DB y "no respeta la restricción", entonces, de todas las afirmaciones con igual primera componente $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$, en cada reparación aparece exactamente una de ellas, es decir, al menos una y como máximo una. En resumen, cada una de las fórmulas de $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ es satisfecha con $\mathbf{N}$ en $\mathbf{K}(DB)$, es decir que si ψ es cualquiera de estas fórmulas, entonces $\mathbf{K}(DB) \models \psi[\mathbf{N}]$

Utilizando este hecho, demostraremos el presente lema por inducción sobre la longitud de las demostraciones de fórmulas en CPC a partir del conjunto de premisas $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$:

Demostraciones de longitud 1: Si en CPC se deduce ψ a partir de $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$ en un sólo paso, entonces o bien ψ es un axioma de CPC , o bien $\psi \in \{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$. En el primer caso tenemos directamente que $\mathbf{K}(DB) \models \psi[\mathrm{N}]$ (si se toma cualquier axiomatización de CPC, para cada axioma se puede comprobar directamente, con la definición de satisfacción [3.2] , que es satisfecho en cualquier instancia de base de datos. De hecho, toda tautología clásica resulta ser satisfecha en cualquier instancia y con cualquier denominación. Ver comentarios en el apéndice histórico-filosófico); y en el segundo caso se aplican los comentarios formulados en la presente demostración antes de comenzar la inducción, con lo que igualmente tenemos $\mathbf{K}(DB) \models \psi[\mathrm{N}]$.

Hipótesis inductiva: Suponemos que para toda fórmula ψ que puede ser demostrada en CPC a partir de $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$ en menos de N pasos, se cumple que $\mathbf{K}(DB) \models \psi[\mathrm{N}]$.

Paso inductivo: Sea ψ una fórmula que ha sido demostrada en CPC a partir de $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$ en N pasos. Si ψ es un axioma de CPC o está en el conjunto de premisas $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$, ya hemos visto que se cumple $\mathbf{K}(DB) \models \psi[\mathrm{N}]$. Y si ψ fue obtenida con *Modus Ponens*, en la deducción formal habían aparecido con anterioridad fórmulas $A \rightarrow \psi$ y A , las que, por lo tanto, se deducen en menos de N pasos, y por consiguiente les podemos aplicar la hipótesis inductiva. Tenemos entonces que $\mathbf{K}(DB) \models A[\mathrm{N}]$ y $\mathbf{K}(DB) \models (A \rightarrow \psi)[\mathrm{N}]$. Revisando las definiciones de satisfacción [3.2] y [3.4] , vemos que de estos dos hechos se sigue que $\mathbf{K}(DB) \models \psi[\mathrm{N}]$.

□□

[3.11] COROLARIO (Corrección para $\Sigma_{\mathrm{N}}(DB)$): *Sea DB una instancia de base de datos construida con una única relación binaria, sea* N *una denominación para DB, y sea ψ una fórmula en la que no aparece el símbolo $^{\circ}$. Si $\Sigma_{\mathrm{N}}(DB) \vdash \psi$, entonces $\mathbf{K}(DB) \models \psi[\mathrm{N}]$.*

Demostración: Lema [3.9] , más Lema [3.10] .

□□

Ahora demostraremos que $\Sigma_{\mathrm{N}}^{\infty}(DB) \vdash \psi \;\Rightarrow\; \mathbf{K}(DB) \models \psi[\mathrm{N}]$. El camino es totalmente análogo al de la versión para $\Sigma_{\mathrm{N}}(DB)$; en el caso del resultado [3.13] , la demostración es casi igual a la de [3.10] , sólo se agregan las consideraciones correspondientes a las fórmulas de $\Sigma_{\mathrm{N}}^{\infty}(DB)$ que no aparecen en $\Sigma_{\mathrm{N}}(DB)$, de modo que resumiremos la demostración, remitiendo al lector a la de [3.10] para consultar los detalles; el resultado [3.12] se demuestra exactamente igual que su "correspondiente" [3.9]:

[3.12] **LEMA:** *Sea DB una instancia de base de datos construida con una única relación binaria, sea* N *una denominación para DB, y sea ψ una fórmula en la que no aparece el símbolo de confirmación. Si $\Sigma_{\mathrm{N}}^{\infty}(DB) \vdash \psi$, entonces $\{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\} \vdash_{cpc} \psi$.*

Demostración: Este Lema se deduce directamente del Corolario [2.18] .

□□

[3.13] **LEMA:** *Sea DB una instancia de base de datos construida con una única relación binaria, sea* N *una denominación para DB, y sea ψ una fórmula en la que no aparece el símbolo ^{o} . Si $\{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\} \vdash_{cpc} \psi$ entonces $\mathbf{K}(DB) \models \psi[\mathrm{N}]$.*

Demostración: Revisando la construcción de $\Sigma_{\mathrm{N}}^{\infty}(DB)$ que hemos establecido en la definición [3.7] , vemos que $\{\, \xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) \;:\; \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) \,\}$ está formado por las fórmulas que, según hemos indicado en la demostración del Lema [3.10], conforman el conjunto $\{\, \xi \in \Sigma_{\mathrm{N}}(DB) \;:\; \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB) \,\}$, más las negaciones de todas las letras proposicionales que no pertenecen al dominio de la denominación N , es decir que $\{\, \xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) \;:\; \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) \,\} = \{\xi \in \Sigma_{\mathrm{N}}(DB) \;:\; \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\} \cup \{\, \neg P_i \;:\; P_i \notin Dom(\mathrm{N}) \,\}$. Ya vimos en la demostración de [3.10] que en cada reparación de DB se satisfacen con N todas las fórmulas de $\{\, \xi \in \Sigma_{\mathrm{N}}(DB) \;:\; \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB) \,\}$; y revisando la definición de satisfacción [3.2] , vemos que las fórmulas pertenecientes a $\{\, \neg P_i \;:\; P_i \notin Dom(\mathrm{N}) \,\}$ también son satisfechas con N por cada reparación de DB. Por lo tanto, todas las fórmulas de $\{\, \xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) \;:\; \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) \,\}$ son satisfechas con N en cada reparación de DB . Este hecho puede ahora

tomarse como punto de partida para repetir la inducción que se realizó en la demostración de [3.10] , tal cual fue efectuada, y con ello se demuestra que si $\{\xi \in \Sigma^{\infty}_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma^{\infty}_{\mathrm{N}}(DB)\} \vdash_{cpc} \psi$ entonces $\mathrm{K}(DB) \models \psi[\mathrm{N}]$.

□□

[3.14] COROLARIO (Corrección para $\Sigma^{\infty}_{\mathrm{N}}(DB)$): *Sea DB una instancia de base de datos construida con una única relación binaria, sea* N *una denominación para DB, y sea ψ una fórmula en la que no aparece el símbolo ^{o}. Si $\Sigma^{\infty}_{\mathrm{N}}(DB) \vdash \psi$, entonces $\mathrm{K}(DB) \models \psi[\mathrm{N}]$.*

Demostración: Lema [3.12] , más Lema [3.13] .

□□

f) Completud

Quisiéramos en principio demostrar que para cualesquiera DB y ψ del tipo que estamos considerando, se cumple $\mathrm{K}(DB) \models \psi[\mathrm{N}] \implies \Sigma_{\mathrm{N}}(DB) \vdash \psi$, o equivalentemente $\Sigma_{\mathrm{N}}(DB) \nvdash \psi \implies \mathrm{K}(DB) \not\models \psi[\mathrm{N}]$. Sin embargo, según ya hemos anunciado, esto se cumplirá sólo bajo una cierta restricción, que como veremos es bastante natural. Se trata de lo siguiente:

En el conjunto $\Sigma_{\mathrm{N}}(DB)$ figuran sólo las letras proposicionales que nombran, mediante N, a elementos del dominio activo. Sería entonces de esperar que al menos algunos hechos que dependan del resto de las letras proposicionales no puedan ser demostrados ni refutados a partir de la teoría $\Sigma_{\mathrm{N}}(DB)$. Y en efecto, por ejemplo si P_t es una letra proposicional que no pertenece al dominio de N , entonces ocurre que $\Sigma_{\mathrm{N}}(DB) \nvdash P_t$ y $\Sigma_{\mathrm{N}}(DB) \nvdash (\neg P_t)$; esto parece descartar la completud para $\Sigma_{\mathrm{N}}(DB)$, ya que las definiciones [3.4] y [3.2] (ver también comentario inmediatamente después de [3.2]) nos dicen que $\mathrm{K}(DB) \models (\neg P_t)[\mathrm{N}]$, a pesar de que $\Sigma_{\mathrm{N}}(DB) \nvdash (\neg P_t)$. Sin embargo, estableciendo una condición natural, tendremos para $\Sigma_{\mathrm{N}}(DB)$ una completud "restringida": intuitivamente hablando, ésta se verificará para las fórmulas "que se refieren" a la instancia de base de datos estudiada. Formalmente, la formulación exacta es la siguiente:

Sea DB una instancia de base de datos formada sólo por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R , a la que se asigna la restricción funcional; y sea $\mathbf{N}$ una función de denominación para DB. Renombrando las letras proposicionales si hace falta, podemos suponer que el dominio de $\mathbf{N}$ es $\{P_1, P_2, P_3, \ldots, P_k\}$. Entonces, para cada fórmula ψ del lenguaje proposicional de bloqueo en la que no aparece el símbolo de confirmación ^{o} y que haya sido construida usando solamente letras proposicionales pertenecientes a $\{P_1, P_2, P_3, \ldots, P_k\}$ [es decir $\psi = \psi(P_1, P_2, P_3, \ldots, P_k)$] , se cumple que

$$\Sigma_{\mathbf{N}}(DB) \vdash \psi \implies \mathbf{K}(DB) \models \psi[\mathbf{N}]$$

[OBSERVACION: Si una fórmula ψ contiene letras proposicionales que no pertenecen al dominio de la denominación, entonces alude a al menos alguna afirmación que no se construye con elementos del dominio activo de DB ; es en ese sentido que ψ "no se refiere" a DB. De ser así, es decir si en ψ figura una letra proposicional P_t no perteneciente al dominio de $\mathbf{N}$, entonces el que ψ no pueda deducirse en **BLO-OC** a partir de $\Sigma_{\mathbf{N}}(DB)$, no implica que no se satisfaga en $\mathbf{K}(DB)$; puede satisfacerse, o no. Por ejemplo, si P_t no pertenece al dominio de $\mathbf{N}$, entonces $\mathbf{K}(DB) \models (\neg P_t)[\mathbf{N}]$, a pesar de que $\Sigma_{\mathbf{N}}(DB) \nvdash (\neg P_t)$; en cambio, si por ejemplo P_2 es tal que $\mathbf{N}(P_2)$ pertenece a toda reparación de DB, entonces $\Sigma_{\mathbf{N}}(DB) \nvdash (\neg(P_2 \vee P_t))$ y $\mathbf{K}(DB) \not\models (\neg(P_2 \vee P_t))$] .

Tal es la versión de la completud para el conjunto finito $\Sigma_{\mathbf{N}}(DB)$. La otra versión es para el conjunto infinito $\Sigma^{\infty}_{\mathbf{N}}(DB)$, y en este caso, según hemos anunciado, la completud no requiere de la restricción. La formulación es:

Sea DB una instancia de base de datos formada sólo por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R , a la que se asigna la restricción funcional; y sea $\mathbf{N}$ una función de denominación para DB. Entonces, para cada fórmula ψ del lenguaje proposicional de bloqueo en la que no aparece el símbolo de confirmación ^{o} , se cumple que

$$\Sigma^{\infty}_{\mathbf{N}}(DB) \vdash \psi \implies \mathbf{K}(DB) \models \psi[\mathbf{N}]$$

Comenzaremos con la versión para $\Sigma_{\mathbf{N}}(DB)$:

[3.15] **LEMA :** *Si* $\psi = \overline{\psi}$ *(es decir, si en* ψ *no aparece* $^{\circ}$*) y si* $\Sigma_{\mathbf{N}}(DB) \vdash \psi$ *, entonces* $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\} \vdash_{cpc} \psi$.

Demostración: Este resultado se deduce directamente de [2.23] .

□□

Utilizaremos a continuación el concepto de función veritativa. En "Preliminares" (capítulo 1), junto con su definición, hemos recordado la convención que se adopta respecto a asignaciones veritativas y sus extensiones.

[3.16] **DEFINICION :** Sea σ una asignación veritativa para las fórmulas proposicionales de CPC. Por analogía con el uso habitual en matemáticas de la palabra "soporte", denominaremos **soporte de la asignación veritativa** σ al conjunto de letras proposicionales $\{ P_i : \sigma(P_i) = V \}$.

□□

Denotaremos con el símbolo $Sop(\sigma)$ al soporte de la función veritativa σ (esta es una de las abreviaturas que se usan para "soporte" en su acepción usual).

Con objeto de simplificar las demostraciones de los próximos resultados, adoptaremos la convención siguiente:

[3.17] **NOTACION :** Renombrando las letras proposicionales si hace falta, en lo que resta del presente capítulo asumiremos que cada función de denominación N que aparezca, tiene como dominio al conjunto $\{P_1, P_2, P_3, \ldots, P_k\}$. Y viceversa: cuando aparezca un conjunto $\{P_1, P_2, P_3, \ldots, P_k\}$, asumiremos que se trata del dominio de la función N de la que se esté hablando.

□□

El siguiente resultado es propio de CPC. Su demostración es sencilla, pero de todos modos la reproduciremos, porque es posible que a primera vista el lector no reconozca esta "versión para **BLO-OC**" . De él extraeremos un corolario que se usará en la demostración de la completud. Recordar que en este tercer capítulo seguimos aplicando lo convenido en [2.7] : el símbolo " Σ " representará siempre a un conjunto (arbitrario pero fijo) de fórmulas que no contienen el símbolo ^{o} .

LEMA: *Sea* ψ *una fórmula en la que no aparece el símbolo de confirmación, y supongamos que* $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash_{cpc} \psi$. *Sea* $\{P_{\alpha_1}, P_{\alpha_2}, P_{\alpha_3}, \ldots, P_{\alpha_n}\}$ *un conjunto que contiene a todas las letras proposicionales que figuran en las fórmulas de* $\Sigma \cup \{\psi\}$. *Entonces existe una asignación veritativa* σ *con soporte subconjunto de* $\{P_{\alpha_1}, P_{\alpha_2}, \ldots, P_{\alpha_n}\}$ *tal que* $\sigma(\psi) = F$ *y tal que* $\sigma(\xi) = V$ *para cada* $\xi \in \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$.

Demostración: Para cualquier conjunto K de fórmulas del cálculo clásico CPC, sabemos que se cumple $K \vdash_{cpc} \psi \iff K \models_{cpc} \psi$, es decir ψ se deduce de K si y sólo si ψ es consecuencia tautológica de K. Por lo tanto, la suposición de que $\{\xi \in \Sigma : \neg\neg\xi \in \Sigma\} \nvdash_{cpc} \psi$ nos dice que existe una asignación veritativa δ tal que $\delta(\psi) = F$ y tal que $\delta(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$.

Ahora bien, sabemos que en CPC el valor veritativo bajo δ de una fórmula dada, sólo depende del valor que δ le asigne a las letras proposicionales que figuran en dicha fórmula, de manera que si tenemos una fórmula ϕ de la que sabemos que $\phi = \phi(P_{\alpha_1}, P_{\alpha_2}, \ldots, P_{\alpha_n})$, podemos cambiar a voluntad el valor que δ le asigna a las letras que no están en $\{P_{\alpha_1}, P_{\alpha_2}, P_{\alpha_3}, \ldots, P_{\alpha_n}\}$, sin alterar el efecto de δ sobre ϕ ; en particular, podemos asignar el valor "F" a toda letra que no esté en el conjunto, sin alterar el efecto sobre ϕ . Por lo tanto, la existencia de la asignación δ tal que $\delta(\psi) = F$ y tal que $\delta(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$, implica que existe una asignación veritativa σ tal que $\sigma(\psi) = F$ y que $\sigma(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma : \neg\neg\xi \in \Sigma\}$, con soporte subconjunto de $\{P_{\alpha_1}, P_{\alpha_2}, \ldots, P_{\alpha_n}\}$.

□□

[3.18] COROLARIO: *Sea ψ una fórmula en la que no aparece el símbolo de confirmación, y supongamos que $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\} \vdash_{cpc} \psi$ y que $\psi = \psi(P_1, P_2, \ldots, P_k)$ (recordar lo convenido en [3.17]). Entonces existe una asignación veritativa σ con soporte contenido en $\{P_1, P_2, \ldots, P_k\}$, tal que $\sigma(\psi) = F$, y tal que $\sigma(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$.*

Demostración: Dada la manera en que se construye $\Sigma_{\mathrm{N}}(DB)$, el dominio de la denominación N contiene a todas las letras proposicionales que figuran en el conjunto $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$. Por lo tanto, se aplica el lema anterior.

□□

[3.19] DEFINICION : Sea DB una instancia de base de datos cualquiera, sea N una denominación para DB , y sea σ una asignación veritativa cuyo soporte es subconjunto del dominio de N , es decir $Sop(\sigma) \subseteq \{P_1, P_2, P_3, \ldots, P_k\}$. Utilizando solamente afirmaciones construidas con elementos del dominio activo de DB , es decir sólo con elementos de la imagen de N , definimos la instancia de base de datos $\mathbf{DB}_{\mathrm{N}}(\sigma)$ del modo siguiente:

$$\mathrm{N}(P_i) \in \mathbf{DB}_{\mathrm{N}}(\sigma) \qquad \Longleftrightarrow \qquad \sigma(P_i) = V$$

[Obsérvese que el hecho de que $Sop(\sigma) \subseteq \{P_1, P_2, P_3, \ldots, P_k\}$ asegura que $\mathbf{DB}_{\mathrm{N}}(\sigma)$ está bien definida, al garantizar que el valor veritativo de $\sigma(P_n)$ sólo puede ser V si $P_n \in \{P_1, P_2, P_3, \ldots, P_k\}$: de no ser así, la expresión $\mathrm{N}(P_n)$ no estaría definida, por no pertenecer P_n al dominio de N. Además, el hecho de que la instancia $\mathbf{DB}_{\mathrm{N}}(\sigma)$ se construya usando solamente elementos del dominio activo, asegura que ella es propiamente una instancia de base de datos, al garantizar que consta de una cantidad finita de afirmaciones.]

□□

[3.20] EJEMPLO : Consideremos la instancia de base de datos DB dada por $DB = \{R(a,b), R(b,a), R(b,b)\}$; su dominio activo es $\{a,b\}$; consideremos la denominación $\mathrm{N} : \{P_1, P_2, P_3, P_4\} \to \{R(a,a), R(a,b), R(b,a), R(b,b)\}$ dada por $\mathrm{N}(P_1) = R(a,a)$, $\mathrm{N}(P_2) = R(a,b)$, $\mathrm{N}(P_3) = R(b,a)$ y $\mathrm{N}(P_4) = R(b,b)$. Una asignación veritativa con soporte en $\{P_1, P_2, P_3, P_4\}$ puede ser la siguiente: $\sigma_1(P_2) = \sigma_1(P_3) = V$, $\sigma_1(P_i) = F$ para todo número natural i distinto de 2 y de 3 ; en este caso tenemos $\mathbf{DB}_{\mathrm{N}}(\sigma_1) = \{R(a,b), R(b,a)\}$. Otra asignación

con soporte contenido en el dominio de la denominación $\mathbf{N}$ es la definida por $\sigma_2(P_1) = \sigma_2(P_2) = \sigma_2(P_4) = V$, $\sigma_2(P_i) = F$ para todo número natural i distinto de 1, 2 y 4; en este caso tenemos $\mathbf{DB_N}(\sigma_2) = \{R(a,a), R(a,b), R(b,b)\}$. Obsérvese que la instancia de base de datos $\mathbf{DB_N}(\sigma_1)$ es una reparación de la instancia DB (con respecto a la restricción funcional), mientras que $\mathbf{DB_N}(\sigma_2)$ no lo es.

□□

[3.21] **LEMA:** *Sea* $\mathbf{N}$ *una denominación para una instancia* DB *, y sea* σ *una asignación veritativa con soporte subconjunto del dominio* $\{P_1, P_2, \ldots, P_k\}$ *de* $\mathbf{N}$ *. Entonces para toda fórmula* ψ *en la que no aparece el símbolo de confirmación, se tiene que* $\sigma(\psi) = V \iff \mathbf{DB_N}(\sigma) \models \psi[\mathbf{N}]$.

Demostración: Demostraremos el lema por inducción sobre la complejidad de las fórmulas que no contienen el símbolo de confirmación, es decir sobre el número de conectivos lógicos que figura en cada fórmula tal.

Fórmulas con 0 conectivos: Si ψ es una fórmula en la que no aparecen conectivos, entonces ella es una letra proposicional P_i ; tenemos entonces las siguientes equivalencias, que primero escribimos y a continuación justificamos:

$$\sigma(P_i) = V \;\Leftrightarrow\; P_i \in \{P_1, P_2, \ldots, P_k\} \text{ y } \mathbf{N}(P_i) \in \mathbf{DB_N}(\sigma) \;\Leftrightarrow\; \mathbf{DB_N}(\sigma) \models P_i[\mathbf{N}]$$

donde el primer "si y sólo si" se justifica porque $Sop(\sigma) \subseteq \{P_1, P_2, \ldots, P_k\}$ y por la definición de $\mathbf{DB_N}(\sigma)$ [3.19] , y el segundo se justifica por la definición de satisfacción en instancias de base de datos [3.2] .

Hipótesis inductiva: Suponemos que para toda fórmula en la que no aparece el símbolo de confirmación y que contiene menos de N conectivos lógicos, se cumple que $\sigma(\psi) = V \iff \mathbf{DB_N}(\sigma) \models \psi[\mathbf{N}]$

Paso inductivo: Sea ψ una fórmula en la que aparecen N conectivos lógicos, tal que ella no contiene el símbolo de confirmación. Hay cuatro casos posibles:

Primero, que ψ sea de forma $(\neg\phi)$; en este caso tenemos las siguientes equivalencias, que primero escribimos y a continuación justificamos:

$$\sigma(\psi) = V \quad \Leftrightarrow \quad \sigma(\phi) = F \quad \Leftrightarrow \quad \mathbf{DB_N}(\sigma) \not\models \phi[\mathrm{N}] \quad \Leftrightarrow \quad \mathbf{DB_N}(\sigma) \models (\neg\phi) = \psi\,[\mathrm{N}]$$

donde el primer "si y sólo si" se justifica por el modo en que las asignaciones veritativas se extienden a partir de su acción sobre las letras proposicionales; el siguiente, porque la igualdad $\psi = (\neg\phi)$ y la hipótesis de que en ψ no aparece el símbolo $^\circ$, implican que en ϕ tampoco puede aparecer dicho símbolo, y como los conectivos de ϕ son menos que N (tiene un conectivo menos que ψ, de modo que contiene $N-1$), cumple con la hipótesis inductiva; y el último, por la definición de satisfacción en instancias de base datos, establecida en [3.2].

Segundo, que ψ sea de forma $(\phi_1 \wedge \phi_2)$; en este caso tenemos las siguientes equivalencias, que tienen respectivamente similar justificación que las del caso anterior:

$$\sigma(\psi) = V \quad \Leftrightarrow \quad \sigma(\phi_1) = V \text{ y } \sigma(\phi_2) = V \quad \Leftrightarrow \quad \mathbf{DB_N}(\sigma) \models \phi_1[\mathrm{N}] \text{ y } \mathbf{DB_N}(\sigma) \models \phi_2[\mathrm{N}]$$

$$\Leftrightarrow \quad \mathbf{DB_N}(\sigma) \models (\phi_1 \wedge \phi_2)[\mathrm{N}]$$

Tercero, que ψ sea de forma $(\phi_1 \vee \phi_2)$; en este caso, con igual justificación que antes, tenemos:

$$\sigma(\psi) = F \quad \Leftrightarrow \quad \sigma(\phi_1) = F \text{ y } \sigma(\phi_2) = F \quad \Leftrightarrow \quad \mathbf{DB_N}(\sigma) \not\models \phi_1[\mathrm{N}] \text{ y } \mathbf{DB_N}(\sigma) \not\models \phi_2[\mathrm{N}]$$

$$\Leftrightarrow \quad \mathbf{DB_N}(\sigma) \not\models (\phi_1 \vee \phi_2)[\mathrm{N}]$$

Y cuarto, que ψ sea de forma $(\phi_1 \rightarrow \phi_2)$; en este caso tenemos:

$$\sigma(\psi) = F \quad \Leftrightarrow \quad \sigma(\phi_1) = V \text{ y } \sigma(\phi_2) = F \quad \Leftrightarrow \quad \mathbf{DB_N}(\sigma) \models \phi_1[\mathrm{N}] \text{ y } \mathbf{DB_N}(\sigma) \not\models \phi_2[\mathrm{N}]$$

$$\Leftrightarrow \quad \mathbf{DB_N}(\sigma) \not\models (\phi_1 \rightarrow \phi_2)[\mathrm{N}]$$

□□

[3.22] **COROLARIO:** *Sea* N *una denominación para una instancia* DB, *y sea* σ *asignación veritativa con soporte contenido en el dominio* $\{P_1, P_2, \ldots, P_k\}$ *de* N. *Si* ψ *es una fórmula escrita sin usar el símbolo* $^\circ$ *y tal que* $\sigma(\psi) = F$, *entonces* $\mathbf{DB_N}(\sigma) \not\models \psi[\mathrm{N}]$.

□□

[3.23] **LEMA:** *Sea DB una instancia de base de datos cualquiera; sea* $\mathbf{N}$ *una denominación para DB ; y sea σ asignación veritativa con soporte contenido en el dominio de* $\mathbf{N}$ *, tal que $\sigma(\xi) = V$ para todo elemento ξ del conjunto $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$. Entonces la instancia de base de datos $\mathbf{DB}_{\mathbf{N}}(\sigma)$ es una reparación de la instancia DB.*

Demostración: Revisando la construcción de $\Sigma_{\mathbf{N}}(DB)$ que hemos establecido en [3.5], vemos que el conjunto $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ consta de las fórmulas siguientes:

- Fórmula $\neg P_i$ cada vez que la afirmación $\mathbf{N}(P_i)$ no está en DB , según partes (a) y (b) de [3.5] .

- Fórmula P_i cada vez que la afirmación $\mathbf{N}(P_i)$ pertenece a DB y "respeta la restricción", según partes (a) y (b) de [3.5] .

- Toda fórmula descrita en la parte (c) de [3.5] que no es doble negación [es decir, todas las fórmulas que, cuando DB contiene varias afirmaciones con igual primera componente $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$, dicen "por lo menos una de ellas $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$ es cierta"].

- Toda fórmula descrita en la parte (b') de [3.5] que no es doble negación [es decir, todas las fórmulas que, cuando DB contiene varias afirmaciones con igual primera componente $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$, dicen "como máximo una de ellas $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$ es cierta"].

Sea σ una asignación veritativa tal que $Sop(\sigma) \subseteq \{P_1, P_2, \ldots, P_k\}$. Lo que haremos será verificar que si la instancia de base de datos $\mathbf{DB}_{\mathbf{N}}(\sigma)$ no es una reparación de DB , entonces no es cierto que $\sigma(\xi) = V$ para todo elemento ξ del conjunto $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$.

Supongamos entonces que la instancia $\mathbf{DB}_{\mathbf{N}}(\sigma)$ no es una reparación de DB. Si esto ocurre, debe darse al menos uno de los siguientes casos:

(a) Puede ocurrir que $\mathbf{DB}_{\mathbf{N}}(\sigma)$ contenga una afirmación $R(a, b)$ que no está en DB . En tal caso, dado que la instancia $\mathbf{DB}_{\mathbf{N}}(\sigma)$ por su definición sólo contiene afirmaciones construidas con el dominio activo de DB , existe $P_j \in \{P_1, P_2, \ldots, P_k\}$ que nombra a $R(a, b)$ [es decir $\mathbf{N}(P_j) = R(a, b)$] , de manera que $\mathbf{N}(P_j) \in \mathbf{DB}_{\mathbf{N}}(\sigma)$. Por la definición de $\mathbf{DB}_{\mathbf{N}}(\sigma)$ [3.19] , tenemos $\sigma(P_j) = V$, y por lo tanto $\sigma(\neg P_j) = F$. Pero como $\mathbf{N}(P_j) \notin DB$, revisando

la descripción de $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ con que hemos comenzado esta demostración, se tiene que $(\neg P_j) \in \{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$. Por lo tanto hemos encontrado en $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ una fórmula a la que σ no le asigna el valor V .

(b) Si no se da el caso anterior, es decir si $\mathbf{DB}_{\mathbf{N}}(\sigma)$ contiene solamente afirmaciones que aparecen en DB , puede ocurrir que en $\mathbf{DB}_{\mathbf{N}}(\sigma)$ falte una afirmación $R(a,b)$ que pertenece a DB y que respeta la restricción funcional [o sea $R(a,b) \in DB$ y $b_j \neq b \implies R(a,b_j) \notin DB$] . Existe entonces $P_j \in \{P_1, P_2, \ldots, P_k\}$ tal que $\mathbf{N}(P_j) = R(a,b)$. Como por la definición de $\mathbf{DB}_{\mathbf{N}}(\sigma)$ [3.19] se tiene $\mathbf{N}(P_j) \in \mathbf{DB}_{\mathbf{N}}(\sigma) \iff \sigma(P_j) = V$, la suposición de que $R(a,b)$ no aparece en $\mathbf{DB}_{\mathbf{N}}(\sigma)$ nos dice que $\sigma(P_j) = F$. Revisando nuevamente la descripción de $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$, vemos que $P_j \in \{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$, de modo que en el presente caso también hemos encontrado en $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ una fórmula a la que σ no le asigna el valor V .

(c) Si no se da ninguno de los dos casos anteriores, puede ocurrir que $\mathbf{DB}_{\mathbf{N}}(\sigma)$ contenga dos afirmaciones $R(a,g_i)$ y $R(a,g_j)$ con igual primera componente [es decir, al menos dos: puede haber otras $R(a,g_s)$]. Tenemos entonces letras proposicionales P_{γ_i} , P_{γ_j} en $\{P_1, P_2, \ldots, P_k\}$ tales que $\mathbf{N}(P_{\gamma_i}) = R(a,g_i)$ y $\mathbf{N}(P_{\gamma_j}) = R(a,g_j)$, de modo que $\mathbf{N}(P_{\gamma_i}) \in \mathbf{DB}_{\mathbf{N}}(\sigma)$ y $\mathbf{N}(P_{\gamma_j}) \in \mathbf{DB}_{\mathbf{N}}(\sigma)$, y por lo tanto $\sigma(P_{\gamma_i}) = V = \sigma(P_{\gamma_j})$. Las dos afirmaciones $R(a,g_i)$ y $R(a,g_j)$ tienen que pertenecer a DB , porque hemos supuesto que no ocurre el caso (a) de la presente demostración; con ello, revisando nuevamente la descripción de $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ con que hemos comenzado esta demostración, si miramos su cuarta parte, junto con la correspondiente parte (b') de la definición de $\Sigma_{\mathbf{N}}(DB)$ establecida en [3.5], vemos que para la disyunción

$$\left(\neg P_{\gamma_2} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \cdots$$

$$\cdots \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_2} \wedge \cdots \wedge \neg P_{\gamma_{r-2}} \wedge \neg P_{\gamma_{r-1}}\right)$$

correspondiente a las afirmaciones que tienen a "a" como primera componente, se tiene que en cada uno de sus disyuntos

$$\left(\neg P_{\gamma_2} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) , \ \ldots \ , \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_2} \wedge \cdots \wedge \neg P_{\gamma_{r-2}} \wedge \neg P_{\gamma_{r-1}}\right)$$

falta la negación de sólo una de las letras proposicionales correspondientes a las afirmaciones que tienen a "a" como primera componente, de manera que en

cada uno de estos disyuntos figura por lo menos una de las dos negaciones $(\neg P_{\gamma_i})$ y/o $(\neg P_{\gamma_j})$. Por lo tanto, el hecho de que $\sigma(\neg P_{\gamma_i}) = F = \sigma(\neg P_{\gamma_j})$ nos dice que cada disyunto es falso bajo la asignación σ , y por lo tanto la disyunción también lo es; es decir que

$$\left(\neg P_{\gamma_2} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \cdots$$

$$\cdots \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_2} \wedge \cdots \wedge \neg P_{\gamma_{r-2}} \wedge \neg P_{\gamma_{r-1}}\right)$$

es una fórmula perteneciente a $\left\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\right\}$ a la que σ no le asigna el valor V .

(d) Finalmente, supongamos que no ocurre ninguno de los tres casos anteriores. Entonces el único modo de que $\mathbf{DB}_{\mathbf{N}}(\sigma)$ no sea una reparación de DB es que en $\mathbf{DB}_{\mathbf{N}}(\sigma)$ falte una afirmación $R(a, g_i)$ que está en DB y que no respeta la restricción, y que falten **todas** las afirmaciones pertenecientes a DB que tengan la misma primera componente "a" que $R(a, g_i)$: en efecto, si hubiese en $\mathbf{DB}_{\mathbf{N}}(\sigma)$ más de una de estas afirmaciones, caeríamos en el caso (c) recién visto, y por otro lado, si de cada grupo de afirmaciones con igual primera componente que en DB no respetan la restricción hubiese en $\mathbf{DB}_{\mathbf{N}}(\sigma)$ exactamente una, entonces, habiendo supuesto que no ocurren los casos (a) y (b), $\mathbf{DB}_{\mathbf{N}}(\sigma)$ sería una reparación de DB . Luego sólo nos falta estudiar el caso en que se tienen afirmaciones, llamémoslas $R(a, g_1), R(a, g_2), \ldots, R(a, g_r)$, que tienen igual primera componente, están en DB , no hay otra en DB con esa primera componente, y ninguna de ellas está en $\mathbf{DB}_{\mathbf{N}}(\sigma)$.

En este caso tenemos letras proposicionales P_{γ_1} , P_{γ_2} , ... , P_{γ_r} tales que $\mathbf{N}(P_{\gamma_1}) = R(a, g_1)$, $\mathbf{N}(P_{\gamma_2}) = R(a, g_2)$, ... , $\mathbf{N}(P_{\gamma_r}) = R(a, g_r)$. Como hemos supuesto que ninguna de estas afirmaciones pertenece a $\mathbf{DB}_{\mathbf{N}}(\sigma)$, tenemos que $\sigma(P_{\gamma_1}) = \sigma(P_{\gamma_2}) = \ldots = \sigma(P_{\gamma_r}) = F$; esto significa que $\sigma\left(P_{\gamma_1} \vee P_{\gamma_2} \vee \cdots \vee P_{\gamma_r}\right) = F$; revisando nuevamente la descripción con que hemos comenzado la presente demostración, podemos ver que que la fórmula $\left(P_{\gamma_1} \vee P_{\gamma_2} \vee \cdots \vee P_{\gamma_r}\right)$ pertenece a $\left\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\right\}$, ya que corresponde al caso (c) de [3.5] . Por lo tanto, nuevamente hemos encontrado en el conjunto $\left\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\right\}$ una fórmula a la que σ no le asigna el valor V .

□□

[3.24] **TEOREMA (Completud para $\Sigma_{\mathbf{N}}(DB)$):** *Sea DB una instancia de base de datos formada por afirmaciones de tipo $R(a,b)$ para una cierta relación binaria R , a la que se asigna la restricción funcional; y sea $\mathbf{N}$ una función denominación para DB con dominio $\{P_1, P_2, P_3, \ldots, P_k\}$. Entonces para cada fórmula ψ del lenguaje proposicional de bloqueo en la que no aparece el símbolo de confirmación y tal que $\psi = \psi(P_1, P_2, P_3, \ldots, P_k)$, se cumple que*

$$\Sigma_{\mathbf{N}}(DB) \nvdash \psi \quad \Longrightarrow \quad \mathbf{K}(DB) \not\models \psi[\mathbf{N}]$$

Demostración: Como $\Sigma_{\mathbf{N}}(DB) \nvdash \psi$, por Lema [3.15] tenemos que $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\} \nvdash_{cpc} \psi$. Por Corolario [3.18] , esto significa que existe una asignación veritativa σ con soporte contenido en el dominio $\{P_1, P_2, \ldots, P_k\}$ de $\mathbf{N}$, tal que $\sigma(\psi) = F$ y tal que $\sigma(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$. Esto nos dice dos cosas: primero, que por Corolario [3.22] tenemos $\mathbf{DB}_{\mathbf{N}}(\sigma) \not\models \psi[\mathbf{N}]$; y segundo, que por Lema [3.23] la instancia $\mathbf{DB}_{\mathbf{N}}(\sigma)$ es una reparación de DB . En resumen, lo que hemos mostrado es que la instancia de base de datos $\mathbf{DB}_{\mathbf{N}}(\sigma)$ es una reparación de DB que no satisface ψ con $\mathbf{N}$; por la definición de satisfacción [3.4] , se tiene que $\mathbf{K}(DB) \not\models \psi[\mathbf{N}]$.

□□

Con esto hemos establecido la versión "restringida" de la completud, referente al conjunto finito $\Sigma_{\mathbf{N}}(DB)$. Análogamente a lo que hicimos en el caso de la corrección, para demostrar ahora la versión correspondiente al conjunto infinito $\Sigma^{\infty}_{\mathbf{N}}(DB)$ seguiremos un camino totalmente paralelo al de la versión para $\Sigma_{\mathbf{N}}(DB)$ que acabamos de completar; y, nuevamente, varios de los resultados que ahora veremos se demuestran de modo muy parecido al de su "correspondiente" ya visto, de manera que en algunos de ellos resumiremos la demostración, remitiendo al lector a demostraciones previas para consultar los detalles. Procedamos:

[3.25] **LEMA :** *Si* $\psi = \overline{\psi}$ *(es decir, si en* ψ *no aparece* $^\circ$*) y si* $\Sigma_{\mathrm{N}}^{\infty}(DB) \vdash \psi$ *, entonces* $\{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\} \vdash_{cpc} \psi$.

Demostración: Este resultado se deduce directamente de [2.23] .

□□

[3.26] **LEMA:** *Sea* ψ *una fórmula en la que no aparece el símbolo* $^\circ$ *, y supongamos que* $\{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\} \nvdash_{cpc} \psi$ *. Entonces existe una asignación veritativa* σ *tal que* $\sigma(\psi) = F$ *y tal que* $\sigma(\xi) = V$ *para cada* $\xi \in \{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\}$.

Demostración: Esto es consecuencia directa del siguiente hecho conocido, que ya hemos recordado con anterioridad, referente al cálculo clásico: para cualquier conjunto K de fórmulas de CPC, una fórmula ψ se deduce de K si y sólo si ψ es consecuencia tautológica de K.

□□

[3.27] **LEMA:** *Sea* DB *una instancia de base de datos cualquiera; sea* $\mathbf{N}$ *una denominación para* DB *; y sea* σ *asignación veritativa tal que* $\sigma(\xi) = V$ *para todo elemento* ξ *del conjunto* $\{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\}$ *. Entonces la instancia de base de datos* $\mathbf{DB}_{\mathrm{N}}(\sigma)$ *es una reparación de* DB.

Demostración: De la inclusión $\Sigma_{\mathrm{N}}(DB) \subseteq \Sigma_{\mathrm{N}}^{\infty}(DB)$ se deduce que $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\} \subseteq \{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\}$; en consecuencia, si se sabe que $\sigma(\xi) = V$ para todo elemento ξ del conjunto $\{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\}$, entonces se tiene que $\sigma(\xi) = V$ para todo elemento ξ de $\{\xi \in \Sigma_{\mathrm{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}(DB)\}$. Con ello, si σ tuviese soporte contenido en el dominio de $\mathbf{N}$, podríamos aplicar el Lema [3.23], lo que nos proporcionaría la conclusión que queremos. Pero efectivamente esto sí se verifica, por lo siguiente: revisando la definición de $\Sigma_{\mathrm{N}}^{\infty}(DB)$ establecida en [3.7], vemos que para cada letra proposicional P_i que no pertenece al dominio de $\mathbf{N}$ se tiene $(\neg P_i) \in \{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\}$, de modo que $\sigma(\neg P_i) = V$, luego $\sigma(P_i) = F$. En otras palabras, si una asignación veritativa σ es tal que $\sigma(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB) : \neg\neg\xi \in \Sigma_{\mathrm{N}}^{\infty}(DB)\}$, entonces

ella automáticamente tiene soporte contenido en el dominio de **N**. Por lo tanto $\sigma(\xi) = V$ para todo elemento ξ de $\{\xi \in \Sigma_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma_{\mathbf{N}}(DB)\}$ y el soporte de σ está contenido en el dominio de **N** . Podemos entonces aplicar el Lema [3.23], en virtud del cual la instancia de base de datos $\mathbf{DB}_{\mathbf{N}}(\sigma)$ es una reparación de DB.

□□

[3.28] TEOREMA (Completud para $\Sigma^{\infty}_{\mathbf{N}}(DB)$): *Sea DB una instancia de base de datos formada por afirmaciones de tipo $R(a, b)$ para una cierta relación binaria R , a la que se asigna la restricción funcional; y sea* **N** *una función de denominación para DB . Entonces para cada fórmula ψ del lenguaje proposicional de bloqueo en la que no aparece el símbolo de confirmación, se cumple que*

$$\Sigma^{\infty}_{\mathbf{N}}(DB) \nvdash \psi \implies \mathbf{K}(DB) \not\models \psi[\mathbf{N}]$$

Demostración: Como $\Sigma^{\infty}_{\mathbf{N}}(DB) \nvdash \psi$, por Lema [3.25] tenemos que $\{\xi \in \Sigma^{\infty}_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma^{\infty}_{\mathbf{N}}(DB)\} \nvdash_{cpc} \psi$. Por Lema [3.26] , esto significa que existe una asignación veritativa σ tal que $\sigma(\psi) = F$ y tal que $\sigma(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma^{\infty}_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma^{\infty}_{\mathbf{N}}(DB)\}$. Esto nos dice dos cosas: primero, que por Lema [3.27] la instancia $\mathbf{DB}_{\mathbf{N}}(\sigma)$ es una reparación de DB ; y segundo, que $\mathbf{DB}_{\mathbf{N}}(\sigma) \not\models \psi[\mathbf{N}]$, porque en la demostración de [3.27] hemos visto que si una asignación veritativa σ es tal que $\sigma(\xi) = V$ para cada $\xi \in \{\xi \in \Sigma^{\infty}_{\mathbf{N}}(DB) : \neg\neg\xi \in \Sigma^{\infty}_{\mathbf{N}}(DB)\}$, entonces ella automáticamente tiene soporte contenido en el dominio de **N**, de manera que podemos aplicar el Corolario [3.22], en virtud del cual $\mathbf{DB}_{\mathbf{N}}(\sigma) \not\models \psi[\mathbf{N}]$.

En resumen, lo que hemos mostrado es que la instancia $\mathbf{DB}_{\mathbf{N}}(\sigma)$ es una reparación de DB que no satisface ψ con **N** ; por la definición de satisfacción [3.4] , se tiene que $\mathbf{K}(DB) \not\models \psi[\mathbf{N}]$.

□□

Terminamos así de mostrar exahustivamente que esta interpretación para **BLO-OC** es correcta y completa. Para finalizar el capítulo, esbozaremos a continuación cómo podría ser el funcionamiento de **BLO-OC** en otros casos.

g) Otras restricciones

En nuestro trabajo con instancias de base de datos que son inconsistentes con la restricción funcional, hemos utilizado una serie de resultados que obtuvimos en el capítulo 2, referentes a **BLO-OC**, que son aplicables a cualquier conjunto de premisas construidas sin usar el símbolo de confirmación. No es sorprendente, entonces, que existan algunas otras restricciones de integridad tales que para describir inconsistencia respecto a ellas, basta con modificar la construcción de los conjuntos $\Sigma_{\mathrm{N}}(DB)$ y $\Sigma_{\mathrm{N}}^{\infty}(DB)$, de un modo sencillo. Sin embargo, hay también otras restricciones para las cuales no bastan modificaciones simples del trabajo que hemos desarrollado. Veamos algunos casos, de uno u otro tipo:

(1) Generalicemos la restricción funcional del modo siguiente: dado un objeto a del dominio activo, se admitirán como máximo n objetos b_i tales que $R(a,b_i) \in DB$. La restricción funcional sería el caso correspondiente a $n = 1$. Por ejemplo, $R(A,B)$ podría significar "la persona A ha inscrito en su plan de salud a su familiar B ", y en ese caso la restricción indicaría "se acepta que una persona inscriba como máximo a n de sus familiares en su plan de salud".

Tomemos como ejemplo el caso $n = 3$; aquí la fórmula de primer orden que describe la restricción es la siguiente:

$$\forall x \forall y_1 \forall y_2 \forall y_3 \forall y_4 \big(\neg R(x,y_1) \vee \neg R(x,y_2) \vee \neg R(x,y_3) \vee \neg R(x,y_4) \vee y_1 = y_2$$

$$\vee y_1 = y_3 \vee y_1 = y_4 \vee y_2 = y_3 \vee y_2 = y_4 \vee y_3 = y_4 \big)$$

Examinando la construcción del conjunto $\Sigma_{\mathrm{N}}(DB)$ establecida en [3.5] , tenemos que las partes (a) y (b) de esta construcción permanecerán iguales en el presente caso [adaptando la descripción de la parte (b) a la nueva restricción, del modo natural]; son las partes (c) y (b') las que cambiarán, para reflejar la nueva restricción. Por ejemplo, con esta restricción ($n = 3$) tomemos la instancia $DB = \{ R(a,a) , R(a,b) , R(a,c) , R(a,d) , R(b,a) , R(b,b) , R(d,a) \}$; en este caso, las afirmaciones $R(a,a)$, $R(a,b)$, $R(a,c)$, $R(a,d)$ no respetan la restricción [en cambio $R(b,a)$ y $R(b,b)$ sí la respetan, pues ahora se admiten hasta tres afirmaciones con igual primera componente]. Tomemos la siguiente denominación para DB :

$\mathbf{N}(P_1) = R(a,a)$, $\mathbf{N}(P_2) = R(a,b)$, $\mathbf{N}(P_3) = R(a,c)$, $\mathbf{N}(P_4) = R(a,d)$,
$\mathbf{N}(P_5) = R(b,a)$, $\mathbf{N}(P_6) = R(b,b)$, $\mathbf{N}(P_7) = R(b,c)$, $\mathbf{N}(P_8) = R(b,d)$,
$\mathbf{N}(P_9) = R(c,a)$, $\mathbf{N}(P_{10}) = R(c,b)$, $\mathbf{N}(P_{11}) = R(c,c)$, $\mathbf{N}(P_{12}) = R(c,d)$,
$\mathbf{N}(P_{13}) = R(d,a)$, $\mathbf{N}(P_{14}) = R(d,b)$, $\mathbf{N}(P_{15}) = R(d,c)$, $\mathbf{N}(P_{16}) = R(d,d)$.

Entonces el equivalente al conjunto $\Sigma_{\mathbf{N}}(DB)$ que corresponde al presente caso, es la unión de los cuatro conjuntos de fórmulas siguientes, correspondientes a las respectivas partes de [3.5] adecuadamente modificadas:

(a) $\{ P_1 , P_2 , P_3 , P_4 , P_5 , P_6, \neg P_7 , \neg P_8 , \neg P_9 , \neg P_{10} , \neg P_{11} , \neg P_{12}$

$P_{13} , \neg P_{14} , \neg P_{15} , \neg P_{16} \}$

(b) $\{ \neg P_1 , \neg P_2 , \neg P_3 , \neg P_4 , \neg\neg P_5 , \neg\neg P_6, \neg\neg\neg P_7 , \neg\neg\neg P_8 , \neg\neg\neg P_9$

$\neg\neg\neg P_{10} , \neg\neg\neg P_{11} , \neg\neg\neg P_{12} , \neg\neg P_{13} , \neg\neg\neg P_{14} , \neg\neg\neg P_{15} , \neg\neg\neg P_{16} \}$

(c) $\{ (P_1 \wedge P_2 \wedge P_3) \vee (P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_2 \wedge P_3 \wedge P_4)$,

$\neg\neg[(P_1 \wedge P_2 \wedge P_3) \vee (P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_2 \wedge P_3 \wedge P_4)] \}$

(b') $\{ (\neg P_1 \vee \neg P_2 \vee \neg P_3 \vee \neg P_4)$, $\neg\neg(\neg P_1 \vee \neg P_2 \vee \neg P_3 \vee \neg P_4) \}$

Las partes (c) y (b') reflejan el hecho de que, de las cuatro afirmaciones $R(a,a)$, $R(a,b)$, $R(a,c)$ y $R(a,d)$, en cada reparación hay al menos tres de ellas y como máximo tres de ellas.

El caso general: si la restricción indica que se aceptan como máximo n segundas componentes para una misma primera componente, entonces para cada conjunto dado de afirmaciones en DB que no respete la restricción, digamos habiendo m afirmaciones con igual primera componente (con $m > n$), la fórmula correspondiente a la parte (c) es disyunción de $\binom{m}{n}$ conjunciones de n letras proposicionales cada una (va también, como antes, su doble negación); y la fórmula correspondiente a la parte (b') es disyunción de $\binom{m}{m-n}$ conjunciones de $(m-n)$ letras proposicionales negadas (más la doble negación). Así, en el ejemplo que acabamos de ver, se tenía $n = 3$, y hay un sólo grupo de afirmaciones que no respeta la restricción, con $m = 4$; entonces la fórmula de (c) que hemos escrito consta de $\binom{m}{n} = 4$ disyuntos, cada uno conjunción de $n = 3$ letras proposicionales; y la fórmula de (b') que hemos escrito, consta de $\binom{m}{m-n} = 4$ disyuntos, cada uno conjunción de $(m-n) = 1$ letra proposicional negada.

En resumen, las restricciones de este tipo se pueden tratar de manera casi idéntica a la que se utilizó en el caso de la restricción funcional, con sólo modificaciones menores.

(2) Restricción "Bifuncional" : Esta restrición se aplica a afirmaciones binarias, e impone que, dado un objeto a del dominio activo, puede existir un sólo objeto b tal que $R(a,b) \in DB$, y viceversa, dado un objeto b del dominio activo, puede existir un sólo a tal que $R(a,b) \in DB$. En otras palabras, esta restricción equivale a dos restricciones funcionales; de hecho, muchas veces es presentada de ese modo. Por ejemplo, $R(A,B)$ podría significar "la persona A está casada con la persona B" . Esta restricción puede describirse mediante la fórmula siguiente:

$$\forall x \forall y \forall z \Big[\big(\neg R(x,y) \vee \neg R(x,z) \vee y = z \big) \wedge \big(\neg R(y,x) \vee \neg R(z,x) \vee y = z \big) \Big]$$

A diferencia del caso anterior, establecer un conjunto de fórmulas que describa la aplicación de esta restricción a una base de datos cualquiera, es mucho más complicado que en el caso de la restricción funcional. Si tomamos por ejemplo la instancia de base de datos $DB_1 = \{ R(a,a) , R(a,b) , R(a,c) , R(b,c) \}$, vemos que sus reparaciones con respecto a esta restricción son las instancias $\{ R(a,a) , R(b,c) \}$, $\{ R(a,b) , R(b,c) \}$ y $\{ R(a,c) \}$; si consideramos ahora la instancia $DB_2 = \{ R(a,a) , R(a,b) , R(a,c) , R(b,a) , R(b,b) \}$, sus reparaciones son las instancias $\{ R(a,c) , R(b,a) \}$, $\{ R(a,c) , R(b,b) \}$, $\{ R(a,a) , R(b,b) \}$ y $\{ R(a,b) , R(b,a) \}$. Examinando cuidadosamente estos y otros ejemplos, al intentar enunciar el caso general se observa que, una vez fijada una denominación, la construcción del conjunto de fórmulas adecuado es bastante complicada de describir (para, repetimos, el caso general). De hecho, en este punto empieza a observarse que la construcción de las teorías que permiten describir en **BLO-OC** a las reparaciones de una instancia cualquiera respecto a una restricción de integridad dada, es, al menos en muchos casos, esencialmente equivalente a materializar las reparaciones (calcularlas computacionalmente), lo cual ha sido mostrado inadecuado para ser utilizado de modo práctico en computación (en general; en casos particulares puede no serlo).

(3) Relaciones N-arias : Dado que hemos utilizado funciones de denominación para nombrar afirmaciones como un todo, la ariedad de las relaciones que se utilicen es irrelevante. Así, si por ejemplo utilizamos una relación ternaria T e imponemos la restricción de que para cada par de primeras componentes se admite una única tercera, es decir si utilizamos la restricción dada por la fórmula $\forall x \forall y \forall z \forall w(\neg T(x,y,z) \vee \neg T(x,y,w) \vee z = w)$, una vez definida una denominación el procedimiento será absolutamente idéntico al caso de tener relaciones binarias con restricción funcional. Algo similar ocurre para cualquier otro caso, como los que hemos aludido en la presente sección. Esto es natural, ya que, según hemos comentado, en una lógica proposicional sólo es posible aludir a afirmaciones relacionales como un todo (como hacemos al utilizar las funciones de denominación), de modo que su estructura interna, incluyendo su ariedad, es inexistente a ojos de los sistemas proposicionales.

(4) Instancias con más de una relación : Hasta ahora hemos considerado solamente instancias de base de datos formadas por afirmaciones construidas usando una única relación. Lo más frecuente es que las bases de datos se construyan usando varias relaciones. Dentro de la clase de las instancias formadas por más de una relación, hay algunas para las que el trabajo que hemos efectuado puede aplicarse sin cambios sustanciales. Por ejemplo, supongamos que tenemos dos relaciones con diferente ariedad, digamos una relación binaria R y una relación ternaria T ; a ellas necesariamente se aplican restricciones diferentes (las restricciones pueden ser conceptualmente iguales, como en el ejemplo que vimos en el punto (3), que "en realidad es lo mismo" que la restricción funcional; pero en sentido estricto las restricciones son fórmulas de primer orden, que forzosamente han de ser diferentes si la ariedad de R es distinta a la de T); esto hace que en ocasiones funcionen de modo básicamente independiente la una de la otra; si sabemos cómo aplicar **BLO-OC** a cada una por separado, lo que ocurrirá en muchos casos es que se definirá una función de denominación para la instancia de base de datos, se construirán los conjuntos correspondientes a las partes (a) y (b) de [3.5] del mismo modo en que se hizo en dicha definición (adaptando la descripción de la parte (b) a la nueva restricción, del modo natural), y las fórmulas correspondientes a las partes (c) y (b') referentes a T no tendrán letras

proposicionales en común con las referentes a R, porque el conjunto de afirmaciones de DB que no respetan la restricción T es, por ariedad, necesariamente disjunto con el conjunto de las afirmaciones que no respetan a R, y en muchos casos cada fórmula correspondiente a (c) y (b') solamente contendrá letras que nombran a afirmaciones que no respetan a una sola de las restricciones (como ocurre en el caso de la restricción funcional). Un ejemplo sería tomar una R binaria con restricción funcional y una T ternaria con la restricción de la que hablamos en el punto anterior (3) : en este caso la construcción del conjunto de fórmulas sigue el esquema recién descrito. En cambio, hay otros conjuntos de restricciones en que algunas de ellas, aún teniendo diferente ariedad, pueden influirse unas a otras, pudiendo complicarse mucho la construcción equivalente a cualquiera de las cuatro partes, (a) , (b) , (c) y/o (b') , de [3.5] .

CAPÍTULO 3 - APÉNDICE

En la demostración de [3.10], en el primer paso de la inducción (correspondiente a las demostraciones de longitud 1), se menciona que toda tautología clásica es satisfecha en cualquier instancia de base de datos y con cualquier denominación. El primero de estos dos hechos es sabido para el caso habitual, en el que se usa lógica de primer orden, y usando el concepto de función denominación, esta situación queda traducida a nuestro caso proposicional; esta observación incluye la definición de satisfacción [3.2], que podemos considerar como la traducción proposicional de la definición habitual, correspondiente a lógica predicativa.

Dado que la definición estándar de satisfacción de fórmulas de primer orden en bases de datos es tal que toda instancia satisface toda tautología, en particular toda reparación de una instancia inconsistente satisface toda tautología. Por lo tanto, en un sistema deductivo que pretenda reflejar el funcionamiento de las bases de datos inconsistentes mediante sus reparaciones, como es el caso de **BLO-OC**, necesariamente deben poder deducirse todas las tautologías clásicas a partir de las fórmulas que describen a la instancia de base de datos estudiada. Ahora bien, parece muy difícil definir un sistema deductivo paraconsistente que permita deducir toda tautología a partir de **cada** conjunto de fórmulas que describe a una instancia inconsistente, pero que no permita deducirlas como tesis de la lógica (es decir, sin usar premisas); esta situación casi permite descartar de entrada una serie de lógicas paraconsistentes, como candidatas a ser usadas para describir bases de datos inconsistentes mediante reparaciones, porque la mayoría de los sistemas deductivos paraconsistentes que han sido presentados hasta ahora no permiten deducir toda tautología clásica (usando la terminología de [ref. 2] aludida en el apéndice histórico-filosófico del Capítulo 2, la mayoría de las lógicas paraconsistentes no son paraconsistentes clásicas). Por ejemplo, ninguno de los sistemas deductivos de la jerarquía de sistemas C_n permite deducir toda tautología clásica, de manera que ellos casi pueden ser inmediatamente descartados como candidatos para ser utilizados en la descripción de instancias inconsistentes mediante reparaciones. En resumen, si se desea abordar este problema (estudio de bases de datos inconsistentes usando reparaciones), es sumamente útil estar sobre aviso de esta condición necesaria: de cada conjunto de fórmulas que describa a una instancia inconsistente, debe poder deducirse cada tautología clásica.

* * * * *

Según hemos visto en [2.20], **BLO-OC** es una lógica monótona, es decir que si tenemos conjuntos de fórmulas K_1 y K_2 tales que $K_1 \subseteq K_2$, entonces para cualquier fórmula ϕ se cumple que $K_1 \vdash \phi \Longrightarrow K_2 \vdash \phi$. Algunos autores han sugerido que para describir bases de datos inconsistentes lo indicado parece ser buscar una lógica no monótona, ya que si la información que contiene una instancia no contradice a las restricciones de integridad, puede ser usada para efectuar deducciones, pero al agregar nueva información pueden surgir inconsistencias con las restricciones, lo que en principio inutiliza la información; así, por ejemplo, si una afirmación $P(a)$ es consistente con las restricciones, se debe poder deducir, pero si se agrega información que la vuelve inconsistente con las restricciones, ella ya no debería poder deducirse. Esta idea parece sensata, y vale la pena mencionarla. En el caso de **BLO-OC**, la monotonía no genera problemas, ya que lo que hemos hecho es trabajar con una instancia de base de datos fija, y no pensar en ella como mutable en el tiempo, es decir que nuestro enfoque no considera el caso de una misma instancia a la que se le va agregando información.

CAPÍTULO 4

LAS LÓGICAS DE BLOQUEO

Hemos trabajado ampliamente con **BLO-OC** , sistema deductivo al que nos hemos referido como un ejemplo en particular de las lógicas de bloqueo. En el ejemplo [2.4] se mostró la deducción formal en **BLO-OC** de la fórmula P_1 a partir del conjunto de premisas $\Sigma = \{P_1 , \neg\neg P_1\}$; vimos ahí que a partir de estas premisas, que están escritas sin el símbolo ^{o}, con la regla de inferencia [5] se introdujo una fórmula que sí tenía dicho símbolo, lo cual fue indispensable para poder aplicar las otras cuatro reglas, pues para usarlas se necesitan fórmulas que contienen el símbolo; a continuación, la premisa P_1 sólo pudo integrarse a la demostración con el uso de las reglas de inferencia [1] y [2] ; Después, siempre en [2.4], examinamos una deducción sin premisas, a partir de sólo axiomas, en la que se requirieron las reglas [2] y [3] para poder operar sobre ellos. Estos hechos reflejan lo que podríamos considerar como un mecanismo de bloqueo y desbloqueo de premisas y axiomas, cuyo funcionamiento es básicamente el siguiente: si partimos de premisas que no contienen el símbolo de confirmación, no es posible aplicarles ninguna de las cuatro primeras reglas de inferencia de **BLO-OC**; la única que se les puede aplicar es la regla [5], Obtención de Confirmación; ésta, a su vez, sólo puede aplicarse a las premisas si la doble negación de alguna de ellas es también una premisa, es decir que no se puede usar la regla [5] sobre las premisas si no hay entre ellas al menos un par de forma $\{ \psi , \neg\neg\psi \}$. Una vez aplicada la regla [5] a $\{ \psi , \neg\neg\psi \}$, se habrá obtenido la fórmula ψ^o ; con ella y la premisa ψ se obtiene $\psi \wedge \psi^o$ mediante la regla [1] ; y recién ahora, aplicando la regla [2], se logra "hacer llegar" la premisa ψ a la demostración formal. Notemos además que, después de lo anterior, si ahora se quiere aplicar *Pseudo - Modus Ponens* para usar la premisa ψ de manera "realmente deductiva" (dentro de poco concretaremos esta expresión), se tendrán que usar ambas fórmulas ψ y ψ^o . De manera análoga, para aplicar *Pseudo - Modus Ponens* a un axioma $\phi \wedge \phi^o$, éste debe primero ser "separado" mediante las reglas de inferencia [2] y [3] , de lo contrario permanecerá "bloqueado".

Ahora bien, si hubiésemos elegido otra axiomatización de CPC como base para los axiomas de **BLO-OC**, este mecanismo seguiría funcionando exactamente igual; asimismo, si eliminamos uno de los axiomas, o si elegimos por ejemplo una axiomatización de la lógica proposicional intuicionista, podemos

construir una lógica similar a **BLO-OC**, en la que se podrían efectuar menos deducciones formales que en **BLO-OC**, pero en la que el mecanismo de bloqueo funcionaría igual. Esto nos sugiere que podemos establecer un grupo de sistemas deductivos, caracterizados por este mecanismo de bloqueo y desbloqueo.

En **BLO-OC**, todas las reglas salvo *Pseudo - Modus Ponens* son parte del mecanismo, de manera que éste se basa, en principio, en las otras cuatro reglas. Sin embargo, si para cada par de premisas de forma $\{ \psi , \neg\neg\psi \}$ sustituyésemos la doble negación por la confirmación, es decir cambiando el par $\{ \psi , \neg\neg\psi \}$ por el par $\{ \psi , \psi^{o} \}$, en **BLO-OC** podrían demostrarse las mismas fórmulas que con las premisas originales, aunque por supuesto mediante otras deducciones (esto puede demostrarse); entonces el mecanismo de bloqueo puede funcionar sin el uso de la Obtención de Confirmación (regla [5]), al menos para determinados conjuntos de premisas. En resumen, el "núcleo" del funcionamiento del bloqueo y desbloqueo de premisas y axiomas está constituido, al menos en cierto sentido, por las tres primeras reglas de inferencia, actuando en combinación con la definición de deducción formal (ya que es esta definición la que impide que las premisas se incorporen directamente a la demostración, al establecer que lo único que se puede hacer con un conjunto de premisas es aplicarle las reglas de inferencia).

Asimismo, el funcionamiento de estas tres reglas es el que dicta la elección de la forma de los axiomas de la lógica de bloqueo: escogiendo un conjunto cualquiera de axiomas, como los de CPC o los del cálculo proposicional intuicionista, si ponemos cada uno de ellos en conjunción con la fórmula que indica su confirmación, obtendremos un conjunto de fórmulas de forma $\psi \wedge \psi^{o}$, sobre el cual las tres primeras reglas de inferencia actuarán de modo similar al que lo hacen en **BLO-OC**.

Recordemos también que, según se comentó inmediatamente antes del Lema [2.25], cuando se usan premisas sin el símbolo ^{o} , el poder deductivo "efectivo" de **BLO-OC** reside esencialmente en la regla de *Pseudo - Modus Ponens*, pues con las otras cuatro reglas puede efectuarse en **BLO-OC** un grupo muy limitado de deduciones formales a partir de premisas sin ^{o} (el sentido preciso de estas afirmaciones está dado por el enunciado de [2.25]). Esto reafirma los comentarios antes formulados, respecto a que las tres primeras reglas de inferencia se relacionan principalmente con el mecanismo de bloqueo y desbloqueo de premisas y axiomas, y a que el uso "realmente deductivo" de una premisa o un axioma se logra a través de *Pseudo - Modus Ponens*.

En base a las observaciones anteriores, definiremos la clase de las lógicas de bloqueo del modo siguiente:

[4.1] DEFINICION: Denominaremos **sistema deductivo proposicional de bloqueo** a todo sistema deductivo proposicional basado en el lenguaje proposicional de bloqueo básico (definido en [2.1]) o alguna extensión de él, que cumpla con las cuatro propiedades siguientes: primero, la definición de deducción formal es la misma de **BLO-OC**, es decir la que se estableció en [2.3]; segundo, el conjunto de reglas de inferencia incluye por lo menos a las reglas [1], [2] y [3] de **BLO-OC**, dadas en [2.3]; tercero, los axiomas son de forma $\psi \wedge \psi^{o}$; y cuarto, hay al menos una regla de inferencia que se aplica a conjuntos de fórmulas que son unión de conjuntos de forma $\{ \phi , \phi^{o} \}$.

□□

[NOTA : El papel del cuarto requisito establecido en la definición anterior es, por así decirlo, asegurar que las tres primeras reglas de inferencia efectivamente se usarán; en el caso de *Pseudo - Modus Ponens*, éste se aplica a los conjuntos de forma $\left\{ A , A^{o}, A \rightarrow B , (A \rightarrow B)^{o} \right\}$, cada uno de los cuales es unión de dos conjuntos de forma $\{ \phi , \phi^{o} \}$, y hemos visto que para aplicar esta regla se requiere del uso de las tres primeras. La regla (o reglas) a que se refiere este cuarto requisito de la definición, debería(n) ser, en general, la(s) que cumple(n) el papel propiamente deductivo de la lógica de bloqueo, aunque nuestra definición no obliga a que sea así.]

La partícula "OC" del nombre **BLO-OC** es una indicación de que esta lógica de bloqueo tiene la regla de Obtención de Confirmación (regla [5]). Este hecho es importante, porque, como dijimos en el primer párrafo del presente capítulo, las otras cuatro reglas de **BLO-OC** requieren de fórmulas con el símbolo de confirmación sobre las que aplicarse, de modo que para el uso de premisas escritas en lenguaje clásico es indispensable el uso de la Obtención de Confirmación; y en nuestro trabajo del capítulo 2 vimos que es conveniente trabajar con conjuntos de premisas que no contengan el símbolo ^{o} , pues esta condición es útil para demostrar una serie de propiedades. En este sentido, podemos considerar que,

si bien la regla de Obtención de Confirmación no es indispensable para el funcionamiento del mecanismo de bloqueo y desbloqueo de premisas y axiomas, es sumamente útil tenerla. De manera que podríamos decir que **BLO-OC** es una lógica de bloqueo práctica y "minimal", en el sentido de que tiene las tres reglas básicas, tiene la útil regla de Obtención de Confirmación, y tiene una única regla "propiamente deductiva", *Pseudo - Modus Ponens* .

A continuación, en el siguiente capítulo, presentaremos una segunda lógica proposicional de bloqueo, que será una ampliación de **BLO-OC**, y que nos proveerá de una segunda interpretación para un sistema deductivo paraconsistente perteneciente a la clase de las lógicas de bloqueo (notar que la definición de esta clase no implica que sus integrantes sean paraconsistentes, con seguridad algunos lo son pero probablemente otros no; el que veremos a continuación sí lo es). Finalmente, en el último capítulo, esbozaremos la extensión de **BLO-OC** a lógica de primer orden.

CAPÍTULO 5

GRUPOS DE DISCUSIÓN : BLO-OC-$\Diamond$

Consideremos un grupo de personas que intercambian opiniones, siendo cada participante un razonador "clásico", es decir que cada integrante razona al modo de la lógica clásica. Si todos los participantes están de acuerdo en alguna afirmación, el grupo considera, para efectos prácticos, que esta afirmación es verdadera; mientras que cuando no hay acuerdo con respecto a un punto, el grupo debería considerar que no se sabe si la afirmación en cuestión es verdadera o falsa. Por ejemplo, en la actualidad todos los participantes de cualquier grupo de discusión están de acuerdo en que la tierra es redonda, y consecuentemente se asume, como grupo, que dicha afirmación es verdadera; nadie dice a los demás integrantes "bueno, todos nosotros estamos seguros de que lo es, pero puede que en algún lugar alguien no lo esté, así que no lo demos por hecho" : en la práctica, si todos los participantes están de acuerdo, el grupo da por verdadera la afirmación. Por otro lado, cuando hay desacuerdo, el grupo como tal no concluye que la afirmación examinada es verdadera, ni que es falsa. [NOTA: El primer estudio de un caso como éste parece ser el efectuado por Stanislaw Jaśkowski en su lógica "discursiva" o "discusiva". Ver apéndice histórico-filosófico].

En esta situación, las afirmaciones contradictorias surgen de modo natural; sin embargo, ellas no llevan al grupo de discusión a concluir que cualquier cosa es cierta, ni siquiera a que ambas afirmaciones contradictorias son ciertas (al menos cuando los participantes razonan al estilo clásico, que es el caso que estamos considerando). Podemos ver la situación así: en el caso del grupo de discusión, la existencia de una contradicción tiene un significado (o una interpretación) distinta de la que tiene en un único razonador, que sería el caso en que tradicionalmente de ella se deduce cualquier cosa; en el contexto que ahora estamos considerando, lo que debería deducirse de una contradicción, como grupo, es que aunque con certeza la afirmación ha de ser verdadera o falsa, al no saberse si lo correcto es la afirmación o su negación, de cada una de ellas dos sólo se puede afirmar que existe la posibilidad de que sea verdad. En **BLO-OC**, del conjunto $\{ \psi , \neg\psi \}$ no podía deducirse nada salvo las tesis de esta lógica (las fórmulas que se deducen en **BLO-OC** sin premisas). Ahora queremos que de estas premisas se deduzca además, como mínimo, que cada una de ellas tiene posibilidad de ser verdadera (aunque no debe deducirse que la una o la otra es verdadera, ni ambas).

Podemos reflejar algunos aspectos de la situación que estamos comentando, definiendo una lógica de bloqueo adecuada. Para definir esta lógica, tomaremos el lenguaje proposicional de bloqueo básico, establecido en [2.1], y le agregaremos el conectivo unario $\Diamond$, que representará, como es usual en lógica matemática, la condición de ser posible, es decir que la fórmula $\Diamond\psi$ pretenderá significar "es posible que ψ" . Con ello, agregaremos a **BLO-OC** reglas de inferencia y axiomas referentes al nuevo símbolo.

[5.1] DEFINICION: Agregando el símbolo $\Diamond$ al lenguaje proposicional de bloqueo básico definido en [2.1], denominaremos **BLO-OC-$\Diamond$** al sistema deductivo conformado por la misma definición de deducción formal que **BLO-OC**, dada en [2.3], y por las reglas de inferencia y los axiomas de **BLO-OC**, dados en [2.3], más las reglas y axiomas siguientes :

REGLAS DE INFERENCIA: Si A y B son fórmulas cualesquiera, aparte de las de **BLO-OC**, la siguiente es una regla de inferencia:

[6] $\quad \psi \vdash [\Diamond\psi] \wedge [(\Diamond\psi)^o]$ (Obtención de Posibilidad)

AXIOMAS: Si A y B son fórmulas cualesquiera, aparte de los de **BLO-OC**, las siguientes fórmulas son axiomas:

[A11] $\quad \Big[(\Diamond A) \vee B \rightarrow \Diamond(A \vee B)\Big] \wedge \Big[(\Diamond A) \vee B \rightarrow \Diamond(A \vee B)\Big]^o$

[A12] $\quad \Big[(\Diamond A) \vee (\Diamond B) \rightarrow \Diamond(A \vee B)\Big] \wedge \Big[(\Diamond A) \vee (\Diamond B) \rightarrow \Diamond(A \vee B)\Big]^o$

□□

Dado un grupo de discusión, le asignaremos ahora un conjunto de fórmulas que represente las afirmaciones que se están formulando. Este conjunto cumple un papel similar al que los conjuntos $\Sigma_N^*(DB)$ y $\Sigma_N^\infty(DB)$ cumplían en nuestra descripción de las bases de datos inconsistentes, aunque, como veremos, en algunos aspectos es diferente.

[5.2] DEFINICION (Conjunto $\Sigma(G)$): Llamemos G al grupo de discusión que estamos considerando. Al hacer uno de los participantes una primera afirmación, la representa con la letra proposicional P_1 , la incorpora como un primer elemento del conjunto $\Sigma(G)$, y pregunta a los demás participantes si alguno de ellos niega esta afirmación; si alguien lo hace, se incorpora la fórmula $(\neg P_1)$ al conjunto $\Sigma(G)$, y si nadie la niega, se incorpora la fórmula $(\neg\neg P_1)$. Si alguien formula una segunda afirmación, tiene dos opciones: la primera opción es agregar a $\Sigma(G)$ la fórmula P_2 y preguntar si alguien la niega, dependiendo de lo cual se agrega $(\neg P_2)$ o $(\neg\neg P_2)$, y la segunda opción es, en caso de querer sostener una afirmación que se refiere a la antes formulada, construir la correspondiente fórmula en base a P_1, y entonces preguntar si alguien la niega; por ejemplo, si uno de los participantes afirmó P_1 y otro lo negó [con lo cual en $\Sigma(G)$ tenemos las fórmulas P_1 y $(\neg P_1)$], alguno de los participantes podría ahora afirmar $(P_1 \vee P_2)$, que se incorpora a $\Sigma(G)$; si alguien lo niega, se agrega $\neg(P_1 \vee P_2)$, y si nadie lo niega, se agrega $\neg\neg(P_1 \vee P_2)$. Así se continúa con cada afirmación que cualquier participante quiera agregar, ya sea representada por una nueva letra proposicional, ya sea construida a partir de las fórmulas que previamente se han incorporado. Una vez terminadas las declaraciones de los participantes, se ha completado el conjunto $\Sigma(G)$.

□□

[5.3] OBSERVACION: Hemos supuesto que cada participante del grupo razona clásicamente. Entonces, si a $\Sigma(G)$ se han incorporado por ejemplo P_1 y $(\neg\neg P_1)$, el siguiente participante no puede afirmar $(\neg P_1)$, pues la presencia de $(\neg\neg P_1)$ en $\Sigma(G)$ significa que a todos se les preguntó si niegan P_1 y nadie lo negó. En principio esto impide que en $\Sigma(G)$ aparezcan P_1 , $(\neg P_1)$ y $(\neg\neg P_1)$. Si ocurriese que alguien primero no negó P_1 y después sí lo hace, los demás, que anteriormente no lo negaron, deberían ahora negar la afirmación $(\neg P_1)$, de modo que en $\Sigma(G)$ aparecen P_1 , $(\neg P_1)$ y $(\neg\neg P_1)$. En este caso, a partir de $\Sigma(G)$ en **BLO-OC-$\Diamond$** se podrán deducir P_1 y $\Diamond(\neg P_1)$, lo que significaría que P_1 es verdadera y que $(\neg P_1)$ es posible, lo que delata la inconsistencia de uno de los participantes. Y si todos los participantes incurriesen en la mencionada contradicción, en $\Sigma(G)$ aparecerían P_1 , $(\neg P_1)$, $(\neg\neg P_1)$ y $(\neg\neg\neg P_1)$, con lo cual a partir de $\Sigma(G)$ se podrán deducir P_1 y $(\neg P_1)$, lo que mostraría que el grupo, como tal, se contradice a sí mismo, y por lo tanto debería poder deducir cualquier cosa, lo que efectivamente ocurrirá.

□□

[5.4] **OBSERVACION:** En el capítulo 3, al trabajar con bases de datos, las teorías $\Sigma_N(DB)$ y $\Sigma_N^\infty(DB)$ constaban solamente de letras proposicionales y de negaciones y/o dobles negaciones y/o triples negaciones de letras proposicionales, de modo que cada fórmula que aparecía en ellas contenía una sola letra proposicional. En cambio $\Sigma(G)$, aunque juega un papel equivalente al de los conjuntos antedichos, puede contener afirmaciones compuestas, formadas también con los otros conectivos clásicos, con lo cual en una de estas afirmaciones pueden aparecer varias letras proposicionales.

□□

Notar también que todas las reglas de inferencia y todos los axiomas de **BLO-OC** siguen siendo válidos, la nueva regla de inferencia sólo produce fórmulas que tienen el símbolo nuevo $\Diamond$, y los nuevos axiomas contienen el símbolo $\Diamond$; por lo tanto, lo que puede deducirse de $\Sigma(G)$ en **BLO-OC-**$\Diamond$ sin usar el símbolo $\Diamond$ es lo mismo que se deduciría en **BLO-OC** . Y como para cada grupo de discusión G todas las fórmulas que pertenecen al conjunto $\Sigma(G)$ están escritas sin utilizar el símbolo de confirmación, para deducciones sin el símbolo $\Diamond$ se aplicaría todo lo visto en el primer capítulo.

Veamos un ejemplo referente al significado de la nueva regla :

[5.5] **EJEMPLO:** El primer orador del grupo de discusión G formula la afirmación P_1 ; otro participante niega tal afirmación, y con ello los dos primeros elementos del conjunto $\Sigma(G)$ serán las fórmulas P_1 y $(\neg P_1)$; el primer orador pregunta entonces si hay al menos acuerdo en que $(P_1 \vee P_2)$, y esta vez todos asienten. Tenemos entonces $\Sigma(G) = \{ P_1 , \neg P_1 , (P_1 \vee P_2) , \neg\neg(P_1 \vee P_2) \}$. En **BLO-OC** se tiene que $\Sigma(G) \vdash (P_1 \vee P_2)$, y por consiguiente en **BLO-OC-**$\Diamond$ también, reflejándose así el hecho de que para el grupo como tal, la afirmación $(P_1 \vee P_2)$ es cierta. Ahora bien, por aplicación de la nueva regla [6] a las premisas, más la regla [2], en **BLO-OC-**$\Diamond$ tenemos $\Sigma(G) \vdash \Diamond P_1$ y $\Sigma(G) \vdash (\Diamond \neg P_1)$, de modo que el desacuerdo respecto a la afirmación P_1 se ve adecuadamente reflejado, en el sentido de que no se deduce ni P_1 ni $(\neg P_1)$, pero sí se deduce que cada una de estas dos posibilidades es factible.

□□

[5.6] **EJEMPLO:** Utilicemos el mismo grupo de discusión del ejemplo anterior, de modo que $\Sigma(\mathbf{G}) = \{ P_1 , \neg P_1 , (P_1 \vee P_2) , \neg\neg(P_1 \vee P_2) \}$. En **BLO-OC-$\Diamond$** tenemos $\Sigma(\mathbf{G}) \vdash \Diamond P_1 \wedge (\Diamond P_1)^o$; por otra parte, el esquema axiomático [A6] nos indica que la fórmula $\Big[\Diamond P_1 \rightarrow ([\Diamond P_1] \vee P_3)\Big] \wedge \Big[\Diamond P_1 \rightarrow ([\Diamond P_1] \vee P_3)\Big]^o$ es un axioma; con él y con $\Diamond P_1 \wedge (\Diamond P_1)^o$, después de aplicar a cada una de estas dos conjunciones las reglas de inferencia [2] y [3], obtenemos a través de *Pseudo-Modus Ponens* la fórmula $([\Diamond P_1] \vee P_3) \wedge ([\Diamond P_1] \vee P_3)^o$; usando nuevamente las reglas [2] y [3], obtenemos ahora $([\Diamond P_1] \vee P_3)$ y su confirmación $([\Diamond P_1] \vee P_3)^o$; gracias al axioma nuevo [A11], podemos agregar a la deducción la fórmula $[(\Diamond P_1) \vee P_3 \rightarrow \Diamond(P_1 \vee P_3)] \wedge [(\Diamond P_1) \vee P_3 \rightarrow \Diamond(P_1 \vee P_3)]^o$; separándola mediante las reglas [2] y [3], tenemos $(\Diamond P_1) \vee P_3 \rightarrow \Diamond(P_1 \vee P_3)$ y $[(\Diamond P_1) \vee P_3 \rightarrow \Diamond(P_1 \vee P_3)]^o$; con ellas y con las ya obtenidas $([\Diamond P_1] \vee P_3)$ y $([\Diamond P_1] \vee P_3)^o$, mediante *Pseudo-Modus Ponens* tenemos $[\Diamond(P_1 \vee P_3)] \wedge [\Diamond(P_1 \vee P_3)]^o$; finalmente, aplicando la regla [2], llegamos a que $\Sigma(\mathbf{G}) \vdash \Diamond(P_1 \vee P_3)^o$, lo cual tiene sentido: si en el grupo de discusión no había unanimidad respecto a que P_1 fuese falsa, entonces para el grupo existe la posibilidad de que sea verdadera al menos alguna de las dos afirmaciones P_1 , P_3 (ver el ejemplo anterior [5.5] para recordar las afirmaciones de los participantes).

□□

[5.7] **OBSERVACION:** Si ningún participante se pronuncia sobre una cierta afirmación P_i , a partir de $\Sigma(\mathbf{G})$ no puede deducirse ni P_i , ni $(\neg P_i)$, ni $\Diamond P_i$ ni $\Diamond\neg P_i$, lo cual es razonable. Pero además, esto muestra que la nueva regla de inferencia es significativa, en el sentido siguiente: si en **BLO-OC-$\Diamond$** para cualquier fórmula ψ se pudiese deducir $\Diamond\psi$, se tendría que "cualquier cosa es posible", con lo que sostener que determinada afirmación es probablemente verdadera sería una perogrullada, y de este modo la nueva regla, Obtención de Posibilidad, sería en realidad superflua. El hecho de que esto no ocurra nos indica que la nueva regla de inferencia tiene significado; probablemente la contraparte intuitiva del hecho que estamos comentando es la siguiente: como lo que la lógica **BLO-OC-$\Diamond$** pretende describir son las opiniones del grupo de discusión, mientras éste no se pronuncie sobre una cierta afirmación, **BLO-OC-$\Diamond$** la ignora, como si no existiera, salvo para afirmaciones obvias, es decir para tautologías, como por ejemplo $P_i \vee \neg P_i$.

□□

Con respecto a los axiomas, hemos agregado las dos fórmulas "naturales" [A11] y [A12], cuyo uso permite deducir sólo afirmaciones claramente razonables. Puede pensarse en agregar a la lógica **BLO-OC-$\Diamond$** otros axiomas, referentes a la conjunción o a la implicación, pero esta posibilidad debería ser estudiada con cuidado, porque pueden surgir algunos problemas; por ejemplo, a primera vista puede parecer razonable el axioma $\Diamond A \wedge \Diamond B \vdash \Diamond(A \wedge B)$ (en conjunción con su confirmación), pero en casos como el del Ejemplo [5.5], se deduciría la fórmula $\Diamond(P_1 \wedge \neg P_1)$, que es algo con lo que ninguno de los razonadores clásicos que participan en el grupo de discusión estaría de acuerdo.

En todo caso, si se quisiera establecer rigurosamente completud y corrección para **BLO-OC-$\Diamond$** respecto a grupos de discusión, lo más probable es que sería necesario agregar más axiomas, pero antes que nada habría que estudiar, a nivel semántico, cuál es exactamente el conjunto de todas las afirmaciones que el grupo como un todo debería considerar como "posiblemente verdaderas".

Para finalizar este capítulo, observemos que si el razonamiento de cada participante del grupo no fuese clásico, sino por ejemplo intuicionista, bastaría con cambiar los axiomas, basándolos no en CPC sino en el cálculo proposicional intuicionista.

CAPÍTULO 5 - APÉNDICE

La primera lógica propuesta para describir el funcionamiento de un grupo de discusión mediante las herramientas formales de la lógica matemática, fue la presentada por Stanislaw Jaśkowski en 1948 [referencia 7]. El sistema deductivo que el autor formula en este artículo es propuesto como un modo de tratar no solamente el caso específico de los grupos de discusión, sino también otras situaciones similares; propone llamar "sistema discursivo" a todo aquel del cual no se puede decir que incluye exclusivamente opiniones concordantes entre sí, y aclara que el sistema que está proponiendo, llamado "sistema discursivo bivalente de cálculo proposicional D_2", es uno de entre muchos posibles sistemas discursivos (en castellano se los llama también "discusivos"). Entre otras cosas, el autor se ocupa de articular este sistema de modo que se evita expresamente que en él se puedan efectuar las deducciones que se utilizan habitualmente en lógica clásica para derivar una fórmula cualquiera a partir de una contradicción.

El enfoque adoptado por Jaśkowski al formular D_2 en relación a grupos de discusión, es diferente del que se ha adoptado al tratarlos con **BLO-OC-$\Diamond$**. Hay, eso sí, algunas características "no clásicas" comunes a ambos sistemas deductivos; por ejemplo, ambos son no adjuntivos, en el sentido de que en ellos no es cierto que $\{A, B\} \vdash A \wedge B$.

* * * * *

Para finalizar, indiquemos que Jaśkowski mostró que en su lógica D_2 el principio del Pseudo-Escoto no funciona del mismo modo que en el cálculo clásico, en el sentido de que en D_2 no se pueden efectuar las deducciones "estándar" de una fórmula cualquiera a partir de una contradicción, pero aclaró también que esto no basta para asegurar que esta lógica es paraconsistente, pues podrían existir otras formas, distintas de las habitualmente utilizadas, para deducir cualquier fórmula a partir de una contradicción; de este modo, el autor prevenía al lector respecto a que en el artículo no quedaba demostrado que D_2 fuera un sistema deductivo paraconsistente (todo esto se explicaba usando otras palabras, ya que el término "paraconsistencia" fue establecido recién en la década de 1970).

CAPÍTULO 6

BLO-OC DE PRIMER ORDEN

Para finalizar nuestro trabajo, definiremos un sistema deductivo de primer orden (un sistema deductivo predicativo) correspondiente al sistema proposicional **BLO-OC**, y efectuaremos algunas observaciones referentes a él.

Dados los comentarios formulados en el Capítulo 4 con respecto a los axiomas de las lógicas de bloqueo, lo natural es tomar una axiomatización del cálculo clásico de primer orden, y poner cada una de estas fórmulas en conjunción con su confirmación, de modo análogo a lo hecho en **BLO-OC**. Tomaremos entonces los diez axiomas de **BLO-OC** , y a ellos agregaremos los siguientes esquemas axiomáticos:

(a) $[\forall x A \rightarrow A(x/t)] \wedge [\forall x A \rightarrow A(x/t)]^o$, donde la variable x es sustituida por el término t que es de tipo adecuado (según la definición, en lógica clásica predicativa, de "sustituible").

(b) $[\forall x(A \rightarrow B) \rightarrow (\forall x A \rightarrow \forall x B)] \wedge [\forall x(A \rightarrow B) \rightarrow (\forall x A \rightarrow \forall x B)]^o$

(c) $[A \rightarrow \forall x A] \wedge [A \rightarrow \forall x A]^o$ si x no aparece libre en A .

(d) $(x = x) \wedge (x = x)^o$

(e) $[(x = y) \rightarrow (A \rightarrow A')] \wedge [(x = y) \rightarrow (A \rightarrow A')]^o$, donde A es atómica y A' se obtiene de A sustituyendo x por y las veces que se quiera (posiblemente ninguna).

Análogamente al caso proposicional, en este sistema será posible deducir todas las fórmulas que son tesis del cálculo clásico de primer orden.

Si quisiéramos aplicar este sistema deductivo a bases de datos inconsistentes, lo primero que observaríamos sería que ya no se requiere definir una función denominación, pues las afirmaciones relacionales se pueden construir directamente.

Así, en el caso de la restricción funcional, para establecer un conjunto de fórmulas equivalente a $\Sigma_{\mathrm{N}}(DB)$ o a $\Sigma_{\mathrm{N}}^{\infty}(DB)$, la parte (a) consistiría simplemente en escribir directamente las afirmaciones relacionales que integran la base de datos, con lenguaje de primer orden. También observemos que, en este mismo caso, la parte (b') puede ser sustituida por la fórmula que describe la restricción funcional, más su doble negación: en efecto, cada fórmulas de tipo

$$\left(\neg P_{\gamma_2} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_3} \wedge \cdots \wedge \neg P_{\gamma_{r-1}} \wedge \neg P_{\gamma_r}\right) \vee \cdots$$

$$\cdots \vee \left(\neg P_{\gamma_1} \wedge \neg P_{\gamma_2} \wedge \cdots \wedge \neg P_{\gamma_{r-2}} \wedge \neg P_{\gamma_{r-1}}\right)$$

que integra la parte (b') de $\Sigma_{\mathrm{N}}(DB)$ y de $\Sigma_{\mathrm{N}}^{\infty}(DB)$, será ahora una fórmula de primer orden, de tipo

$$\left(\neg R(a, c_{\alpha_2}) \wedge \neg R(a, c_{\alpha_3}) \wedge \cdots \wedge \neg R(a, c_{\alpha_r})\right) \vee \left(\neg R(a, c_{\alpha_1}) \wedge \neg R(a, c_{\alpha_3}) \wedge \cdots \wedge \neg R(a, c_{\alpha_r})\right) \vee \cdots$$

la cual puede ser demostrada en nuestro sistema deductivo de primer orden, a partir de la fórmula $\forall x \forall y \forall z (\neg R(x, y) \vee \neg R(x, z) \vee y = z)$, que describe a la restricción funcional.

El funcionamiento del mecanismo de bloqueo confiere a este sistema deductivo predicativo ciertos aspectos en común con su correspondiente proposicional **BLO-OC**; por ejemplo, el modo de deducir cualquier tesis de la lógica clásica de primer orden es sugerido por los ejemplos examinados en [2.4], donde se compara una deducción de este tipo en **BLO-OC** con su "correspondiente" en CPC.

Bibliografía

Se presentará primero la bibliografía referente a Lógica Matemática, y a continuación la referente a Computación Teórica. En cada caso, tras listar los trabajos, se orienta brevemente al lector acerca del tema de cada uno de ellos.

Lógica Matemática

[1] Arruda, Ayda: "A Survey of Paraconsistent Logic", incluido en el libro "Mathematical Logic in Latin America [Proceedings of the IV Latin American Symposium on Mathematical Logic, Santiago, 1978]" (editado por Ayda Arruda, Rolando Chuaqui y Newton C. A. da Costa; editorial North-Hollad Publishing Company), págs. 1 - 41 (1980).

[2] Béziau, Jean-Ives: "Are paraconsistent negations negations?", incluido en el libro "Paraconsistency-The Logical Way to the Inconsistent" (editado por Walter Carnielli, Marcelo Coniglio, e Itala M. Loffredo D'Ottaviano; editorial Marcel Dekker), págs. 465 - 486 (2002).

[3] Bobenrieth, Andrés: "Inconsistencias ¿Por qué no?". Editorial Tercer Mundo Editores (1996).

[4] Carnielli, Walter, y Marcos, Joao: "A Taxonomy of C-systems", incluido en el libro "Paraconsistency-The Logical Way to the Inconsistent" (editado por Walter Carnielli, Marcelo Coniglio, e Itala M. Loffredo D'Ottaviano; editorial Marcel Dekker), págs. 1 - 94 (2002).

[5] da Costa, N. C. A.: "Sistemas Formais Inconsistentes (Tese apresentada à Facutade de Filosofia, Ciências e Letras da Universidade do Paraná, em concurso para profesor catedrático da cadeira de Análise Mathemática e Análise Superior)", Rio de Janeiro: Nucleo de Estudios e Pesquisas Cientificas do Rio de Janeiro (1963) - [Curitiba: Editora UPFR, 2a. ed. (1993)].

[6] da Costa, N. C. A.: "On the Theory of Inconsistent Formal Systems", incluido en "Notre Dame Journal of Formal Logic", vol. 15, págs. 497 - 510 (1974).

[7] Jaśkowski , Stanislaw : " Rauchnek zdań dla systemów dedukcyjnych sprzecznych", incluido en "Studia Societatis Scientiarum Torunensis", Sectio A, I, páginas 57 - 77 (1948).

[8] [Traducción al inglés del artículo anterior] Jaśkowski, Stanislaw: "Propositional calculus for contradictory deductive systems", incluido en "Studia Logica" t. XXIV, págs. 143 - 157 (1969).

El libro [3], de Andrés Bobenrieth, entrega una detallada historia de las lógicas paraconsistentes, tanto de sus orígenes y desarrollo, como de los análisis y discusiones filosóficas que a su alrededor se han generado. Si se desea en cambio obtener una visión resumida y rápida de los orígenes e ideas fundamentales de la lógica paraconsistente, puede consultarse el artículo de Ayda Arruda [1], en el que además figura una amplia bibliografía referente al tema.

En cuanto a los aspectos técnicos básicos sobre paraconsistencia, los artículos [4] y [2], que figuran en el libro "Paraconsistency-The Logical Way to the Inconsistent", proveen, respectivamente, el marco teórico básico de la paraconsistencia, y algunos aspectos de carácter muy general del funcionamiento "en abstracto" de las lógicas paraconsistentes (antecedidos, en este segundo artículo, de diversos comentarios de carácter informal o filosófico).

Nos hemos referido al hito que constituyó la llamada "Jerarquía de Sistemas C_n", creada por Newton C. A. da Costa, y publicada por primera vez en 1963 en su país natal, Brasil (en portugués). Este trabajo es [5]. Posteriormente, el autor publicó un artículo en inglés, conteniendo los aspectos esenciales del artículo original, y agregando además otros resultados que había ido obteniendo; este artículo es [6], de modo que, además de ser seguramente más accesible que el artículo original, en él se encontrará un ejemplo arquetípico de sistema lógico paraconsistente, estudiado en diversos aspectos.

En el apéndice del Capítulo 5 hemos hablado del trabajo del lógico polaco Stalislaw Jaśkowski. El artículo original al que nos hemos referido, publicado en 1948 (en polaco) es [7]. El lector puede consultar la traducción al inglés publicada en 1969, que es [8].

Computación Teórica

[9] Abiteboul, Serge; Hull, Richard; y Vianu, Víctor: "Foundations of Databases". Editorial Addison-Wesley Publishing Company (1995).

[10] Arenas, Marcelo; Bertossi, Leopoldo; y Chomicki, Jan: "Consistent Query Answers in Inconsistent Databases", incluido en "ACM Symposium on Principles of Database Systems", págs. 68 - 79 (1999).

[11] Barceló, Pablo, y Bertossi, Leopoldo: "Repairing Databases with Annotated Predicate Logic", incluido en el libro "Ninth International Workshop on Non-Monotonic Reasoning (NMR02), Special Session: Changing and Integrating Information: From Theory to Practice", (editado por S. Benferhat y E. Giunchiglia), págs. 160 - 170 (2002).

[12] Barceló, Pablo, y Bertossi, Leopoldo: "Logic Programs for Querying Inconsistent Databases", incluido en "International Symposium on Practical Aspects of Declarative Languages", editorial Springer-Verlag (2003).

El libro [9] contiene la teoría básica del tema planteada mediante el lenguaje de la lógica matemática.

Los aspectos computacionales del tercer capítulo del presente trabajo se basan en un método ideado por Marcelo Arenas, Leopoldo Bertossi y Jan Chomicki; el artículo original es [10], de 1999. En [11] y [12], Leopoldo Bertossi y Pablo Barceló amplían la idea, agregando nuevos resultados.