UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA BIOMÉDICA

DESARROLLO DE UN ALGORITMO Y SOFTWARE PARA LA INFERENCIA DE MODELOS DE REDES BOOLEANAS. APLICACIÓN EN SISTEMAS BIOLÓGICOS

AUTOR: MARIO RUBIO CHAVARRÍA

TUTOR: GREGORIO RUBIO NAVARRO

Curso Académico: 2019-20

A mi abuelo. Por decir siempre lo que pensaba. Porque creyó en mí hasta el final. Y por enseñarme que todo se consigue a base de esfuerzo, especialmente el ser buena persona.

Resumen

El término cáncer comprende un conjunto de patologías que destacan por una inabarcable variabilidad dentro del ámbito de la Medicina. De entre todas ellas, en este trabajo nos centramos en el cáncer de vejiga no músculo invasivo (NMIBC). A diferencia de otras variedades de cáncer, el NMIBC no consta de una clasificación eficaz. Por el contrario existen múltiples clasificaciones con un grado variable de convergencia entre sí. Esto ha obligado en los últimos años a buscar procedimientos alternativos de clasificación.

Así es como se ha llegado a plantear el modelado matemático como solución. El problema del modelado tradicional es que, aplicado al problema que nos atañe, es demasiado costoso en términos computacionales. Por lo tanto, se necesitan mecanismos alternativos de modelado. Entre los múltiples enfoques que existen, en los últimos años ha cosechado numerosos éxitos el modelado con redes booleanas. Se ha demostrado que representa de forma efectiva la naturaleza de los sistemas a modelar, al mismo tiempo que es eficiente desde un punto de vista computacional.

En la actualidad, los modelos de redes booleanas son obtenidos por combinatoria, buscando el modelo que mejor representa los datos experimentales disponibles. Este proceder supone la obtención de modelos empíricos, fijados *ad hoc* para que demuestren las conclusiones que previamente se persigue que demuestren. En este trabajo se persigue sentar las bases de un proyecto mayor, con el que suplir este problema.

Con este objetivo en mente, ofrecemos un *software* respaldado teóricamente para realizar de forma sistemática y automática la inferencia de modelos de redes booleanas. A diferencia del enfoque tradicional, optamos por acotar la combinatoria mediante un proceder deductivo, para después validar el resultado obtenido mediante resultados experimentales. Así obtenemos modelos validados experimentalmente, aunque no empíricos, dado que todo el proceso de construcción de los mismos se justifica biológicamente. Es por esto que el algoritmo codificado en este *software* reduce el tiempo de cálculo, porque no todos los modelos posibles (combinaciones) son biológicamente coherentes y, en consecuencia, no todos los modelos sobreviven al procedimiento de inferencia.

Palabras Clave: redes booleanas, inferencia, modelos, EMT, NMBIC, cáncer de vejiga.

Abstract

The term cancer is used to speak about a wide range of pathologies. From all of them, in this work, the attention is focused on the non-muscle invasive bladder cancer (NMIBC). In contrast to other kinds of cancer, there is not an effective classification for the NMIBC. On the contrary, there are multiple classifications which do not meet their objective satisfactorily. This is the reason why in the last decades it has been performed an intense effort to achieve new classification procedures.

In the search for new methods, mathematical modelling has been postulated as a possible solution for the problem of cancer classification. Nonetheless, the traditional modelling paradigms are too expensive from a computational point of view. Therefore, it is needed alternative modelling techniques. Among all the existing approaches, the modelling with boolean networks has yielded promising results in the last years. It has been proven that this kind of modelling is an efficient and realistic procedure of representing nature.

Currently, the boolean networks models are obtained through combinatorial algorithms. Looking for the expressions which best explain previous experimental results. This course of action implies the obtention of empiric models devised to obtain the desired conclusions. In this project, the objective is to form the basis of a greater work, with which solve this problem.

With this in mind, it is offered a software built upon a new theoretical model to achieve a systematic and automatized boolean networks inference. Unlike the traditional approach, the strategy is to limit the need for combinatorics, through a deductive procedure, to obtain the models to be validated utilizing experimental results. The deduction process, driven by biological knowledge known beforehand, enables to represent the true nature of the phenomena. This is why the obtained models are not empiric. Consequently, this approach reduces the time spent in combinatorics because not all the combinations suit the inference procedure.

Keywords: boolean networks, inference, models, EMT, NMBIC, bladder cancer.

Índice general

Notación

Por el carácter teórico y técnico del trabajo se maneja un vocabulario y nomenclatura que explicamos en esta sección.

- $\wedge$: operación de AND lógico. En este trabajo se designa también como producto lógico. Equivalente a $*$ en otras nomenclaturas. Ejemplo: $1 \wedge 1 = 1$. En nomenclatura de BoolNet es representado mediante &.
- $\neg$: operación de NOT lógico. Ejemplo: $\neg 0 = 1$. En nomenclatura de BoolNet es representado mediante !.
- $\vee$: operación de OR lógico. En este trabajo se la designa en ocasiones como suma lógica. Ejemplos: $0 \vee 0 = 0$, $1 \vee 0 = 1$. En nomenclatura de BoolNet es representado mediante |.
- $\oplus$: operación módulo 2. Equivale a sumar los valores a uno y otro lado del operador, dividir la suma entre 2 y quedarse con el resto de la división. Ejemplo: $2 \oplus 3 = 1$. Empleando variables lógicas puede representar la negación (NOT) con la unidad. Ejemplo: $1 \oplus 1 = 0$, $0 \oplus 1 = 1$.
- $\otimes$: producto de Kronecker.

Parte I

Memoria

Capítulo 1

Introduccion

El empleo de las Redes Booleanas no ha sido una decisión arbitraria. Por el contrario, es una de las respuestas que se formulan a la cuestión de la clasificación del cáncer. En este capítulo exponemos cuál ha sido el razonamiento que ha llevado a técnicos y médicos a considerar esta particular variante de modelado como la solución al problema que enfrentan.

1.1 El diagnóstico del cáncer

En la actualidad, los tumores extirpados son agrupados de acuerdo a distintas clasificaciones clínicas basadas en el criterio del facultativo, en este caso del patólogo. Las distintas categorías de clasificación se corresponden con perfiles clínicos y, hasta la fecha, pese a su naturaleza cualitativa (pues están basadas en la apariencia externa de la muestra) han desempeñado bien su objetivo en la mayoría de las variantes de cáncer estudiadas. No obstante, existen múltiples variantes de cáncer para las que no existe clasificación efectiva.

Cuando se habla del cáncer siempre es necesario concretar todo lo posible. Esto es así debido a que el cáncer antes que una enfermedad es una familia de patologías, caracterizada por su diversidad y extensión. Tanto es así, que incluso el mismo cáncer en el mismo paciente podría desarrollar perfiles clínicos diferentes atendiendo a múltiples factores. En este contexto de extrema variabilidad, dentro de este trabajo se ha optado por focalizar todos los esfuerzos dentro del cáncer de vejiga. En concreto en el cáncer de vejiga no músculo invasivo (NMIBC por sus siglas en inglés).

¿Por qué el NMIBC? Por múltiples razones. La primera de ellas es que es esta variedad de cáncer en la que se concentran los esfuerzos del grupo de investigación del IMM (Instituto de Matemática Multidisciplinar) en el que me he integrado para realizar este proyecto. No obstante, más allá de la coyuntura en la que se ha desarrollado este trabajo, el NMIBC, como se desarrolla más adelante, no tiene una clasificación efectiva. En otras palabras, es un variedad de cáncer sin criterios clínicos objetivos, cuantitativos, que permitan una predicción efectiva de la evolución del paciente. Estamos hablando de biomarcadores, indicadores cuantitativos que actualmente constituyen la punta de lanza en la lucha contra otras variedades de cáncer tales como el gioblastoma multiforme (Müller Bark y col., 2020). Es por esto que la meta perseguida en este trabajo no solo se constituye como un reto de altura, sino como una pregunta todavía por responder tras décadas

de investigación. Una pregunta cuya respuesta atiende a una necesidad clínica real y que dista mucho de la capacidad de este proyecto. Es por este motivo que este trabajo pretende arrojar una chispa de luz dentro de una línea de investigación que necesariamente ha de brillar mucho más.

Conforme a la Sociedad Española de Oncología Médica (SEOM), una de las clasificaciones más empleadas para este tipo de cáncer es el sistema TNM, desarrollado por el American Joint Committee on Cancer (AJCC). En esta clasificación, al tumor se le asignan 3 letras en función de su posición con respecto la pared de la vejiga (T), su propagación a los ganglios linfáticos adyacentes (N) y su colonización de regiones lejanas (M). Las distintas combinaciones de letras son agrupadas en 4 estadios diferentes para poder realizar una valoración de la posible evolución clínica del paciente. Esta clasificación se nutre de una fase prequirúrgica y otra postquirúrgica, es decir, con el tumor extirpado. Es esta última la que se considera que aporta una mayor fiabilidad a la clasificación y, como se indica en el párrafo anterior, se construye sobre el criterio del patólogo. Un criterio cualitativo basado en la observación al microscopio.

Esta clasificación, pese a ser una de las mejores alternativas y de haber sido empleada durante décadas, a menudo no cumple con su objetivo. De forma que consigue resultados incoherentes, incluso erróneos, con respecto a la evidencia previa. Esto se debe a que los distintos perfiles de esta familia de tumores parten de manifestaciones clínicas similares en las etapas iniciales de la enfermedad, lo que dificulta no solo el diagnóstico sino la creación de una clasificación coherente de las distintas variedades tumorales. Así es como en las últimas décadas han aparecido múltiples clasificaciones, destacando el trabajo de la universidad de Lund (Marzouka y col., 2018) o el de la universidad de Aarhus (Lerner y col., 2016). Así mismo, se han logrado avances en la caracterización tumoral a través del análisis genético de las muestras (Hedegaard y col., 2016; Lerner y col., 2016; Choi y col., 2017). Sin embargo, ninguna de estas clasificaciones llega ni a cumplir su objetivo ni a ser coherente con el resto. Aunque es cierto que tienen puntos coincidentes, en opinión del autor entorno a Hedegaard y col., 2016; Choi y col., 2017. Por lo tanto, tras años de investigación, a día de hoy sigue siendo necesaria una clasificación efectiva del NMIBC para su utilización clínica.

En la búsqueda de enfoques alternativos con los que solucionar las incoherencias de clasificaciones anteriores, el estado del arte (Joo y col., 2018) se introduce en el modelado matemático como herramienta para abordar esta tarea. Es decir, si bien no es posible agrupar a los pacientes en grandes categorías taxonómicas, centrémonos en un análisis individualizado, personalizado, de la evolución de cada paciente. Construido sobre parámetros cuantitativos como los que puede ofrecer el análisis genético de las muestras tumorales. Es en este punto cuando nos alejamos del mundo de la Medicina para buscar soluciones dentro del de la Ingeniería. Procedimientos ampliamente empleados en la actualidad para la construcción de aeronaves, el cultivo de cosechas o el desarrollo de prótesis pueden ser aplicados, directa o indirectamente, para la obtención de un método efectivo de clasificación del cáncer de vejiga.

Como se adelantaba anteriormente, los biomarcadores son de gran importancia en el contexto clínico actual, y constituyen el elemento perfecto sobre el que centrar este modelado matemático. Aunque se pueden emplear otras variantes de biomarcadores (Müller Bark y col., 2020), dado el enfoque biológico adoptado en este proyecto, hemos elegido trabajar con biomarcadores genéticos. Que son los que actualmente se emplean en el estado del arte para modelar procesos biológicos presentes en esta y otras variedades de cáncer (Joo y col., 2018).

1.2 Modelos de gran dimensionalidad

El modelado matemático, es decir, la representación de sistemas dinámicos mediante expresiones matemáticas, se desarrolla dentro del Cálculo y, más en profundidad, dentro de la Teoría de Control. Entiéndase la primera tanto en su vertiente diferencial como lógica. En Ingeniería, para primero representar los fenómenos, y posteriormente desarrollar los sistemas de control oportunos (PI, PID, etc.) se emplea la Teoría de Control, construida en sus totalidad sobre el cálculo de ecuaciones diferenciales. Es así como la Teoría de Control ya ha sido empleada en multitud de procesos biológicos como el cultivo celular o el control de la fermentación. Así pues, desde un enfoque ingenieril, el primer paso en el estudio de la dinámica del cáncer sería su formulación mediante un sistema de ecuaciones diferenciales. La caracterización matemática, no de la enfermedad, sino del proceso biológico subyacente.

No obstante, los procesos biológicos no son procesos físicos como con los que pueda lidiar la Ingeniería en la práctica habitual. Complejos como pueden ser, estos procesos (p. ej. el despegue de un transbordador o la automatización de una cadena de producción) constan de un número reducido de agentes participantes. Lo que a priori reduce la complejidad de los sistemas de representación y la adquisición de la información. Por el contrario, los sistemas biológicos se caracterizan por un gran número de agentes presentes en sus dinámicas. Así mismo, estos agentes toman partido en gran cantidad de subsistemas anejos, lo que dificulta no solo la representación de los mismos sino el manejo de su información. Es por esto que la representación de estos sistemas es ardua, desalentadora, porque supone realizar simplificaciones, asumir preceptos para conseguir modelos de grandes dimensiones, es decir, con un gran número de parámetros que ajustar a consecuencia del gran número de variables. En otras palabras, agentes que participan de las dinámicas del sistema. Esta dimensionalidad trae consigo dos problemas.

1. **Falta de información**: la falta de información es un problema biológico, derivado de la naturaleza del fenómeno a modelar. Se deriva del hecho de que nunca va a ser posible representar el sistema en su totalidad. Debido al gran número de participantes así como su grado de interrelación entre sí y con otros sistemas, la representación de cada agente y el número de representaciones nos obliga primero a simplificar las expresiones empleadas y luego a limitar el número de variables, lo que supone de facto simplificar el sistema en sí, imponer restricciones o crear variables que aglutinen efectos tanto desconocidos como complejos.
2. **Tiempo de cálculo**: este es un problema técnico derivado de la complejidad del modelo. Alta dimensionalidad implica un alto número de parámetros que calcular.Como resultado, la inferencia de modelos en un entorno de recursos computacionales limitados supone grandes tiempos de cálculo. Tomemos como referencia el entrenamiento de una red neuronal artificial, un problema computacionalmente similar al que abordamos en este trabajo, el cual puede consumir en un ordenador común varios días en función del modelo de red.

Estos problemas no son en absoluto triviales y han de ser enfrentados desde un punto de vista teórico y aplicado. Pongamos por ejemplo que un modelo en Ingeniería pudiera tener 20 variables, cada una con múltiples parámetros dentro de extensas expresiones algebraicas. En el caso biológico nos encontramos con modelos cuya versión reducida puede constar de 400 genes (variables), pero cuyo desarrollo se puede extender hasta abarcar los 2000 o 3000 genes. Con modelos de esta

magnitud es imposible un enfoque tradicional. De modo que el problema de la dimensionalidad ha sido objeto de estudio desde los años sesenta del siglo pasado y todavía suscita cuestiones en algorítmica, Teoría de Grafos y Genética. Cuestiones que distan mucho de ser respondidas a día de hoy aunque han presenciado el desarrollo de métodos de trabajo y representación alternativos como pudieran ser las redes bayesianas. Es en esta familia de enfoques alternativos donde se posiciona nuestro trabajo, dentro de las llamadas Redes Booleanas.

1.3 Modelos de Redes Booleanas

Las redes booleanas son simplificaciones lógicas de sistemas de ecuaciones diferenciales, en esencia, los modelos que hemos comentado en apartados anteriores. Su representación en forma de redes booleanas ha cosechado éxitos importantes (Murrugarra y Dimitrova, 2015; Zhou y col., 2016; Yuan y col., 2017) desde sus inicios a mediados del siglo XX. Existe una correspondencia parcial entre los modelos analíticos y las redes booleanas, de tal forma que las soluciones de dicho modelo se corresponden con los estados finales de la red, lo que se conoce como atractores. Así pues, los perfiles clínicos, es decir, el pronóstico del paciente, lo que en Teoría de Control se conoce como régimen permanente, y que en el modelo está representado por la solución del sistema, en la red booleana se corresponde con el atractor (o atractores si el sistema tuviera varias soluciones). Es por esto que esta variedad de representación es muy potente, porque podemos obtener las mismas conclusiones que obtendríamos con modelos más complejos pero con un tiempo de cálculo reducido. Esto es así debido a que, por la propia naturaleza de las redes, las operaciones y expresiones son mucho más simples, asequibles desde un punto de vista computacional.

No obstante, estos modelos cuentan con una gran limitación: la representación del tiempo. En las redes booleanas el tiempo no se encuentra directamente representado, es por esto que a menudo acabamos en expresiones markovianas de los sistemas estudiados (se profundizará más adelante en este concepto), o en variantes donde se establecen dependencias de mayor alcance entre diferentes estadios de desarrollo del sistema. Es este el precio a pagar por las simplificaciones efectuadas.

En consecuencia, para un mismo modelo existen diferentes redes capaces de representarlo dado un conjunto de restricciones. En otras palabras, hemos cambiado un problema de inferencia por un problema de combinatoria. La realidad es que este problema de combinatoria, aunque es ventajoso en términos de cómputo sigue sin ser trivial. Es decir, encontrar la mejor red va a depender estrictamente de la cantidad de información a priori sobre el sistema a modelar, es decir, del número de restricciones que podamos introducir. En la práctica, cuando las restricciones son insuficientes, a menudo se recurre a simplificaciones o modificaciones sobre el sistema. El objetivo es que la red obtenida tenga sentido biológico, pues no hemos de olvidar que estamos aplicando una solución matemática a un problema biológico. Tomando como símil la Física, a menudo nos encontramos con soluciones con sentido matemático pero no físico, soluciones tales como un valor de tiempo negativo. Este tipo de soluciones son inmediatamente rechazadas. Es exactamente el mismo proceder con los problemas biológicos, con la diferencia de que a menudo estas contradicciones recaen sobre sutiles relaciones bioquímicas que necesitan ser revisadas por expertos. No obstante, aunque en la actualidad las redes son revisadas, nada nos impide introducir restricciones en forma conocimiento a priori, como se demostrará a lo largo de este trabajo.

Por otro lado, las simplificaciones. En Teoría de Control, el régimen permanente no es lo único de interés. Tomemos por caso un freno de automóvil, en su diseño no solo nos interesa cuándo se detendrá el vehículo. Buscamos cuántas oscilaciones producirá así como la magnitud de las mismas. Porque no es lo mismo pasar de 120 Km/h a 30 Km/h en 30 que en 5 segundos. El concepto es similar, no solo nos interesa cuál es el pronóstico del paciente a 5 años sino cuáles serán los estados que atravesará. Por ejemplo, la clasificación del TNM no es única ni lineal, es decir, ni todos los pacientes tienen por qué atravesar todos los posibles escenarios ni todos tienen por qué hacerlo en el mismo orden. Aunque el discurrir natural de la enfermedad sí que obligue a todos a transitar por todos los estadios.

Esta multiplicidad de posibilidades no termina de casar con la lógica booleana, donde un estado del sistema conduce a uno, y solo a uno, del resto de estados. Por consiguiente, en la representación lógica a menudo se desarrolla la complejidad inicial (Das y col., 2019), ya de por sí elevada, de la red a través de diversos mecanismos como la creación de nodos que no están relacionados con agentes o la introducción de variables *dummy*. En otras ocasiones, al perder la información temporal, terminamos por modificar el transitorio original del sistema, apareciendo nuevas restricciones de facto. Bien por un aumento de complejidad, bien por la alteración de la dinámica original del sistema, en ocasiones aparecen normas que entran en conflicto entre sí. Como consecuencia, nuestro problema de combinatoria no tiene solución. En este contexto, a menudo se consulta a expertos para que, sin perder el sentido biológico del fenómeno a modelar, se pueda simplificar el conjunto de restricciones hasta obtener un sistema resoluble (Robeva y col., 2013). Este proceder, pobre donde los haya, es comúnmente empleado. Sin embargo, en este trabajo mostramos cómo podemos evitar recurrir a este mecanismo siempre que contemos con información previa de calidad, coherente, que nos permita aumentar la complejidad inicial del sistema con sentido biológico.

1.4 El objetivo de este trabajo

Como relatábamos en apartados anteriores, el modelado de sistemas biológicos es complejo desde el punto de vista teórico y técnico. Exige el empleo de sofisticadas herramientas matemáticas y un alto poder computacional para llevarlo a término así como, una vez efectuado, necesita de validación por expertos de cara a ser confiable. Durante las últimas décadas, grandes esfuerzos se han llevado a cabo con la intención de reducir estas tres problemáticas dentro del auge de la Medicina de Precisión. Así pues, el objetivo de este proyecto es desarrollar una herramienta capaz de, en primer lugar reducir la potencia de cómputo necesaria para desarrollar modelos biológicos de redes booleanas y, en segundo lugar, agilizar el proceso de validación mediante el empleo de información previa. Para reducir el conjunto de soluciones posibles a las que son biológicamente plausibles.

Para ello partimos de grafos, representaciones gráficas de los sistemas a modelar. Y desarrollamos un algoritmo fundado en Teoría de Grafos (Layek, 2012), la teoría desarrollada en torno a las funciones booleanas anidadas canalizantes (NCBF por sus siglas en inglés) (Li y col., 2013) y la representación algebraica de expresiones lógicas dentro del producto del semi-tensor (STP por sus siglas en inglés) (Cheng y col., 2011; Akutsu, 2018). Este algoritmo se desgrana punto por punto a lo largo de este trabajo, como resultado del Trabajo Fin de Grado del au-

tor. Así mismo, ha sido completamente implementado en Python, en el software disponible en `https://github.com/mrubio-chavarria/boolean_networks`.

Por otro lado, al final del trabajo se encuentran los anexos. *No deben entenderse como parte del documento*. Son información complementaria, gráficos y demostraciones matemáticas, en las que el lector puede estar interesado pero que *no son imprescindibles para el entendimiento del contenido de la memoria*. Es este el motivo por el que se ha decidido incluir el apartado de anexos.

Capítulo 2

Definición y alcance del proyecto

Capítulo donde se estructura toda la ejecución del proyecto en unidades de desarrollo. Toda la labor desempeñada en el trabajo se recoge en este capítulo. Parte de ella planificada o acotada desde el principio.

2.1 Objetivo del proyecto

El objetivo de este trabajo es el desarrollo de una herramienta capaz de reducir la combinatoria empleada en el desarrollo de estos modelos mediante la inferencia e introducción de información en el propio grafo que representa el sistema. Ese objetivo se estructura en: 1) estudio y adaptación del marco teórico existente, 2) desarrollo de algoritmos eficientes de ejecución y 3) codificación del *software* sobre la estructura fijada por dichos algoritmos. Así mismo, una vez que el código haya sido desarrollado, se procederá a su validación mediante su aplicación sobre un sistema biológico.

2.2 Descripción de las unidades de desarrollo

Dada la naturaleza teórica del proyecto, tal vez sea más adecuado emplear el término unidades de desarrollo frente a otras variantes como unidades constructivas. El motivo es que durante la realización del proyecto no se ha construido ningún objeto o maquinaria, por el contrario se han desarrollado marcos teóricos, algoritmos y*software*. Esa es la razón de que se vaya a emplear esa terminología. Se agrupan en las secciones siguientes.

2.2.1 Marco teórico

Estudio del estado del arte en el modelado de sistemas biológicos

Esta unidad de desarrollo se corresponde con el estudio del estado del arte de las técnicas de modelado biológico centradas en el empleo de NCBF (*nested canalizing boolean functions*). En esencia, labor de estudio e investigación cuyo principal fruto ha sido la comprensión de: Kauffman y col., 2003, 2004; Li y col., 2013; Joo y col., 2018.

Estudio del estado del arte en Teoría de Grafos aplicada a redes biológicas

Estudio e investigación de la teoría y mecanismos existentes para la inferencia de relaciones contradictorias entre *pathways*, dentro de la representación de redes biológicas mediante grafos dirigidos. El principal resultado ha sido el descubrimiento y aplicación de los mecanismos descritos en Layek, 2012.

Estudio del Álgebra de Boole desde el Álgebra Lineal

Estudio de toda la teoría existente del producto izquierdo del semi-tensor (STP). Toda la teoría expuesta en Cheng y col., 2011; Akutsu, 2018. Así como el estudio de los algoritmos recogidos en Cheng y col., 2011.

Adaptación y desarrollo sobre los marcos teóricos existentes

Desarrollo de la base teórica de nuevas estrategias para la inferencia de modelos de redes booleanas sobre la integración de los marcos teóricos de las NCBF y la Teoría de Conflictos expuesta en (Layek, 2012). Así como las primeras implementaciones en modelos simplificados. Finalmente, esta labor se concretó en los dos apartados siguientes:

- Combinación de los marcos teóricos de las NCBF y de la Teoría de Conflictos. Demostración de la validez del proceso iterativo para la modificación de la red original. Es de resaltar que el principal fruto de este apartado no son solo los algoritmos sino las demostraciones expuestas los anexos. Tanto los algoritmos como las demostraciones son contenido original de este proyecto.
- Desarrollo de un proceso de validación alternativo de los atractores basado en la aproximación al Álgebra de Boole desde el Álgebra lineal (STP).

2.2.2 Diseño de algoritmos

Inicialmente se preveía la necesidad de algoritmos eficientes sobre los que construir las distintas secciones del *software*. No obstante, no se podía prever ni la naturaleza ni el objetivo de dichos algoritmos dado el carácter exploratorio y de investigación del proyecto. Es por eso que la definición de las unidades de desarrollo descritas en este capítulo, se produjo durante el discurrir del propio proyecto.

Algoritmo para la generación eficiente de NCBF

Unidad de desarrollo destinada a la obtención del algoritmo descrito en el apartado 4.3.2.

Algoritmo para la resolución de conflictos entre pathways dentro de una red biológica

Unidad de desarrollo destinada a la obtención del algoritmo descrito en el apartado 4.5.

Algoritmo para la resolución de conflicto dentro de un nodo en una red biológica

Unidad de desarrollo destinada a la obtención del algoritmo descrito en el apartado 4.5.1.

2.2.3 Desarrollo del software

Arquitectura

Desarrollo del marco de *software* sobre el que implementar los algoritmos. Es decir, los ficheros empleados con las estructuras de datos y relaciones en lo que se refiere a programación funcional y orientada a objetos.

Implementación de los algoritmos

Tiempo dedicado a la codificación de los algoritmos en los ficheros especificados para ello. En este apartado se contabilizan todas las horas dedicadas a revisión del *software* así como la verificación de la implementación de los algoritmos (Capítulo 6).

Capítulo 3

Conceptos iniciales

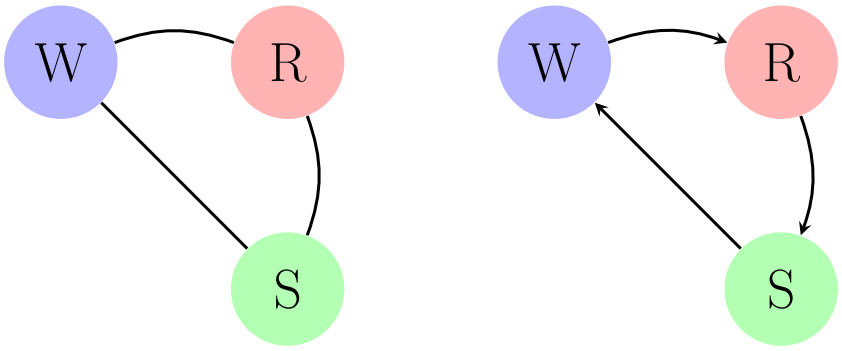
Antes de continuar es necesario introducir varios conceptos para colocar al lector en contexto. Más allá de las definiciones, el objetivo es transmitir las ideas entorno a las que orbitan este y otros muchos trabajos citados en la bibliografía.

Este capítulo tiene dos secciones. Una en la que se introducen todos los conceptos de naturaleza matemática o técnica y otra en la que se exponen las definiciones más biológicas.

3.1 Conceptos técnicos

Grafo

Aunque el concepto de grafo comprende mucho más, en este trabajo un grafo es la representación de una red. Un conjunto conectado de nodos. Representaciones como las que se pueden apreciar en la Figura 3.1. La diferencia entre un grafo no dirigido (Figura 3.1a) y uno dirigido (Figura 3.1b) radica en la direccionalidad de los segmentos, lo que tiene implicaciones en las relaciones entre los distintos nodos. Por ejemplo, en el grafo de la Figura 3.1a, los nodos W y S se encuentran conectados de forma bidireccional. Por el contrario, en la Figura 3.1b, el nodo S se encuentra conectado directamente con el W aunque el W no con el S, tiene que transitar a través del nodo R para lograr dicha comunicación.

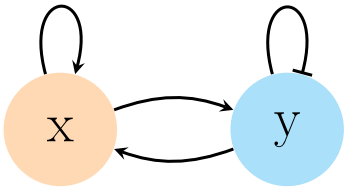


(a) Grafo no dirigido **(b)** Grafo dirigido

Figura 3.1: Principales variantes de grafos.

$$x' = x + y$$
$$y' = 3x - y$$

(a) Expresiones del sistema.



(b) Grafo del sistema

Figura 3.2: Sistema de ejemplo de ecuaciones diferenciales. Representado mediante expresiones matemáticas y mediante un grafo dirigido. En la Figura 3.2a, la prima representa la derivada. En la Figura 3.2b los segmentos acabados en punta indican relaciones de activación mientras que los segmentos no acabados en punta representan relaciones de inhibición. Es decir, el nodo ubicado al inicio del segmento activa o inhibe al que se encuentra próximo al extremo contrario del segmento. El que acaba con una flecha o con una superficie plana.

Sistema de ecuaciones diferenciales

Un sistema de ecuaciones diferenciales es un conjunto de ecuaciones que incorpora derivadas en las expresiones que lo componen. Como se comentaba en apartados anteriores, un sistema de ecuaciones diferenciales puede ser representado mediante una variante de grafo, un grafo dirigido. En tanto que el grafo es una representación abstracta y simplificada del propio sistema. Por ejemplo, en la Figura 4.1 apreciamos un sistema representado de ambas formas.

En el sistema de la Figura 3.2b x e y son funciones de t así como x' e y' representan sus derivadas con respecto a la variable temporal. En el caso de querer representar el sistema de ecuaciones diferenciales mediante un grafo, nos encontraríamos que, dado que es una simplificación, múltiples sistemas de ecuaciones diferenciales pueden ser representados con el mismo grafo. Por ejemplo, si en el sistema de la Figura 3.2a sustituimos la expresión de x' por $x' = e^x + y$, el sistema sigue siendo representado por el mismo grafo.

Red booleana

En principio, una red booleana es una simplificación de un sistema de ecuaciones diferenciales dentro del Cálculo Lógico. Es decir, una forma alternativa de representar los mismos sistemas y expresiones, solo que empleando funciones y operaciones lógicas. Una variable lógica (en lógica bivaluada) es aquella que puede manifestar como valores 0 o 1 exclusivamente. Una función lógica, aquella que depende de variables lógicas, se compone de operaciones lógicas y su variable dependiente expone como valores 0 o 1. Por ejemplo, una red booleana capaz de representar el sistema de la Expresión 3.2a es la que podemos observar en la Expresión 3.1.

$$\begin{aligned} x_{next} &= x \vee y \\ y_{next} &= x \vee \neg y \end{aligned} \tag{3.1}$$

Esto es la representación algebraica de una red booleana. Al igual que ocurre con los grafos, como las redes booleanas son simplificaciones de sistemas de ecuaciones diferenciales, es posible representar el mismo sistema de ecuaciones mediante diferentes redes booleanas. Por otro lado, a partir de la Expresión 3.1 se habla indistintamente de x_{next} y x. El sufijo *next* se emplea para distinguir el valor que presenta el nodo en un paso de tiempo del valor que presenta en el paso siguiente. No obstante, en esencia simbolizan el mismo nodo. El empleo de esta distinción podría

suponer incurrir en cierto «abuso de notación», aunque por claridad en la exposición es conveniente su utilización y aparece en otros trabajos (Layek, 2012).

Por otro lado, en cada expresión, el miembro izquierdo nos indica el nodo representado, mientras que el derecho la función que rige su valor. Como resultado, cada expresión de la red (nodo) es de la forma de la Expresión 3.2. Donde Q es un nodo cualquiera, f la función booleana asociada al nodo Q y P, U y V las variables lógicas de las que depende f.

$$Q = f(P, U, V) \tag{3.2}$$

Acorde con 3.2, la variable independiente de f nos indica el valor que ha de presentar el nodo Q para el tiempo $t + 1$. Mientras que f se evalúa sobre los valores de P, U y V para tiempo t. Es decir, tomando la expresión de y, $y_{next} = x \vee \neg y$, si en tiempo t, $x = 0$ y $y = 1$, en tiempo $t + 1$, $y = 0$, o lo que es lo mismo, $y_{next} = 0$ en tiempo t. Así, el estado de la red en tiempo $t + 1$ viene dado exclusivamente por su propio estado para tiempo t. Conforme a lo expuesto anteriormente, hay que resaltar que el significado del símbolo = no es el común. Es decir, la expresión $y_{next} = x \vee \neg y$ no significa que el nodo y sea una composición de los nodos x e y. El = indica una relación concreta: *si el nodo «x» se activa o el nodo «y» se inhibe en el paso temporal t, el nodo «y» se activará en el paso temporal t + 1.* Así pues, sirve para describir las relaciones entre los comportamientos de los nodos con desfase de un paso temporal. Es decir, define el comportamiento de los nodos, pero no a los nodos en sí.

Por otro lado, antes comentábamos que las redes con las que lidiamos son sistemas markovianos, es esta una de las pruebas más claras, aunque en la Sección 4.8 se profundizará en ello. Por último, entiéndase como transiciones, o tránsitos, el paso del sistema del estado que presenta para un tiempo t, al que presenta para un tiempo $t + 1$.

Atractor

Las redes booleanas son simplificaciones de sistemas de ecuaciones diferenciales. Llegados a este punto, los atractores son a las redes booleanas lo que el régimen permanente al sistema de ecuaciones diferenciales. El estado que presenta el sistema para tiempo infinito. Se entiende por estado de la red el conjunto de valores que expresan las variables (nodos) que la componen para un tiempo dado. Por ejemplo. en el sistema de la Figura 3.1b, un estado sería el $W = 0, R = 1, S = 0$ (010) y otro el $W = 1, R = 1, S = 0$ (110).

De este modo, un atractor puede estar constituido por un único estado (y se dice que es *steady* o simple), o por varios (y estamos ante un *cyclic* o compuesto). Cuando la red entra en un atractor, da igual cuántos pasos de tiempo pasen, nunca va a salir del mismo. En el caso de entrar en un atractor simple, el sistema va a transitar desde y hacia ese estado eternamente. En el caso de entrar en un atractor compuesto, el sistema va a discurrir en bucle dentro del conjunto de estados que componen el atractor.

Es de resaltar que las redes booleanas siempre van a tener al menos un atractor. Los atractores compuestos son los responsables de este comportamiento porque, a diferencia de los sistemas de ecuaciones diferenciales, estas redes se componen de una serie discreta de estados posibles. Por lo

tanto, al final, la red siempre va a transitar hacia un estado que ya visitó con anterioridad dado que no puede generar nuevos estados. Esto implica que siempre se va a encontrar un atractor y, por lo tanto, siempre va a existir una variedad de régimen permanente.

En el caso extremo estaríamos hablando de un atractor compuesto, constituido a partir de todos los estados posibles de la red aunque, de nuevo, nos encontraríamos ante un atractor. Es más, en el caso concreto que nos atañe, redes booleanas de n variables, ese atractor lo habríamos de encontrar en un número menor o igual a $2^n - 1$ transiciones dado que, si asumimos que partimos de un estado inicial y la red consta de 2^n estados, después de $2^n - 1$ transiciones necesariamente habríamos de encontrar otro estado por el cuál ya habríamos transitado. Hay que destacar que estas soluciones, aunque en muchos casos no cueste demasiado extraerlas del grafo, no deben ser extraídas de él. Esto se debe a que los atractores han de ser una adaptación de datos experimentales, a menudo provenientes de estudios externos (Joo y col., 2018, Layek, 2012) empleados para validar la propia red.

Tabla de verdad

Toda función lógica se puede representar mediante lo que se conoce como tabla de verdad. En la Tabla 3.1 encontramos las tablas de verdad de las funciones de la Expresión 3.1. Esta representación consiste en agrupar cada una de las posibles combinaciones de los argumentos de las variables con su resultado asociado. Es especialmente útil dado que nos permite acceder a representaciones alternativas de las funciones como se expondrá más adelante.

Tabla 3.1: Tablas de verdad de las funciones de la Expresión 3.1.

x	y	x_{next}	y_{next}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

Support de una función

El *support* de una función es el conjunto formado por todas las combinaciones de valores posibles de los argumentos de la función, que hacen a la variable dependiente valer 1. Por ejemplo, los *support* de las funciones de la Expresión 3.1 se muestran en la Tabla 3.2.

Tabla 3.2: Tablas de verdad de las funciones de la Expresión 3.1.

x_{next}		y_{next}	
x	y	x	y
0	1	0	0
1	0	1	0
1	1	1	1

De esta forma, el *support* de la función x es el conjunto $\{(0,1),(1,0),(1,1)\}$ donde para cada tupla el primer valor se corresponde con la variable x y el segundo con la variable y. Análogamente, el *support* de la función y es $\{(0,0),(1,0),(1,1)\}$.

Maxitérminos y minitérminos

La tabla de verdad nos abre el camino a representaciones alternativas de las funciones lógicas, es decir, a la *conjunctive normal form*, o producto de sumas, y a la *disjunctive normal form*, o suma de productos. Estas dos representaciones se contruyen sobre lo que denominamos minitérminos (productos de variables de la función) y maxitérminos (sumas de variables de la función). Entiéndanse por sumas y productos lógicos, las operaciones OR y AND respectivamente.

De las dos, quizás la idea más intuitiva es la de suma de productos, también conocida como suma de minitérminos. Para obtener la suma de minitérminos de una función:

1. Acudimos a la tabla de verdad de la función.
2. Seleccionamos aquellas combinaciones de variables, filas de la tabla, que consiguen que la función valga 1 (el *support* de la función).
3. Expresamos cada combinación como un producto de las variables que la componen, si el valor de una variable es 0 ha de aparecer negada. Cada uno de estos productos recibe el nombre de minitérmino.
4. Hacemos un sumatorio lógico de todos los mineterminos, es decir, los encadenamos con operaciones OR.

Así, la formulación como suma de productos de x_{next} e y_{next} es como se describe en la Expresión 3.3. Donde por cada ecuación tenemos 3 minitérminos (resaltados entre paréntesis).

$$\begin{aligned} x_{next} &= (\neg x \wedge y) \vee (x \wedge \neg y) \vee (x \wedge y) \\ y_{next} &= (\neg x \wedge \neg y) \vee (x \wedge \neg y) \vee (x \wedge y) \end{aligned} \tag{3.3}$$

Análogamente, podemos expresar cada una de estas funciones mediante el producto de maxitérminos. El proceso es básicamente el inverso:

1. Acudimos a la tabla de verdad de la función.
2. Fijamos la atención en aquellas combinaciones de variables, filas de la tabla, que consiguen que la función valga 0.
3. Expresamos cada combinación como suma de las variables que la componen, si el valor de una variable es 1 ha de aparecer negada. Cada uno de estos sumatorios recibe el nombre de maxitérmino.
4. Hacemos un productorio lógico de todos los maxitérminos, es decir, los encadenamos con operaciones AND.

Las expresiones de x_{next} e y_{next} en producto de sumas son como se muestra en la Expresión 3.4, cada maxitérminos está resaltado en paréntesis.

$$\begin{aligned} x_{next} &= (x \vee y) \\ y_{next} &= (x \vee \neg y) \end{aligned} \tag{3.4}$$

Al ser expresiones muy sencillas, únicamente tenemos un maxitérmino en cada una de ellas. Un ejemplo de función cualquiera con muchos maxitérminos es el que se expone en la Expresión 3.5.

$$T = (A \vee B) \wedge (\neg T \vee A \vee B) \wedge (\neg A \vee \neg T \vee B) \tag{3.5}$$

Mapas de Karnaugh

Tanto la suma de productos como el producto de sumas son por sus propiedades formas muy prácticas de representar funciones lógicas. Por ejemplo, el producto de sumas se emplea a menudo para el diseño de algoritmos (Akutsu, 2018) y el producto de sumas tiene propiedades que lo hacen extremadamente cómodo para la resolución de conflictos biológicos, como se expondrá más adelante. No obstante, ninguna de estas representaciones destaca por ser compacta. Todo lo contrario, se caracterizan por ser formas innecesariamente extensas de visualizar funciones lógicas. Siempre y cuando se obtengan desde una tabla de verdad.

Como alternativa a las tablas de verdad existen los mapas de Karnaugh (Karnaugh, 1953). Una mapa de Karnaugh es una forma alternativa de organizar una tabla de verdad. Pongamos por ejemplo la función descrita en la Expresión 3.6. Su mapa de Karnaugh se muestra en la Tabla 3.3.

$$Q = (E \wedge \neg T) \vee \neg(F \wedge Q) \tag{3.6}$$

Tabla 3.3: Mapa de Karnaugh de la función descrita en la Expresión 3.6.

ET \ *FQ*	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	1
10	1	1	1	1

(a) Mapa sin los agrupamientos

ET \ *FQ*	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	1
10	1	1	1	1

(b) Mapa con los agrupamientos

La potencia de los mapas de Karnaugh radica en que una vez construidos podemos simplificar enormemente la función lógica, de tal forma que obtenemos la propia función como suma de productos (o producto de sumas) de la forma más compacta posible. Por lo general, mediante operaciones algebraicas es posible profundizar en la compactación de la propia expresión aunque nunca en su simplificación. Con el coste de abandonar las dos formas de representación expuestas anteriormente.

Persiguiendo la suma de productos agrupamos las casillas en el mapa con el valor de 1. Los agrupamiento siguen un conjunto de reglas.

1. El número de casillas que forman el grupo ha de ser potencia de 2 (1, 2, 4, 8, etc.).
2. Las agrupaciones deben formar rectángulos o cuadrados. No es posible realizar grupos disjuntos o en forma de L.
3. Las agrupaciones han de ser lo más grandes posible, aunque se solapen.

Con estas reglas se puede, sobre el mapa de la Tabla 3.3a, hacer 3 grandes agrupaciones: 1) todas las casillas bajo las columnas 00 y 01, 2) todas las casillas de la fila 10 y 3) todas las casillas bajo las columnas 10 y 00. El mapa de Karnaugh, cuando se acaba por un extremo comienza por el extremo opuesto, es por eso por lo que podemos agrupar las casillas de las columnas 10 y 00. Así, las agrupaciones obtenidas las podemos observar en la la Tabla 3.3b.

Se obtienen tantos minitérminos como grupos. En cada uno de ellos nos fijamos en los valores que etiquetan las casillas, los de las filas y columnas. Analizando todos esos valores solo hemos de quedarnos con aquellas variables cuya etiqueta asociada no varía. El minitérmino lo obtendremos de realizar el producto de las variables que no varían a lo largo de las etiquetas. Una vez más, si el valor conservado es un 0, la variable ha de aparecer negada en el minitérmino. Posteriormente sumamos (OR) los minitérminos. Así, la forma simplificada de la Expresión 3.6 se muestra en la Expresión 3.7.

$$Q = \overline{Q} \vee \overline{F} \vee E \wedge \overline{T} \tag{3.7}$$

La razón de realizar agrupaciones en rectángulos se basa en las posiciones de las etiquetas que identifican cada casilla en el mapa de Karnaugh. Cada casilla en el mapa es un minitérmino con todas las variables, en el ejemplo de la Tabla **??** la primera casilla es el minitérmino $E = 0, T = 0, F = 0, Q = 0$ (0000). Como podemos observar, entre dos etiquetas de fila o columna adyacentes únicamente cambia un 1 o 0, 1 bit. No siguen la numeración. Así si yo hiciera dos grupos adyacentes con las normas anteriores, pongamos todas las casillas bajo las columnas 00 y 11, obtendría los minitérminos $\neg F \wedge \neg Q$ y $\neg F \wedge Q$. Así, podemos simplificar para obtener:

$$\neg F \wedge \neg Q \vee \neg F \wedge Q = \neg F \wedge (\neg Q \vee Q) = \neg F$$

Que es el minitérmino asociado al grupo constituido por ambas columnas. Con esta colocación de las etiquetas, al realizar los grupos en forma de rectángulo estamos realizando implícitamente simplificaciones de este tipo. Es por esto que los grupos han de tener la forma descrita y han de ser lo más grande posible. Cuanto mayor sea el grupo, mayor será la simplificación y menor la expresión del minitérmino resultante.

Mediante el mismo procedimiento podemos construir productos de sumas. La única diferencia es que el criterio de valores pasa a ser el opuesto (agrupamos por 0 y si las etiquetas conservadas presentan un valor de 1 sus respectivas variables han de estar negadas). Así el producto de maxitérminos más simplificado es el de la Expresión 3.8.

$$Q = (E \vee \neg F \vee \neg Q) \wedge (\neg T \vee \neg F \vee \neg Q) \tag{3.8}$$

Por comodidad de uso, en vez de emplear tablas de verdad, a lo largo de este trabajo se han utilizado mapas de Karnaugh. Por lo tanto, mientras no se indique lo contrario, todo mapa es un mapa de Karnaugh.

3.2 Conceptos biológicos

Nodo

Como indica el título del proyecto, toda la teoría expuesta está aplicada a sistemas biológicos. Concretamente a redes genéticas. Por tanto, los nodos de las redes que empleamos se corresponden con genes. No obstante, cuantificar cómo de activo se encuentra un gen no es una tarea que se pueda abordar directamente. Dicha medición se efectúa a través de la concentración del metabolito codificado (o relacionado) en dicho gen.

En otras palabras, la actividad de un gen se mide por la concentración de aquellas moléculas que rige. En la mayor parte de casos estamos hablando de proteínas codificadas en el código del propio gen. No obstante, no siempre la relación es tan directa. A menudo la acción de un gen se refleja en el efecto de otras moléculas, no necesariamente proteínas, que toman partido en la regulación del metabolismo. Por ejemplo μRNA. En cualquier caso, la actividad de un gen va a ser medida a través de la concentración de alguna sustancia participante en el metabolismo celular. Es así como el valor 1 significa «alta concentración» y el valor 0 significa «baja concentración».

Pathway

En teoría de grafos un *pathway* es el conjunto integrado por una arista y un vértice que une, relaciona, dos elementos del grafo. Así lo que en Biología se conoce como ruta, un serie en cadena de reacciones químicas entre sustancias, es en teoría de grafos una sucesión de *pathways* consecutivos. Los *pathways* considerados en este trabajo establecen vínculos de activación o inhibición. Así los *pathways* son restricciones biológicas de la forma «A inhibe/activa a B». Donde A y B son genes.

Una pregunta pertinente es cuál es el significado de activación e inhibición. En un principio, la acción de inhibir implica disminuir la actividad de un gen mientras que la activación implica todo lo contrario. No obstante, lo único que se introduce en el modelo son concentraciones por lo que ambos roles son más complejos de lo que a priori pueda parecer.

En este trabajo, dado un gen B, hemos asumido como activador de B cualquier gen A cuya relación con B (*pathway*) implique cualquiera de los siguientes comportamientos: 1) a baja concentración de la molécula relacionada con A encontramos baja concentración de la relacionada con B o 2) a alta concentración de la molécula de A encontramos alta concentración de la de B. Por el contrario, un inhibidor sería cualquier gen A cuyo *pathway* con B impone cualquiera de los siguientes dos comportamientos: 1) a baja concentración de la sustancia asociada a A encontramos alta concentración de la asociada a B o 2) a alta concentración de la sustancia relacionada con A encontramos baja concentración de la relacionada con B. Aunque intuitivamente cada una de estas parejas de comportamientos se pueda resumir en uno solo, desde el punto de vista matemático no es posible. De ahí la diferenciación.

Gene Regulatory Network (GNR)

Una red de regulación genética (o *gene regulatory network*) es un conjunto de genes que directa o indirectamente se constituye en un sistema más o menos delimitado debido a sus relaciones. Es decir, conjuntos de genes con un gran número de dependencias entre sí. Es por eso que dichos genes se comportan como agentes integrados en un sistema a modelar.

En el contexto biológico, las redes booleanas son ampliamente empleadas para modelar *GRN*. Es este el motivo por el que a lo largo de este trabajo, cuando se habla de sistemas biológicos, en realidad se habla de *GRN*. Aunque el modelado de redes booleanas puede extrapolarse a sistemas diferentes de *GRN*, biológicos o no.

Fenotipo y genotipo

El genotipo de un individuo es la información genética del individuo, por ejemplo el gen EYCL1 (responsable del color de ojos azul y verde). Que puede presentar múltiples variantes. El fenotipo, es la manifestación observable del genotipo. En nuestro ejemplo, el color de ojos.

Marcador

En este trabajo se entiende por marcador una sustancia que caracteriza un estado del sistema, por su abundancia o ausencia. Es decir si la sustancia A fuera marcador del estado B, cuando el sistema se encuentre en el estado B, habrá una alta concentración (o baja según el marcador) del marcador B.

Capítulo 4

Modelo teórico

En este capítulo se expone la base teórica de todo el trabajo. En la propuesta se combinan diferentes aparatos matemáticos para dar lugar a una metodología cohesionada, robusta. Dada la novedad de nuestro enfoque, es necesario justificar la validez de los pilares sobre los que se levanta. De lo contrario, no sería válida su aplicación. Es en este apartado donde se lleva a cabo dicha justificación desde una base analítica, fundada en la bibliografía. En los anexos se pueden encontrar todas las demostraciones que en este apartado aparecen citadas.

La metodología consta de dos fases, a ser aplicadas en orden cronológico. Aunque complementarias, las dos constituyen en sí la base de múltiples trabajos y merecen ser introducidas por separado. Es esa la razón por la que se le dedican diferentes apartados. En conjunto, podemos hablar por un lado de las *nested canalizing boolean functions* (NCBF) y por otro de la Teoría de Conflictos expuesta en Layek, 2012.

4.1 Del grafo al modelo

Como se ha mencionado en apartados anteriores, nosotros perseguimos la generación del modelo de red a partir, exclusivamente, de un grafo. Es importante recalcar esto último. En ciencia y tecnología es común encontrarse con sistemas de ecuaciones diferenciales. Su relevancia abarca desde el control de procesos de fermentación al diseño de frenos hidráulicos y, por tanto, su presencia es palpable en multitud de disciplinas.

Esto es así debido a que una ecuación diferencial describe la relación entre 2 o más variables de interés. Por ejemplo la distancia y el tiempo. En ocasiones la relación no es directa y ha de ser representada a través de varias expresiones. Es entonces cuando aparecen los sistemas de ecuaciones diferenciales.

En este punto, ¿cuál es la relación entre grafos y sistemas de ecuaciones diferenciales? El concepto de grafo es muy amplio y consta de una extensa teoría detrás. De entre sus múltiples aplicaciones, un grafo dirigido puede ser la representación gráfica de un sistema dinámico. Así, un grafo es Internet, de la misma manera que lo es una red genética constituida por múltiples nodos, entes, en este caso genes o proteínas. Por tanto, es lógico pensar que si los sistemas de ecuacio-

nes diferenciales pueden ser representados por grafos, algunos grafos pueden ser representados mediante sistemas de ecuaciones diferenciales.

El problema del enfoque tradicional, basado en ecuaciones diferenciales, radica en la complejidad del sistema y, sobre todo, en su variabilidad. A partir de un determinado tamaño, es imposible representar la expresión del sistema y cuantificar los agentes que intervienen. No obstante, no olvidemos que aunque en la práctica un sistema de grandes dimensiones no es representable, en la teoría incluso un sistema de un millón de entes lo es.

Por tanto, el nuestro es un problema de representación. Así pues, hemos optado por cambiar nuestro vehículo de comunicación. De expresiones analíticas a expresiones lógicas. Un mismo sistema puede ser representado mediante una expresión analítica, tradicional, y una expresión lógica. La ventaja de las expresiones lógicas radica en su simplicidad, con la contrapartida de cierta pérdida de información con respecto a las expresiones analíticas. No obstante, los modelos lógicos han cosechado numerosos éxitos en los últimos años, y se han desarrollado numerosos trabajos en la materia (Joo y col., 2018, Zhou y col., 2016, Murrugarra y Dimitrova, 2015).

Sin embargo, ¿cómo es un modelo lógico? En este trabajo, un modelo (modelo lógico) es un conjunto de funciones lógicas que representan el comportamiento de los diferentes nodos que componen el sistema a modelar, la GRN. En estos modelos, el estado del sistema en tiempo $t+1$ depende únicamente del estado del sistema en tiempo t. Esto implica que las representaciones de los sistemas tratados en este trabajo son representaciones markovianas de un sistema dinámico, como se explica en la Sección 4.8.

4.2 El caso de ejemplo

Con la intención de estructurar la exposición de todos los procedimientos y teorías desarrolladas en este trabajo, hemos creado un ejemplo representativo de las redes que nos encontraremos en otros entornos. Presenta las relaciones que hemos hallado en otras publicaciones (Joo y col., 2018) y surge de adaptar el que encontramos en Layek, 2012.

El ejemplo consta del grafo que podemos observar en la Figura 4.1 y de los atractores que encontramos en la Tabla 4.1.

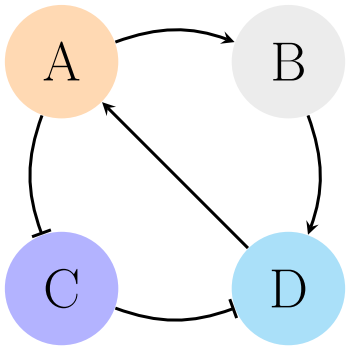


Figura 4.1: Grafo original de ejemplo. Una flecha acabada en punta indica una relación de activación mientras que una acabada con la forma -| indica una relación de inhibición. Por ejemplo, el nodo A activa al nodo B mientras que inhibe al nodo C.

Tabla 4.1: Atractores buscados dentro del grafo ejemplo. Cada una de las filas de esta tabla se corresponde con un atractor simple.

A	B	C	D
0	0	1	0
1	1	0	1

En este ejemplo, hemos supuesto cuatro nodos cuyas relaciones se describen en la Figura 4.1. Cada uno de los nodos de la red se corresponde con un gen, por lo que de ahora en adelante entiéndase al hablar de genes que nos referimos a nodos de la red y vice versa. Normalmente, la actividad o inactividad de un gen viene dada por una gran cantidad sintetizada de la proteína o metabolito que codifica dicho gen. Entiéndase por tanto, que cuando un gen esta activo es que existe evidencia de una gran cantidad sintetizada de dicho metabolito. Así pues, altos valores de concentración de proteína son representados por el valor de 1 en la red mientras que un bajo valor se representa con un valor de 0.

A lo largo de los siguientes apartados exponemos el algoritmo que esquematizamos al final del capítulo. Los ejemplos expuestos posteriormente se nutren del caso expuesto en este apartado.

4.2.1 Los conflictos dentro del grafo

Volvamos al caso de ejemplo, tomemos el grafo representado en la Figura 4.1. Pensemos por un momento que no se ha efectuado ningún tipo de simplificación, que nuestra representación es perfecta y que, por lo tanto, no solo la limitación de los valores a 0 y 1 sino que la formulación de las reglas que podemos extraer del propio grafo es absolutamente precisa. Podríamos concluir entonces que las expresiones de los distintos nodos de la red, conforme a la nomenclatura expuesta en el Capítulo 3, son las que aparecen en el Sistema 4.1 [1]:

$$\begin{aligned} A_{next} &= D \\ B_{next} &= A \\ C_{next} &= \neg A \\ D_{next} &= \neg C \wedge B \end{aligned} \tag{4.1}$$

A priori, no hay ningún problema. Es decir, según nuestro criterio, hemos construido una red que responde a lo expresado en el grafo y que cumple con los atractores fijados, nuestro criterio de validación. Por tanto, hemos conseguido una expresión válida de la red.

Esta afirmación, si bien es correcta, no incluye un pequeño matiz, ¿qué ocurriría si simultáneamente se activaran el nodo C y el B? ¿Se activaría D? Implícitamente hemos asumido que C ha de estar inactivo al mismo tiempo que B activo para que D se active. No obstante, esto no tiene por qué ser así. Aparece el problema de la prioridad en la activación, ¿qué nodo prevalece bajo

[1] Recordemos que, como se indica en el Capítulo 3, el símbolo = describe únicamente el comportamiento. Por ejemplo, en la primera ecuación se describe que cuando se active el nodo D, en el paso de tiempo siguiente se activará el nodo A. En ningún caso se está afirmando que el nodo A sea el nodo D.

qué condiciones? Para abordar esta cuestión en profundidad, por comodidad, descomponemos el grafo en los *pathways* que lo forman. Obtenemos la siguiente lista:

1. $A \xrightarrow{1:1,1} B$
2. $A \xrightarrow{1:1,0} C$
3. $B \xrightarrow{1:1,1} D$
4. $C \xrightarrow{1:1,0} D$
5. $D \xrightarrow{1:1,1} A$

Antes de continuar, hemos de resaltar que hasta ahora se ha definido que un *pathway* relaciona dos nodos. Así, por ejemplo, cuando se activa el primero, al paso de tiempo siguiente se activa el segundo. No obstante esto no ha de ser así. Podemos entender el *pathway* en un sentido amplio, de tal forma que cuando la primera expresión se muestra verdadera para tiempo t, en tiempo $t+1$ la segunda es cierta. En ambos casos, expresiones compuestas de múltiples elementos. Entonces, es inmediato percatarse de que realmente la noción tradicional no es más que una particularidad del caso general dado en la Expresión . Donde F y Q son expresiones lógicas, p pasos temporales, u el valor que ha de manifestar F en tiempo t y h el valor que ha de manifestar Q en tiempo $t+p$.

$$F \xrightarrow{p:u,h} Q \tag{4.2}$$

Todos los *pathways* anteriores son casos concretos de la expresión que se muestra en la Figura 4.2. En esta figura, F y Q son funciones lógicas. En los casos anteriores estas funciones estaban constituidas por una única variable, aunque esto no ha de ser siempre así, como se mostrará más adelante. Por otro lado, tenemos p. Es el número de pasos de tiempo que separan los valores que han de manifestar ambas expresiones, u y h. Es decir, si la expresión F en un tiempo t presenta el valor de u ($F = u$), en tiempo $t + p$ la expresión Q ha de manifestar h ($Q = h$). Un ejemplo de *pathway* alejado de la noción tradicional se describe en la Expresión 4.3. Mediante este nomenclatura es como representamos todas las expresiones lógicas a lo largo de este proyecto.

$$A \vee B \wedge C \xrightarrow{p:u,h} D \vee E \tag{4.3}$$

En este caso $F = A \vee B \wedge C$ y $Q = D \vee E$. Es esta noción generalizada de *pathway* la que explotamos en la Sección 4.5.1. Volviendo al conjunto de *pathways* que obtenemos del grafo, la restricción dada en el *pathway* $A \xrightarrow{1:1,1} B$ indica que si para un tiempo t existe alta concentración del metabolito codificado en el gen A ($A = 1$), necesariamente para un tiempo $t + 1$ ha de existir una alta concentración del metabolito codificado en el gen B ($B = 1$).

Centrémonos ahora en el nodo D e intentemos aplicar simultáneamente las condiciones representadas en los *pathways* 3 y 4. Podríamos considerar que son compatibles, ¿se solapan los *support* de las funciones que representan dichas condiciones? Más adelante profundizaremos mucho más en este concepto, aunque por el momento podríamos asumir que no. Tal asunción sería completamente falsa. Para ver con claridad el porqué es falsa, representemos la función booleana del nodo D mediante mapas de Karnaugh contemplando los *pathways* 3 y 4 por separado.

Como podemos observar en la Tabla 4.2, ambas regiones no son compatibles. En otras palabras, cada restricción intenta imponer un valor diferente sobre D, pero los *support* de las funciones

Tabla 4.2: Tablas con los mapas de Karnaugh de los *pathways* 3 y 4. En ambas tablas, la X significa un valor indeterminado de la función representada, en este caso D. Las regiones incompatbles entre ambos mapas han sido marcadas en rojo. Únicamente se encuentran representados los valores introducidos por los *pathways* 3 y 4.

AB \ CD	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

(a) *Pathway* 3

AB \ CD	00	01	11	10
00	X	X	0	0
01	X	X	0	0
11	X	X	0	0
10	X	X	0	0

(b) *Pathway* 4

sobre las que se construyen ambas restricciones (*pathways*) se solapan, tienen valores comunes [2]. Por tanto, en la representación espacial de la función, en esencia el mapa de Karnaugh, podemos apreciar que de aplicar con igual prioridad ambos *pathways* nos encontramos con que para idénticas combinaciones de variables, nuestro nodo D habría de presentar simultáneamente dos valores diferentes. O lo que es lo mismo, las regiones de ambas tablas, caracterizadas en rojo, no se pueden solapar. Por lo tanto, necesariamente tenemos que elegir entre una alternativa u otra. Implícitamente, como podemos observar en la Tabla 4.2, al elegir el producto de ambas nos decantamos por hacer prevalecer el *pathway* 4 sobre el 3 en la región de conflicto.

Tabla 4.3: Mapa del nodo D (Expresión 4.1) donde hemos priorizado el *pathway* 4 frente al 3 en la resolución del conflicto.

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	0	0
10	0	0	0	0

Llegados a este punto, ¿existe algún mecanismo que rija la resolución de estos conflictos? Es decir, la única forma de resolver estos conflictos es hacer combinaciones de funciones y esperar que cumplan la condición de los atractores, como el caso en el que nos encontramos, ¿o por el contrario existe algún procedimiento, alguna lógica interna de la que nos podamos aprovechar, para resolver estos conflictos de forma que las funciones obtenidas sean coherentes, no solo entre sí sino con los atractores? Esto restringiría enormemente la búsqueda y aceleraría el proceso de inferencia. Lo cual es de interés dado que no siempre vamos a tener 4 genes.

[2] El *pathway* 3 es de la forma $B \xrightarrow{1:1,1} D$, por tanto, la expresión que gobierna la condición es B. El *support* de B son las combinaciones de argumentos que se corresponden con los casillas con un 1 en la Tabla 4.2. En el *pathway* 4 ocurre lo mismo pero con las casillas marcadas con un 0.

La respuesta es que sí, sí que existe dicho mecanismo. Aunque antes es necesario introducir otro concepto.

4.3 Una forma eficiente de generar redes

Antes de continuar con el ejemplo, tenemos que introducir un nuevo actor, las funciones booleanas canalizantes anidadas (*nested canalizing boolean functions*, NCBF). En secciones anteriores nos encargamos de acotar la combinatoria mediante la inferencia de información, los conflictos, de forma que obtenemos nuevas restricciones que imponer sobre las funciones empleadas. Es decir, desechamos funciones que no son coherentes con la propia naturaleza del grafo y, por lo tanto, no lo van a ser con la de los atractores. Aún así, la cantidad de funciones que pudiéramos llegar a analizar escapa con mucho de lo que un ordenador convencional puede manejar en un tiempo razonable. Esto es así dado que en ningún momento hemos de olvidar que nuestro ejemplo consta únicamente de 4 variables, lo que dista con mucho de un caso real.

Ha llegado el momento de formular otra pregunta, ¿las funciones booleanas que rigen procesos biológicos presentan alguna estructura en particular? De ser cierto, no sería el único caso con ejemplos como la función de Hill o la proporción áurea ampliamente documentados. La respuesta es que sí, concretamente, los nodos de redes biológicas tienden a manifestar funciones catalogadas como NCBF (Joo y col., 2018). Entonces, ¿qué es una NCBF?

4.3.1 Funciones booleanas canalizantes anidadas (NCBF)

Para introducir este concepto tomemos una vez más una función cualquiera (no necesariamente lógica), la Expresión 4.4. Esta función depende de 3 variables: *P, U* y *V*. En *f*, la variable *P* es esencial, lo que significa que existe un valor p tal que $f(P = p, U, V) = q$. Independientemente del valor de U y V. Es decir, si $P = p$, $Q = q$ sin importar los valores del resto de variables no esenciales.

$$Q = f(P, U, V) \tag{4.4}$$

En *f*, p recibe el nombre de valor canalizante y q el de valor canalizado. Una función se dice que es canalizante anidada cuando todas sus variables son esenciales. Y si esta función es lógica, entonces estamos ante una NCBF. En otras palabras, una NCBF no es otra cosa que una función que se comporta como la de la Expresión 4.5.

$$\begin{aligned} f(x_1 = a_1, ...) &= b_1 \\ f(x_1 \neq a_1, x_2 = a_2, ...) &= b_2 \\ &... \\ f(x_1 \neq a_1, x_2 \neq a_2, ..., x_n = a_n) &= b_n \end{aligned} \tag{4.5}$$

Donde f es una función lógica con variables $x_1, x_2, ..., x_n$, todas ellas esenciales cuyos valores canalizantes son $a_1, a_2, ..., a_n$ con valores canalizados $b_1, b_2, ..., b_n$. No todas las variables canalizantes tienen la misma prioridad, en el caso de contradicción una se impone. Por eso, en la Expresión 4.5, como las variables ya están ordenadas por prioridad decreciente, si $x_1 = a_1$, $f = b_1$ indepen-

dientemente del resto variables. Así, este comportamiento se repite con cada una de ellas y por eso no representamos en la expresión anterior las variables que se encuentran por debajo en el juego de prioridades. Porque su valor es irrelevante. Así mismo, con las mismas variables esenciales, cambiando las prioridades, podemos obtener diferentes expresiones.

Es esta la noción intuitiva de NCBF. Estas funciones, introducidas por Kauffman y col., 2003, Kauffman y col., 2004 se han empleado ampliamente en modelado de circuitos genéticos entre otros procesos biológicos. Sus excepcionales resultados en este tipo de modelado han llevado a la aceptación de que, en efecto, los procesos biológicos tienden a ser controlados por esta variedad de funciones (Joo y col., 2018). Por otro lado, el análisis que ofrece Li y col., 2013 aporta una visión extraordinaria no solo de las NCBF sino de múltiples herramientas para su manejo. Herramientas que hemos empleado en este trabajo. Especialmente la definición que proporciona:

NCBF es toda función que tiene una única expresión bajo la forma:

$$f(x_1, x_2, ...x_n) = M_1(M_2(...(M_{r-1}(M_r \oplus 1) \oplus 1)... \oplus 1) \oplus b \tag{4.6}$$

Donde $\oplus$ simboliza la operación de módulo 2 y se cumple que $M_i = \prod_{j=1}^{k_i}(x_{i_j} \oplus a_{i_j})$ (Li y col., 2013). La notación con álgebra de módulo 2 es un procedimiento equivalente a formular expresiones lógicas con operaciones convencionales. No obstante, no es la nomenclatura empleada en este trabajo. Esta notación es importante porque es empleada en la Expresión 4.6, que sí es ampliamente utilizada a lo largo de este proyecto. El lector puede comprobar la veracidad de esta afirmación tratando cada una de las variables en la Expresión 4.6 como lo que son, variables lógicas.

La idea que subyace en la Expresión 4.6 es simplemente que cuando todas las variables son esenciales, o bien todas tienen el mismo valor canalizado, han de estar agrupadas en capas de distinta prioridad (M_i) en base al valor canalizado que persiguen. La estructuración de dichas variables ha de ser como la que observamos en la Expresión 4.5, cuya representación general es la Expresión 4.6. Otra de las grandes ideas que se ocultan tras esta expresión es que cada una de las representaciones es única (Li y col., 2013). En otras palabras, podemos agrupar las variables esenciales en capas de múltiples estructuras con múltiples combinaciones de valores canalizantes y canalizados. Y no va a haber dos funciones iguales. Lo que conceptualmente es muy interesante desde el punto de vista biológico como indicaremos más adelante.

Independientemente de lo comentado en el párrafo anterior, desde el punto de vista matemático, la Expresión 4.6 nos brinda una serie de propiedades de gran utilidad. El algoritmo que mostramos a continuación se construye enteramente bajo dichas propiedades.

4.3.2 Algoritmo para la generación de NCBF

En el apartado anterior introducimos el concepto de que las funciones booleanas responsables de procesos biológicos se concentran dentro de la NCBF. Por lo tanto, si centramos nuestra búsqueda en las NCBF tendremos más probabilidades de hallar la expresión que buscamos. Consiguiendo de esta manera reducir el tiempo de búsqueda significativamente. Llegados a este punto solo necesitamos un mecanismo eficiente para generar todas las posibles NCBF. Dicho proceso, tal como ha sido diseñado en este proyecto consta de dos fases:

1. Obtención de un ejemplo de cada una de las posibles familias de NCBF que podemos expresar a partir de nuestro grafo.
2. Obtención del resto de NCBF de cada una de las familias por combinatoria a partir de los ejemplos.

El segundo paso no entraña gran complejidad debido a que al final nos encontramos trabajando con cadenas de caracteres (strings) con una estructura fija, por lo que es un proceso de combinatoria basado en librerías existentes, ya optimizado y documentado. El reto reside en el primero.

Antes de abordar el primer paso, es conveniente percibir que a partir de la estructura de la Ecuación 4.6, podemos deducir múltiples simplificaciones para nuestro problema:

1. El valor para a_{i_j}, independientemente de cuál sea la variable, siempre va a ser 1, dado que suponemos que un nodo ejerce su acción cuando se activa. Tal asunción, aunque no es cierta en muchos casos, no es importante. Lo importante es fijar un valor, 0 o 1, para conseguir el algoritmo que nos permita generar los ejemplos del paso 1 rápidamente. Una vez tengamos los ejemplos, obtener las variantes con valor canalizado 0 es muy rápido desde el punto de vista computacional, gracias a librerías dedicadas.
2. La operación módulo 2 con la unidad ($\oplus 1$) en lógica bivaluada equivale a una negación sobre el miembro izquierdo de la expresión. Por ejemplo, $0 \oplus 1 = 1$ y $11 = 1$.
3. Todas las variables posicionadas dentro de un mismo M_i provocan el mismo valor (valor canalizado) b_i en f. Esto es porque de presentar cualquiera de ellas su valor canalizante, van a lograr la misma operación: $0 \oplus b = b$. Dado que cada capa tiene su propio b, inferido a partir del b más externo gracias a las negaciones anidadas, esto es válido para cualquier capa. Por consiguiente, todas las variables de la misma capa tiene el mismo valor canalizado.
4. El valor canalizado (b_i) de las variables de una capa M_i y el de las variables de la capa siguiente M_{i+1} ha de ser diferente, lo que implica en lógica bivaluada que el primero ha de ser 0 y el segundo 1 o vice versa. Esto se deduce a partir del principio 2. Tomemos dos capas intermedias, al azar: $M_x(M_y(...)\oplus 1)\oplus 1$. Si $M_x = 1$, tenemos que $(M_y(...)\oplus 1)\oplus 1 = M_y(...) = M_y(...)\oplus 0$. Si tomamos $M_x(M_y(...)\oplus 1)\oplus 0$ y $M_x = 1$, obtenemos $(M_y(...)\oplus 1)\oplus 0 = M_y(...)\oplus 1$. Por tanto, los valores canalizados, de facto, consiguen un estructura de negaciones anidadas por capa. Así, necesariamente, el valor canalizado de una capa y el valor canalizado de la siguiente (dado que hay una negación en el medio), ha de ser diferente.
5. De los puntos anteriores deducimos que el valor b de la expresión queda marcado por las variables pertenecientes a la capa más externa (M_1). Así mismo, no es necesario que nos preocupemos por los valores canalizados de las variables ubicadas en las capas internas, las negaciones (operaciones de suma de la unidad con módulo 2) cambian automáticamente el resultado pretendido, alternante entre 0 y 1. Por lo que a la hora de elegir el valor de b para construir la expresión, basta con conocer si los nodos de la primera capa son activadores o inhibidores. Si tienen valor canalizado 1 o 0.

Es de estos puntos de donde concluimos que no es necesario obtener un algoritmo para generar todas las posibles NCBF, basta con obtener un algoritmo capaz de conseguir un ejemplo de todas las posibles estructuras que puede manifestar nuestra función con los datos de los que partimos,

los datos del grafo extendido a partir de los conflictos. Lo demás, tanto las posibles estructuras (combinaciones de números) como las posibles NCBF (combinaciones de caracteres), es fácil de conseguir. Así pues, el algoritmo que consigue un ejemplo de todas las posibles estructuras de NCBF a partir de un conjunto de datos consta de varios puntos.

Para exponer toda la potencia del algoritmo, nos vamos a alejar del ejemplo original. Vamos a suponer que queremos obtener un ejemplo de todas las posibles NCBF que describen un nodo X caracterizado por tener 4 activadores: A, B, C y D, y 3 inhibidores: E, F y G. De la misma manera, suponemos que se nos han dado un vector de enteros como el siguiente:

$$[1, 1, 1, 1, 2, 1] \tag{4.7}$$

Este vector de enteros representa la estructura de NCBF que vamos a obtener con la ejecución del algoritmo. Es decir, representa el número de nodos que hemos de poner en cada capa, en cada M_i. Conforme al vector, en M_1, M_3 y M_5 (capas 1, 3 y 5) ha de haber 1, 1 y 2 activadores respectivamente. Igualmente, en M_2, M_4 y M_6 (capas 2, 4 y 6) ha de haber 1 inhibidor. Así mismo, las capas de activadores e inhibidores se deducen por la suma de los nodos, es decir, las posiciones 1, 3 y 5 han de ser activadores porque hay 4 activadores y entre las 3 posiciones demandan 4 variables. Lo mismo para los inhibidores. Por otro lado, por la propia naturaleza de la Expresión 4.6, sabemos que siempre se alternan activadores e inhibidores. Así, dado el vector numérico, es posible deducir la estructura.

En caso de que los grupos de activadores e inhibidores fueran intercambiables, por la naturaleza del vector y por tener el mismo número de elementos, seleccionaríamos al azar el grupo de la primera capa. No se produciría ningún problema, dado que la estructura alternativa se generaría con las librerías por combinatoria en pasos posteriores a partir de la estructura obtenida.

Procedemos paso a paso con la explicación del algoritmo.

1. Dibujamos el diagrama de árbol tanto para activadores como para inhibidores. Como se muestra a continuación. Para generar el árbol partimos del grupo del nivel superior, el que contiene todos los nodos del conjunto, y vamos haciendo subgrupos partiendo de este. De tal forma que en el nivel inmediatamente inferior encontramos todos los subgrupos posibles con un nodo menos, en este caso 3. Repetimos la operación con los nuevos subgrupos obtenidos hasta llegar al nivel más bajo (1). Es importante destacar que en un nivel no pueden existir subgrupos repetidos. Así mismo, es importante la numeración de los niveles, realizada de arriba a abajo. Si prestamos atención observaremos que coincide en cada nivel con el número de nodos que contienen los subgrupos de ese mismo nivel.

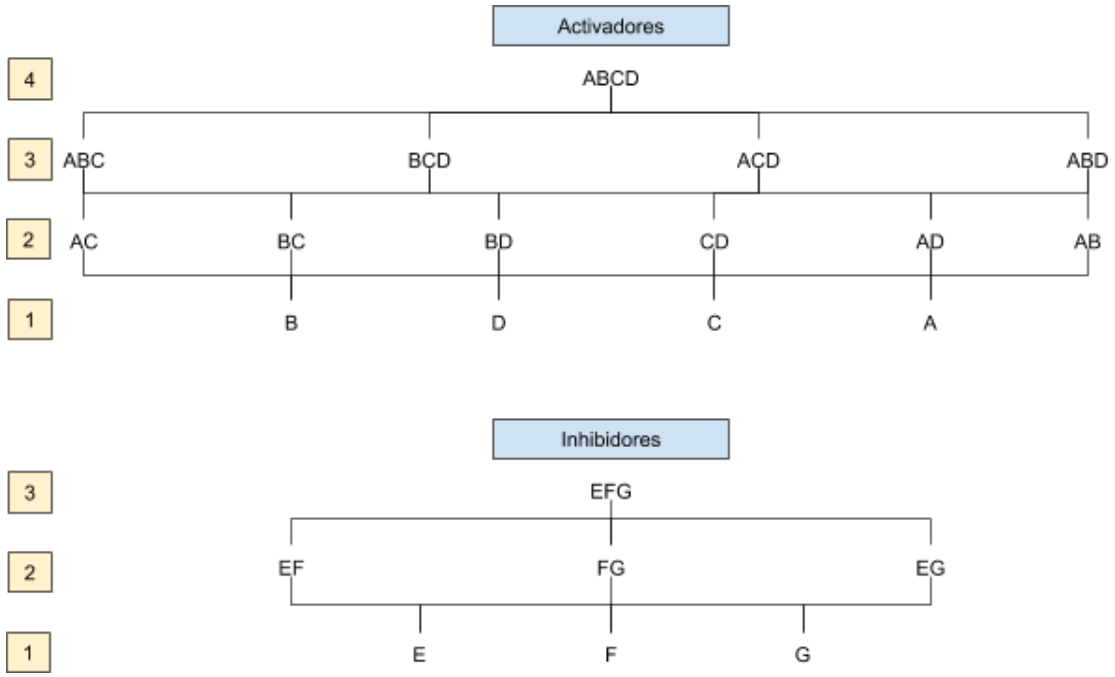


Figura 4.2: Diagrama de árbol de los activadores y de los inhibidores para el ejemplo de la explicación del algoritmo.

2. Cogemos el primer entero del vector y seleccionamos el árbol cuya estructura de niveles es compatible con el vector de enteros, en este caso el árbol de activadores. En realidad solo necesitamos saber de qué árbol hay que seleccionar el primer nodo. Nos fijamos en el nivel cuya numeración se corresponde con el valor del entero. Tomamos un subgrupo al azar. En este caso tomamos el subgrupo "D", perteneciente al nivel 1 (primer entero del vector).

3. En este punto tenemos que introducir el concepto de palabra, es decir, el número de letras, nodos, que hemos seleccionado. En este caso nuestra palabra tiene 1 letra, lo que hemos llamado peso acumulado de 1 ($P_A = 1$). Así mismo, calculo el número de letras de este árbol que me quedan por seleccionar (lo que hemos llamado internamente como variabilidad del árbol), $\Delta E = 3$ porque llevo recogidas 1 de 4 letras de este árbol. Calculo el peso total ($P_T = 7$), valor que va a ser constante durante el resto de pasos del algoritmo y que se corresponde con la suma de todas las entradas del vector de enteros. Calculo el peso complementario, (P_C) con la fórmula $P_C = P_T - P_A - \Delta E$, que para el primer paso es $P_C = 3$. P_C se corresponde con la variabilidad del otro árbol, en el que no nos encontramos durante este paso. En otras versiones del algoritmo, la selección del tamaño de los fragmentos que incorporábamos era completamente aleatoria, de forma que P_C actuaba como valor limitante para dichos fragmentos. No obstante, en la versión final, de cara a controlar la generación de redes, imponemos la estructura con el vector de enteros, por lo que P_C a quedado únicamente como criterio de parada.

4. Acto seguido, nos vamos al otro árbol. Seleccionamos el nivel que nos indica el siguiente valor del vector de enteros y repetimos la operación. Calculando los distintos parámetros.

5. Repetimos los pasos 3 y 4 hasta que los valores obtenidos de ΔE y P_C sean 0. Momento en el que dejamos de introducir fragmentos.

La selección del subgrupo en sí es aleatoria, no es importante, un resultado de una estructura que podríamos obtener es el que se expone en la Tabla 4.4.

Tabla 4.4: Tabla con un ejemplo de ejecución del algoritmo para el vector de enteros 4.7.

Parámetros \ Fragmentos	D	E	A	F	BC	G
P_T	7	7	7	7	7	7
P_A	1	2	3	4	6	7
ΔE	3	2	2	1	0	0
P_C	3	3	2	2	1	0

Es fácil percatarse de que cada uno de los fragmentos se corresponde con las agrupaciones de variables canalizantes que dispondremos en cada una de las capas de la NCBF, en cada M_i. Es así como obtenemos los ejemplos. Una vez obtenido un ejemplo de esta naturaleza, solo tenemos que realizar combinatoria sobre los caracteres para obtener todas las funciones de la familia. Así mismo, el problema de obtener el vector de enteros es similar. Como se indicaba anteriormente, ambos problemas de combinatoria son despreciables desde el punto de vista computacional por ser operaciones de muy bajo coste. De esta forma, obtenemos los vectores de enteros, para cada uno de ellos lanzamos el algoritmo y luego generamos el resto de vectores similares una vez que hemos obtenido todos los ejemplos posibles.

Como otras secciones de este proyecto que se expondrán más adelante, este algoritmo es un contenido *original* del trabajo. Es decir, no ha sido extraído de ninguna fuente externa ni se encuentra dentro de ninguna publicación de la que el autor tenga constancia.

4.4 Procedimiento para la resolución de conflictos

Una vez introducidas las NCBF, podemos retomar los conflictos. Las ideas de conflictos expresadas hasta ahora en este trabajo han sido tomadas de Layek, 2012. De forma resumida, la metodología es la siguiente.

4.4.1 Base teórica para la resolución de conflictos

Tomemos de nuevo los dos *pathways* representados en la Tabla 4.2. Supongamos que el mapa de la derecha ha surgido de combinar diferentes *pathways*, de tal forma que podemos decir que es el mapa de Karnaugh de una función parcialmente definida, digamos de un nodo Q. En este contexto, pretendemos imponer el *pathway* de la izquierda a la función de la derecha. Como apuntábamos la vez anterior, existe una región donde el solapamiento es imposible, la marcada en rojo en la Tabla 4.2. Esa región se caracteriza por los valores de las variables, las casillas del mapa, de $\{0111, 0110, 1111, 1110\}$ para las variables *A, B, C* y *D* respectivamente. O lo que es lo mismo, si resolviéramos el conflicto de forma idéntica, el mapa de la función Q sería como se representa en la Tabla 4.5.

De ahora en adelante, la región que marcábamos en rojo en los mapas de la Tabla 4.2 va a ser representada por una función Ψ, la función problema Recibe su nombre porque se corresponde exclusivamente *con la región del mapa que no podemos solapar*. Por otro lado, su acción sobre el

Tabla 4.5: Mapa del nodo Q.

AB \ CD	00	01	11	10
00	X	X	0	0
01	1	1	0	0
11	1	1	0	0
10	X	X	0	0

nodo, en este caso Q, equivale a un *pathway* de la forma $\Psi \xrightarrow{1:1,\bar{b}} Q$. Así mismo, el *pathway* que se opone a Ψ, Y, (que en el caso anterior se corresponde con el *pathway 3*) lo denotamos como $Y \xrightarrow{1:1,b} Q$. Así, para el ejemplo del nodo Q que se viene desarrollando, $b = 1$ y las expresiones de Ψ y Y son como se muestran en la Expresión 4.8.

$$\begin{aligned} \Psi &= B \wedge C \\ Y &= B \end{aligned} \tag{4.8}$$

Estamos por tanto ante un conflicto entre *pathways*. Llegados a este punto se nos presentan dos opciones diferentes para resolver el conflicto:

1. Asumimos que Y tiene mayor prioridad que Ψ, $(Y > \Psi)$. En este caso, alteramos el *pathway* de Ψ, de tal forma que $\Psi \xrightarrow{2:1,0} Q$. El resultado es el que podemos apreciar en la Tabla 4.5. ¿Pero qué significa $\Psi \xrightarrow{2:1,0} Q$? Significa que las casillas pertenecientes al *suppport* de Ψ no alcanzarán el valor de 0 en un paso temporal de 1 sino de 2, como indica la Tabla 4.5, en el primer paso temporal alcanzarán el valor de 1, que es el que impone Y. Por tanto, para que alcancen el valor de 0 en 2 pasos temporales, es necesario introducir otro pathway adicional:$\Psi \xrightarrow{1:1,1} S(\bar{b})$. Esta notación, extraída de Layek, 2012, nos dice que, como hemos impuesto un valor de 1 (b), vamos a activar en un paso temporal todas aquellas regiones de nuestro mapa parcial (S) que conseguirán que en un paso temporal extra alcancemos el valor de $\bar{b}$. De esta forma alcanzamos el valor de 0 $(\bar{b})$ en 2 pasos temporales como se pretende.

2. La otra alternativa que se nos presenta es aceptar que Ψ tiene mayor prioridad que Y, $(\Psi > Y)$. El mismo conflicto de antes quedaría como se representa en la Tabla 4.6. Como en el punto anterior, la clave consiste en formular un rodeo, de manera que en este caso $Y \xrightarrow{2:1,1} Q$. Para lograrlo, el nuevo *pathway* que introducimos es de la forma $Y \xrightarrow{1:1,1} \overline{\Psi}$. Lo que conceptualmente tiene sentido, es decir, que la región conflictiva del *support* de Y, en lugar de activar $S(\bar{b})$, que active $\overline{\Psi}Y$ (la región de Y no conflictiva que consigue el resultado deseado). Siempre estamos ante la misma idea: *si no es posible lograr el resultado deseado, se ha de activar la región que puede lograr dicho resultado en el paso de tiempo siguiente.* Hay que destacar que este procedimiento de solución se podrá aplicar exclusivamente cuando $supp(\overline{\Psi}) \cap supp(Y) \neq \varnothing$. En los casos en los que $supp(\overline{\Psi}) \cap supp(Y) = \varnothing$ tendremos que recurrir al primer procedimiento de resolución.

El procedimiento expuesto en Layek, 2012, consiste en aplicar iterativamente estas dos soluciones sobre todos los mapas correspondientes a los nodos de la red hasta alcanzar un conjunto

Tabla 4.6: Mapa del nodo Q.

AB \ CD	00	01	11	10
00	X	X	0	0
01	1	1	1	1
11	1	1	1	1
10	X	X	0	0

de mapas, con los *pathways* ya introducidos, en el que no encontramos conflicto alguno entre los *pathways* ¿Este procedimiento siempre nos va a conseguir una solución? La respuesta es que no. En el momento en que entremos en un bucle de aparición de los mismos conflictos, o que en algún momento $S(\bar{b}) = \varnothing$, podemos concluir que el problema es irresoluble y, por tanto, revisar la información de la que disponemos. No obstante, si aplicamos esta metodología a nuestro caso de ejemplo, uno de los resultados que obtendríamos es el que se muestra en la Figura 4.7.

Tabla 4.7: Mapas de Karnaugh de los genes. Obtenidos a partir de la resolución de los conflictos del grafo de la Figura 4.1.

AB \ CD	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	1	1	1
10	X	1	1	X

(a) Mapa del nodo A.

AB \ CD	00	01	11	10
00	X	X	X	X
01	X	X	0	0
11	1	1	0	0
10	1	1	1	1

(b) Mapa del nodo B.

AB \ CD	00	01	11	10
00	X	X	X	X
01	X	X	1	1
11	0	0	1	1
10	0	0	0	0

(c) Mapa del nodo C.

AB \ CD	00	01	11	10
00	X	X	0	0
01	1	1	1	1
11	!	1	1	1
10	X	X	0	0

(d) Mapa del nodo D.

No obstante, este procedimiento tiene tres grandes problemas.

1. Partiendo de un conjunto de *pathways*, *¿en qué orden he de aplicar cada uno de ellos sobre el mapa del nodo?* Como se ha expuesto en lineas anteriores, nos encontramos ante conflictos entre *pathways* concretos, el orden en el que sean aplicados determinará la naturaleza de Ψ y, por tanto, la naturaleza del *pathway* que vayamos a imponer. Dado que no siempre será posible aplicar los dos mecanismos de resolución, el orden de aplicación condiciona en gran medida el resultado final. En Layek, 2012 no se especifica ningún orden.
2. Por otro lado, estos conflictos surgen de la interacción entre pathways, impuestos por nodos o conjuntos de nodos. *¿Cuál es el criterio que determina cuál es el pathway que se ha de imponer en el conflicto?* En Layek, 2012 se propone un criterio, estrictamente matemático, que no consigue resolver satisfactoriamente el problema. En la práctica, cuando la jerarquía de prioridades no sirve para resolver los conflictos, se recurre a simplificaciones previa consulta a expertos.

3. Finalmente, en opinión de quien redacta este trabajo final de grado, estamos ante *un procedimiento subóptimo*, ¿por qué? Porque si todo se trata de un juego de valores en los nodos, de conseguir que estos impongan en un tiempo dado unos valores sobre otros nodos, al partir de un mapa parcialmente en blanco, indudablemente estamos eludiendo valores, en un primer momento desconocidos, que podrían satisfacer cualquiera de las restricciones impuestas por el procedimiento. De esta forma, podríamos encontrar redes capaces de satisfacer las restricciones del problema, a las que sería imposible llegar sin contar con estos valores desconocidos.

4.4.2 Un procedimiento alternativo para la resolución de conflictos

En el contexto de este proyecto, se ha trabajado para conseguir un procedimiento para la resolución de conflictos que pudiera superar estos tres grandes obstáculos. Es por ello que los hemos abordado individualmente.

Orden de aplicación de los conflictos

El primer escollo para solucionar el problema del orden de aplicación de los conflictos son los propios roles de Y y Ψ. Es decir, hasta ahora, siguiendo la teoría descrita en apartados anteriores, cogemos un *pathway*, no importa cuál, y lo aplicamos sobre nuestro mapa. Cogemos el siguiente, comprobamos que no hay conflicto con lo anterior y lo aplicamos, y así sucesivamente. Dependiendo de cuál se aplicó con anterioridad, Ψ manifestará una u otra naturaleza. Por lo tanto, tenemos que eliminar esos roles. El objetivo es claro, nos debe ser indiferente aplicar primero un *pathway* u otro. Para ello desplazamos el objeto de trabajo, del mapa de Karnaugh (representación de la función que controla el gen) al paquete de *pathways* (representación de todas las influencias que actúan sobre el gen). Así, el objetivo ya no es obtener un mapa sobre el que hayan sido aplicados todos los *pathways*, y resueltos satisfactoriamente todos los conflictos. El objetivo pasa a ser obtener una batería de *pathways* que puedan ser aplicados simultáneamente sin que aparezcan conflictos.

Ambos planteamientos son matemáticamente equivalentes, ya que cuando resolvemos un conflicto sobre el mapa entre dos *pathways* Y y P, cuando priorizamos uno, digamos $P > Y$, el procedimiento de resolución sobre el mapa es equivalente a que modifiquemos el *pathway* no priorizado, de tal forma que Y pasa a ser $Y\overline{\Psi}$. Para posteriormente aplicar los pathways P y $Y\overline{\Psi}$, ya sin conflicto ninguno. Por lo que a priori podemos desplazar el foco de atención sin alterar la validez del planteamiento. No obstante, por mucho que decidamos aplicar todos los *pathways* simultáneamente, de aparecer conflictos entre dos conjuntos de *pathways*, indefectiblemente habremos de agrupar por parejas y resolver. Por lo que una vez más estamos aplicando y resolviendo por parejas.

Esto es, sin duda, cierto. No obstante, supone un gran avance por dos motivos. El primero de ellos es que es, efectivamente, entre parejas, no entre un *pathway* por un lado y el conjunto del mapa, en resumen un conjunto de pathways, por el otro. Por lo que la dependencia temporal se produce entre las parejas y lo que surja de la resolución de sus conflictos. En otras palabras, todos los *pathways* han sido aplicados al mismo tiempo, no el *pathway* enésimo ha sido aplicado en tiempo n sobre los resultados de los $n-1$ conflictos anteriores.

Por otro lado, la comparación, al producirse *pathway* a *pathway* elimina completamente el rol de Ψ, dado que ya no estamos trabajando sobre un conjunto de condiciones previamente impuestas. Únicamente tenemos el *pathway* Y y el P. Ahora, alejados ya de los mapas, y eliminados los roles de Y y Ψ. ¿Cómo podemos resolver el conflicto? Como se describía líneas atrás, al final es un *juego de valores*, por lo tanto, desde ese planteamiento la respuesta es sencilla. Únicamente existe un mecanismo de resolución, la imposición del siguiente *pathway* descrito en la Expresión 4.9.

$$\Psi \xrightarrow{1:1,1} S(\overline{b}) \tag{4.9}$$

El razonamiento que detrás de la solución de la Expresión 4.9 es el siguiente. El objetivo de un *pathway* consiste en provocar una respuesta en un nodo o expresión (consecuente), si se dan unas condiciones determinadas (antecedente). Por tanto, lo importante no es cómo conseguimos que el consecuente manifieste el comportamiento, si se cumple el antecedente. Es decir, lo importante es el valor del consecuente en sí. Es intuición de quien escribe estas palabras, que era este el planteamiento de Layek, 2012, tal vez, la demostración más clara sea el primero de los mecanismos de resolución, que deriva en exactamente esta expresión. Entonces, ¿a qué se debe la existencia del segundo mecanismo de resolución, cuando $\Psi > Y$?

Una vez más, desconocemos lo que tenía en mente el creador de dicha teoría, aunque es plausible que pretendiera suministrar un procedimiento alternativo, en caso de que el valor que consiguiera el primero no fuera satisfactorio, partiendo de un conjunto de restricciones biológicas. No obstante, lo más llamativo es la solución que propone, que en esencia significa *redirigir todo el pathway a la región de aplicación del mismo pathway fuera de la región problema, de* Ψ. En consecuencia, es mi parecer que el criterio general es imponer el valor satisfactorio. Así pues, el procedimiento de resolución ha de ser único, dado que el objetivo es único: lograr el valor adecuado en el consecuente. Un valor caracterizado en la Expresión 4.9 por b.

De esta forma, todos los conflictos se resuelven de forma idéntica: 1) el *pathway* no priorizado pasa de ser Y a ser $\overline{\Psi}Y$ y 2) introducimos el *pathway* de la Expresión 4.9. En un principio, es esto lo que buscábamos, una única forma de proceder entre dos *pathways* sin que a ninguno de ellos le asignemos el rol de Ψ. Por lo tanto, la dependencia con respecto el orden de aplicación queda completamente eliminada. El único inconveniente para implementar este procedimiento radica en que no conocemos S dado que ahora no tenemos ningún mapa porque cada pareja de *pathways* se está resolviendo de forma independiente . Así pues, por el momento, supongamos que tenemos un mapa, que conocemos el valor de S. Esta afirmación, por el momento falsa, se aclarará posteriormente. En este punto, simplemente, asumamos que es cierta. De esta forma, el problema del orden de aplicación ha sido resuelto.

El criterio de prioridad

Anteriormente hemos introducido que, dado un conflicto entre dos *pathways*, siempre hemos de priorizar uno u otro. Como se ha expuesto anteriormente, el mecanismo de base matemática recogido en Layek, 2012 no consigue resolver el problema de la prioridad, al mismo tiempo que es nuestra intención evitar las simplificaciones que abundan en la bibliografía (Robeva y col., 2013). En este momento es pertinente realizar una pequeña digresión.

En las últimas décadas se ha desarrollado ampliamente la obtención computacional de filogenias a partir de bases de datos genéticos (entre otras). Esta, una tarea típicamente bioinformática, ha cobrado especial relevancia en los últimos meses a raíz del estallido del SARS-CoV-2, al ser una de las fuentes de evidencia que permiten relacionarlo con otras variedades de coronavirus como el MERS-CoV. Realizar una filogenia es, en esencia, resolver dos problemas matemáticos claramente definidos: el alineamiento de secuencias y la inferencia del árbol. Sin entrar en la definición completa del problema, alinear dos secuencias consiste en desplazar una de ellas de manera que la correspondencia, a nivel local o global, entre ambas cadenas (de caracteres) se maximice. A priori, dos cadenas se van a alinear mejor cuanto menor sea la distancia entre ellas. Entiéndase por distancia la distancia de Levenshtein, entre otras métricas. En esencia, la distancia es el número de transformaciones que ha de sufrir una cadena para transformarse en la otra. En términos genéticos, cuántas bases hay que sustituir, añadir o eliminar. Cada una de estas operaciones tiene un coste asociado a los agentes participantes, es decir, cambiar una adenina por una timina puede implicar un coste de 4, mientras que eliminar una guanina podría suponer un coste de 13.

Los costes de las distintas operaciones se encuentran tabulados y se emplean ampliamente para este y otros procedimientos bioinformáticos. En el caso de alineamiento de proteínas son especialmente famosas las matrices BLOSUM. Los distintos costes se basan en los ratios de sustitución de esos aminoácidos que podemos encontrar en la naturaleza. Volviendo al problema que nos atañe, se propone introducir este concepto para el cálculo de las prioridades. Una matriz, idéntica en estructura a una matriz BLOSUM o PAM, que permita, mediante los valores de los costes, dirimir qué *pathway* ha de implementarse. Por ejemplo, si tuviéramos una red en miniatura de tres genes A, B y C, su matriz podría ser como la que encontramos en la Tabla 4.8. Asumiendo que la prioridad para un gen y su ausencia es la misma (para A y para $\neg A$).

Tabla 4.8: Ejemplo de tabla de prioridades para la resolución de conflictos.

Genes \ Genes	A	AB	AC	B	BC	C	ABC
A	0	1	2	1	3	7	6
AB	2	0	3	4	2	10	7
AC	1	2	0	3	4	3	3
B	3	6	4	0	1	4	5
BC	5	4	1	2	0	2	2
C	4	8	2	6	1	0	1
ABC	4	8	2	6	1	2	0

De esta forma supongamos que tenemos dos *pathways* de la forma $BC \xrightarrow{1:1,1} A$ y $A \xrightarrow{1:1,0} A$ respectivamente. Es inmediato apreciar que existe un conflicto donde $\Psi = A \wedge B \wedge C$. Por lo tanto, ¿cuál de ellos ha de prevalecer? Si consultamos la Tabla 4.8 (comenzando por la fila para consultar el valor en la columna), observamos que BC vs. A da un valor de 5 mientras que A vs. BC nos aporta un valor de 3, por lo tanto podemos concluir que BC tiene prioridad frente a A.

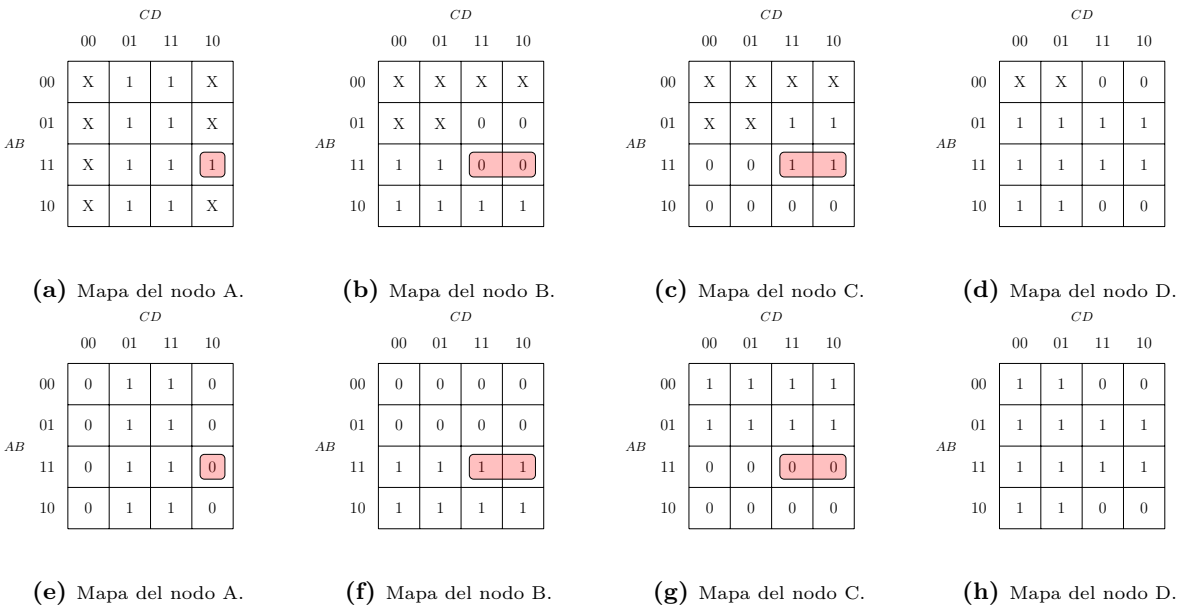
Estas tablas, inspiradas por las matrices BLOSUM o PAM, constituyen un mecanismo perfecto para introducir información biológica a priori. Nuestro proceder difiere en tanto que ha sido adaptado para el problema que nos atañe. No obstante, como se exponía en el párrafo anterior, se

han empleado durante las últimas décadas hasta el día de hoy, por lo que pueden constituir una alternativa perfecta a cualquier procedimiento matemático. La apuesta que se hace en este trabajo por este enfoque se basa en el hecho de que cualquier procedimiento matemático, en ausencia de información podría, en el mejor de los casos, inferirla. No obstante, la variabilidad biológica amenaza con romper cualquier procedimiento de inferencia bajo la asunción de cualquier agente no contemplado en el modelo, latente, esperando para ejercer su acción en situaciones específicas. Es decir, *no es posible inferir cuando no se dispone de suficiente información.*

Por otra parte, el principal inconveniente de este acercamiento reside en el hecho de que estas tablas, con el fin específico que se está exponiendo en este trabajo, todavía están por construir. Es decir, carecemos de información biológica, datos experimentales, con los que fabricarlas. Es por eso que, más adelante, las tablas empleadas han sido generadas aleatoriamente, aunque con restricciones para dotarlas de sentido biológico. Pese a todo, es de resaltar la importancia de introducir conocimiento previo dado que dotará de un mayor sentido al procedimiento de inferencia, lo que permitiría reducir el peso de la combinatoria en este y otros procedimientos. Una vez más, estas tablas constituyen el procedimiento perfecto para introducir dicho conocimiento.

Aprovechando lo desconocido

Uno de los problemas del procedimiento base es que no cuenta con posibles casillas que, por ser desconocido su valor, no podemos introducir dentro de las regiones que satisfacen las condiciones. En esencia, dentro de las regiones destino de Ψ o Y, en otras palabras, de S. De esta manera, estamos obviando posibles soluciones a nuestro problema. ¿Podemos conocer dicho valor? Lo cierto es que no, aunque sí que podemos establecer una aproximación de calidad. Para ello observemos los mapas de Karnaugh expuestos en la Figura 4.3.



(a) Mapa del nodo A. **(b)** Mapa del nodo B. **(c)** Mapa del nodo C. **(d)** Mapa del nodo D.

(e) Mapa del nodo A. **(f)** Mapa del nodo B. **(g)** Mapa del nodo C. **(h)** Mapa del nodo D.

Figura 4.3: Comparación entre los mapas de Karnaugh obtenidos a partir de la resolución de los conflictos del grafo de la Figura 4.1 (a-d) con el obtenido mediante combinatoria de NCBF partiendo del grafo a partir del algoritmo descrito en 4.3 (e-h). Los valores marcados en rojo representan las diferencias entre mapas del mismo nodo.

Si pasamos por alto las X, dado que se encuentran en casillas que podrían presentar cualquier valor y por lo tanto no pueden ser causa de discrepancia, nos encontramos con mapas en todos los casos muy similares. Concretamente, en el caso de la D son perfectamente compatibles. Es este quizá, uno de los hallazgos más importantes de este proyecto, la gran equivalencia, casi convergencia, que existe entre estos dos procedimientos de abordar el mismo problema, la inferencia de redes booleanas y la resolución de conflictos. De esta forma, queda de manifiesto la afirmación de Joo y col., 2018, en efecto, los procesos biológicos tienden a expresarse a través de NCBF.

Aún así, la correspondencia no es perfecta, esto es debido a que en la propia resolución de los conflictos se infiere información que no está presente de forma directa en el grafo. Esto se aprecia en el nodo A, es imposible que una NCBF, partiendo exclusivamente de la información recogida en el grafo, exprese el valor de 1 que podemos encontrar en la casilla A=1, B=1, C=1, D=0 del mapa (a) de la Figura 4.3.

Es por esta gran similitud que, enlazando con lo que exponíamos en el apartado *Orden de aplicación de los conflictos* podemos asumir que sí que conocemos S y, por tanto, podemos emplearlo para resolver la inserción de *pathways*, al mismo tiempo que podemos emplear los valores, a priori desconocidos, de X para calcular las distintas regiones con precisión. En otras palabras, antes de calcular los conflictos, podemos generar un conjunto de NCBF a partir del grafo y tratarlo como el conjunto de expresiones que representan nuestra red. Posteriormente, sobre esta expresión calculamos iterativamente los *pathways* de la propia red hasta alcanzar la convergencia en la expresión.

Es esta la base teórica, la teoría de los conflictos modificada, del algoritmo para el cálculo de los *pathways* que se describe en el apartado siguiente. Un algoritmo para la resolución iterativa de los conflictos dado un conjunto de *pathways*.

4.5 Algoritmo para la resolución de los conflictos entre los *pathways* de un grafo

Este es el algoritmo que se ha introducido con anterioridad y que se construye sobre la teoría expuesta en a lo largo del documento. Es por tanto un algoritmo desarrollado sobre la teoría de conflictos expuesta en Layek, 2012, con las variaciones que explicamos en el apartado anterior. El objetivo es obtener un conjunto de *pathways*, dado un grafo, entre los que no exista ningún tipo de conflicto.

1. Generamos una red compuesta de NCBF a partir del grafo como se ha expuesto en apartados en anteriores.
2. Generamos una matriz de prioridades de forma aleatoria. Con las restricciones para que sea biológicamente coherente. Restricciones como que todos los valores en la diagonal han de ser 0.
3. Disponemos todos los *pathways* en un único grupo.
4. Tomamos el primer nodo de la red y extraemos del grupo de *pathways*, todos aquellos que afecten al nodo tratado. Dividiendo en dos el grupo original.

5. Aplicamos sobre el grupo de los *pathways* del nodo tratado el algoritmo de resolución de conflictos (explicado a continuación). Obtenemos otro subconjunto de *pathways*, que afectan no solo al nodo tratado sino a otros nodos. Fusionamos este grupo de *pathways* con el otro que habíamos abandonado anteriormente. El que contiene los *pathways* que no afectan al nodo tratado.
6. Repetimos los pasos 4 y 5 con todos los nodos de la red.
7. Comprobamos que el conjunto con todos los *pathways*, tras las operaciones con todos los nodos, sea exactamente el mismo que teníamos antes de resolver ningún conflicto en ningún nodo. Si lo es, damos por finalizado el proceso, ese es el conjunto de *pathways* que hay que aplicar. Si no lo es, repetimos los pasos 4, 5, 6 y 7.

4.5.1 Algoritmo de resolución de conflictos

Este algoritmo se plantea para ser integrado dentro del paso 5 del algoritmo anterior. El objetivo de este algoritmo es, dado un conjunto de *pathways*, obtener otro conjunto diferente en el que no exista conflicto alguno entre sus miembros. En este grupo final pueden existir *pathways* que afecten a otros nodos diferentes del tratado, aunque no en el inicial.

1. Dividimos los *pathways* en activadores e inhibidores.
2. Asignamos a cada activador un inhibidor.
3. Si algún *pathway*, tanto activador como inhibidor, no ha sido emparejado (por ejemplo porque el número total sea par), se asigna a un grupo denominado *siguiente*.
4. A cada una de las parejas se les pasa la red compuesta de NCBF y resolvemos el conflicto sobre ella. Resolver el conflicto implica:
   a) Consultar la matriz de prioridades para obtener cuál es el *pathway* que ha de prevalecer.
   b) Calcular la función problema Ψ.
   c) Eliminar el *pathway* con prioridad inferior. Suponiendo que dicho *pathway* es $Y \xrightarrow{1:1,b} Q$, generamos otro de la forma $\overline{\Psi}Y \xrightarrow{1:1,b} Q$.
   d) Introducir el nuevo *pathway* $\Psi \xrightarrow{1:1,1} S(\overline{b})$. Hay que destacar que en cada conflicto imponemos las condiciones de los *pathways* sobre los mapas para acto seguido calcular $S(\overline{b})$ sobre los mapas, la red compuesta de NCBF.
   e) En el caso de que $S(\overline{b})$ conste de varios minitérminos, seleccionar cualquiera de ellos al azar (recordemos que una función booleana da 1 en cuanto cualquiera de sus minitérminos da 1). Es decir, en el miembro derecho del *pathway* nunca va a estar $S(\overline{b})$, sino cualquiera de los minitérminos que la componen.
   f) Devolver los *pathways* nuevos así como el de mayor prioridad.
5. Reunimos todos los pathways que hemos obtenido de resolver los conflictos. Todos los que afecten a otros nodos distintos de aquel que estamos tratando los introducimos en un grupo aparte denominado *extra*.
6. EL resto de *pathways* los agrupamos de nuevo junto con los que dejamos en *siguiente*.

7. Repetimos los pasos 1 a 5 hasta que dejamos de generar conflictos, por tanto no introducimos más *pathways* y hemos comparado todos los activadores con todos los inhibidores.
8. Devolvemos un nuevo grupo de *pathways* que surge de combinar los activadores, inhibidores y *extra*.

De esta forma, partiendo del grafo y con una NCBF, podemos obtener un conjunto de *pathways* compatibles entre sí, que no provoquen ningún conflicto, para ser empleados directamente.

4.6 La unión de la Teoría de Conflictos con las NCBF

A lo largo de los apartados anteriores hemos desgranado dos mecanismos. El primero de ellos, un procedimiento adaptado a nuestro problema para generar NCBF de forma eficiente. El segundo, una modificación de la teoría de conflictos expresada en Layek, 2012 así como el algoritmo para generar estos conflictos automáticamente. No obstante, no es ninguna de estas cosas lo que pretendíamos. Nuestro objetivo es obtener un algoritmo para la inferencia de modelos de redes booleanas.

¿Cómo lo vamos a alcanzar? Combinando ambos mecanismos. Partiendo de las NCBF, pues ya tenemos evidencia propia de que es muy probable que el grafo termine por manifestarse a través de un conjunto de NCBF, e introduciendo las restricciones calculadas a partir de los conflictos. De esta forma, con las restricciones, perfeccionamos la forma de las NCBF. Adentrándonos en la generalidad de funciones booleanas. Para alcanzar este objetivo, el de combinar ambos enfoques, hemos de recurrir a, quizás, otro de los hallazgos más importantes de este proyecto. Los dos teoremas que exponemos a continuación.

- *Teorema* 1: Toda función booleana se puede expresar como combinación de NCBF.
- *Teorema* 2: Dadas dos funciones booleanas f y Q, existe una transformación K, tal que $Q = K(f)$. Donde K es de la forma:

$$Q = K(f) = R \wedge f \vee D$$

  Donde, $R = \bigwedge_i \bigwedge_j \neg M_{ij}$, $D = \bigvee_m \bigvee_n M_{mn}$ y M_{ij} y M_{mn} son NCBF. Hay que resaltar que en esta expresión, $\bigwedge$ y $\bigvee$ son los equivalentes lógicos de $\prod$ y $\sum$. Es decir, $\bigwedge$ representa un productorio lógico ($\bigwedge_{i=1}^{k} = x_1 \wedge x_2 \wedge x_3 ... \wedge x_i$) y $\bigvee$ un sumatorio lógico ($\bigvee_{i=1}^{k} = x_1 \vee x_2 \vee x_3 ... \vee x_k$).

Estas dos afirmaciones, cuyas demostraciones se encuentran en los anexos, nos dan un gran margen de maniobra. Primero de todo, ¿qué es lo que significan? La primera de ellas es clara, con las herramientas de las que disponemos actualmente podemos expresar cualquier función booleana. La segunda es, sin lugar a dudas, la más interesante. En esencia viene a decir que podemos transformar rápidamente cualquier función booleana en cualquier otra, únicamente a partir de NCBF, o lo que es lo mismo, rápidamente y con sentido biológico. Más adelante profundizaremos en esto. Así pues, sabiendo que la hibridación de estos dos planteamientos no es solo matemáticamente posible, sino biológicamente coherente, procedemos a realizarla.

4.6.1 Hibridación directa

Es este el mecanismo de unión más claro. Aunque siempre nos encontramos bajo el paraguas del primer teorema, este método se fundamenta en el segundo. Tomemos de nuevo el ejemplo comparado de la Figura 4.3.

Podemos percibir por las regiones en rojo que existen incompatibilidades entre los mapas obtenidos de los grafos y aquellos que pudiéramos obtener directamente de las NCBF. Como se ha argumentado con anterioridad, es imposible obtener, salvo casos excepcionales como el del nodo D, todos los mapas completos de la red a partir de NCBF porque no se encuentra reflejada la información dentro del propio grafo. Tomemos como ejemplo el nodo A, cuyo único *input* es el D, ¿cómo podríamos inferir sin introducir más datos que para la combinación A=1, B=1, C=1 y D=0 del nodo A, que su valor es 1? Por supuesto, la pregunta es ¿cómo podríamos averiguarlo sin recurrir a la combinatoria? No podemos, es necesario introducir más información. No obstante, podemos tomar como base la función inferior, e imponer directamente las condiciones reflejadas en el mapa superior. Para ello recurrimos al Teorema 2.

En las demostraciones (anexos) introducimos que todos los minitérminos, las expresiones que rigen cada una de las celdas de una mapa de Karnaugh, son NCBF. Así pues, si la expresión de A es $A = D$, la expresión con la restricciones impuestas es $A = D \vee (A \wedge B \wedge C \wedge \neg D)$. Fijémonos en que esta última expresión se ajusta perfectamente a la transformación del Teorema 2, donde solo hay un sumando, $A \wedge B \wedge C\neg D$. Es este el procedimiento en que consiste la hibridación directa, la imposición directa de las restricciones sobre las NCBF. De esta forma las expresiones transformadas son:

$$A = A \vee (A \wedge B \wedge C\neg D)$$
$$B = \neg(A \wedge B \wedge C) \wedge A$$
$$C = A \vee (A \wedge B \wedge C)$$
$$D = B \vee \neg C$$

Los mapas resultantes se muestran en la Figura 4.4. Como se puede apreciar en las expresiones de A, B y C (no en D dado que no la hemos modificado), todas las transformaciones responden a la expresión definida en el Teorema 2. Así mismo, la función del nodo A no es una NCBF, dado que no todas sus variables son canalizantes, por lo que se verifica el Teorema 1. El mapa sobre el que han de imponerse ha de ser el mismo sobre el que se calcularon los propios *pathways*, sobre el que se resolvieron los conflictos.

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	1
10	0	1	1	0

(a) Mapa del nodo A.

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	1	1

(b) Mapa del nodo B.

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

(c) Mapa del nodo C.

AB \ CD	00	01	11	10
00	1	1	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	0	0

(d) Mapa del nodo D.

Figura 4.4: Mapas resultantes de la imposición de los conflictos sobre la red de NCBF.

Como se lleva apuntando durante todo el trabajo no solo los minitérminos son NCBF. Es decir, dependiendo de su morfología, por el Teorema 2, otras funciones, más elaboradas, podrían imponerse sobre las NCBF. Más allá del aspecto matemático, uno de los puntos interesantes aquí es el recorrido inverso, ¿una imposición matemática podría tener sentido biológico? Más aún, una imposición que formalmente se puede enmarcar dentro de la misma familia que las restricciones biológicas, ¿podría tener sentido biológico? La respuesta es que sí. Prestemos atención a las restricciones impuestas por los *pathways*, todas son NCBF. Encontrarse con una NCBF es relativamente común, no obstante, ahí está la evidencia, todas las restricciones trabajadas en este proyecto son NCBF. Es lógico pensar que una solución o imposición matemática, que a nosotros se nos revela como un mero paso intermedio, atendiendo a su forma, puede llegar a tener implicaciones biológicas de trascendencia.

Llegados a este punto cabe preguntarse si es posible generar conocimiento. No olvidemos que el grafo no deja de ser una representación, esquemática, de un sistema de ecuaciones diferenciales. La necesidad de todo el trabajo desarrollado en estas páginas radica en cómo representar esa información cuando los sistemas de ecuaciones diferenciales son incapaces de manejar sistemas de grandes dimensiones. Sin embargo, no es de eso de lo que estamos hablando ahora mismo. La nueva pregunta es si es posible inferir conocimiento del propio grafo, si es posible trascender los datos experimentales. *Hacer Biología en la pizarra*, a la manera de las Matemáticas o la Física. Si podemos ir más allá del conocimiento previo. La respuesta es que sí.

4.7 Más allá

Tomemos, una vez más, el ejemplo de la Figura 4.3. Centrémonos en las regiones en rojo. Hasta ahora sabemos que son regiones en las que los mapas no coinciden. Por otro lado, sabemos que podemos transformar cualquier función, valiéndonos de NCBF, en cualquier otra. Ahora realicemos el camino inverso. Supongamos que si un comportamiento matemático se puede ajustar a lo que sería una NCBF, es que le podemos asociar un carácter biológico. Por tanto, las preguntas serían ¿cuáles son dichos agentes? Y, ¿cuáles son sus expresiones? Ciñéndonos al ejemplo, nos vamos al nodo A. Cuya expresión es $A = D \vee (A \wedge B \wedge C \wedge \neg D)$.

Si entendemos que lo expresado en el párrafo anterior es cierto, y por tanto, el sumando de la derecha se corresponde con un ente biológico por descubrir. La expresión pasa a ser: $A = D \vee E$. Llamamos por tanto a este nuevo ente E. En principio podríamos pensar que $E = A \wedge B \wedge C \wedge \neg D$. No obstante esta asunción sería errónea, ¿por qué? Por las restricciones temporales. Digamos que si en t=0 A=1, B=1, C=1 y D=0, en t=1 obtendremos A = 1 porque en t=0 $A \wedge B \wedge C \wedge \neg D = 1$. Si impusiéramos que $E = A \wedge B \wedge C \wedge \neg D$, en efecto, para t=1, E=1 porque $A \wedge B \wedge C \wedge \neg D = 1$. No obstante, eso implica que para t=1 la expresión que tendríamos de A sería $A = D \vee 1$ y por tanto, sería en $t = 2$ donde A=1. En otras palabras, al introducir la expresión directamente estamos introduciendo un retardo con consecuencias directas sobre la dinámica de la red.

Consecuentemente, hemos de salvar ese retardo. ¿Qué es lo que hace valer 1 a A, B, C y 0 a D? Si conseguimos expresar E en función de esos términos, podremos adelantarnos un paso temporal. Análogamente, supongamos que detrás de la expresión $A \wedge B \wedge C$ se encuentra otro agente biológico, F. Y, por tanto, las expresiones de la red quedan.

$$A = A \vee E$$
$$B = \neg F \wedge A$$
$$C = \neg A \vee F$$
$$D = B \vee \neg C$$

Es inmediato comprobar que nos encontramos ante una red de NCBF. De forma que, a priori, nuestro planteamiento va por buen camino. Es decir, matemáticamente, esta expresión podría ser la descripción de un sistema biológico. Volviendo a la expresión de A, si ahora nos centramos en qué es lo que haría 1 a A, B y C tenemos que para que B=1, F=0 y A=1, y para que C=1, A=1 o F=1. Es decir, para que E fuera cierto, podría ser que necesitáramos simultáneamente que F=0 y F=1, lo que por definición es imposible. En este caso, D no nos afecta. Lo mismo nos ocurre para la descripción de F, dado que también depende de los nodos B y C.

Si la escasez de combinaciones no nos ofrece un camino alternativo en cuanto a agrupaciones de las variables, las NCBF sí que nos permiten obtener expresiones que, bajo ciertas condiciones, puedan responder a estas demandas. La dificultad radica en que desconocemos cuál puede ser su forma. Es decir, ¿cuál puede ser la combinación, estructural de E y F que permita el comportamiento deseado? En este trabajo, dado que no tenemos ninguna información, hemos analizado todas las combinaciones estructurales posibles. Partiendo de la base de que $E = f(A, F, B, C, E)$ y que $F = f(A, E, F)$. Es decir, conocemos de qué variables dependen, pero no cuál es su estructura. Hay que resaltar que no fijamos la dependencia de las variables directas, A, B, C y D en el caso de E y A, B y C en el caso de F, si no de las que dependen estas a su vez. La intención es la que apuntábamos con anterioridad, adelantar las expresiones un paso temporal.

Recurriendo a la combinatoria, obtenemos unas 59900 combinaciones diferentes que podrían suplir los requisitos aparentemente insalvables de F=0 y F=1. Hay que destacar que en ningún momento estamos simplificando o eludiendo la lógica de las expresiones. Simplemente, ese número de funciones surge de jugar con las capas de la Ecuación 4.6 en cada expresión. De tal forma que un valor de una variable pueda no afectar en una expresión pero sí en la otra. En otras palabras, como estamos trabajando con NCBF, por definición, todas sus variables son canalizantes, lo que acorde con Li y col., 2013 implica que son capaces de imponer un valor, dejando margen de libertad en el otro. Así mismo, qué variables han de prevalecer viene marcado por las capas, M_i.

Tomando ese concepto para nuestro problema, F=0 y F=1 se pueden salvar si, en la expresión de E, F fuera variable canalizante de primera prioridad (en M_1) para el valor de 1. Y cuando fuera 0, dejar actuar a otras variables, en este caso A, para los nodos B y C. El juego de prioridad vuelve a actuar sobre A y así sucesivamente. Aunque A logra el efecto deseado para F=0. Es decir, depende de la combinación pero el resultado es posible y la prueba de ello es el gran número de funciones obtenidas.

No obstante, ahora se nos presenta otro problema, 59900 redes diferentes son demasiadas. Más todavía si tenemos en cuenta que partíamos de una red muy simple. En otras palabras, muchas de esas redes carecen de sentido biológico. Es por eso que nosotros hemos recurrido al único criterio de validación que tenemos, los atractores. Si una red tiene sentido biológico y se corresponde con el grafo, necesariamente debe cumplir con los atractores, pues son el régimen permanente del sistema representado en el grafo. Una vez más, aparece ante nosotros otro problema, y es que

los atractores que hemos obtenido del grafo están definidos únicamente para A, B, C y D. Por lo tanto es necesario algún tipo de proyección, con la que podamos evaluar una red de 6 nodos en este caso sobre unos atractores de 4.

Los dos mecanismos de proyección abordados en este trabajo han sido llamados proyección simple y proyección compuesta. El primero de ellos consiste simplemente en buscar aquellas redes que tienen atractores simples cuyos valores de A, B , C y D se corresponden con los atractores de los que disponemos. De esta forma, reducimos el total de redes a 5182. Aunque sigue siendo un número alto, en este caso sí que es manejable. De todas formas, en este trabajo no se han implementado procedimientos de filtrado o de resistencia al ruido como los descritos en Joo y col., 2018. Por lo que aún mucho trabajo se puede desarrollar de cara a reducir el número total de redes a revisar.

4.8 Validación con los atractores

A lo largo de las últimas páginas hemos definido y empleado el término atractor. Como se ha expuesto, un atractor se corresponde con uno de los posibles regímenes permanentes del sistema. Desde el punto de vista biológico, un atractor representa la oportunidad perfecta para incorporar conocimiento al proceso de selección, de tal forma que las soluciones de los sistemas seleccionados se correspondan con las del sistema real. Pese a haber introducido este término, en ningún momento hemos expresado cómo realizar dicha introducción o comparativa. Para realizarla, en este trabajo hemos contemplado dos opciones:

Aproximación basada en Teoría de Grafos

El enfoque tradicional. Una forma de representar una red booleana es mediante un grafo dirigido. Un grafo dirigido donde cada estado transiciona a un único estado y al cual se puede llegar exclusivamente desde otro diferente. Hay que destacar que este grafo no es el mismo grafo que se suministra como datos del problema. Cada uno de los distintos nodos de este grafo es una combinación de valores de las variables de la red. Es, literalmente, la representación gráfica de la red booleana. Así pues, un atractor en Teoría de Grafos es lo que se conoce como Componente Fuertemente Conexa (CFC). Es importante resaltar que, aunque todos los atractores son CFC no todas las CFC son atractores. Por otro lado, también es cierto que para el caso concreto de grafo que representa una red booleana, ambos conceptos son equivalentes.

Es por eso que este enfoque se basa en el Algoritmo de Tarjan (Tarjan, 1972), que sirve para calcular las CFC de grafos dirigidos. Para obtener los atractores de la red: 1) fijamos la tabla de verdad de la red (lo que es lo mismo que obtener el grafo), 2) efectuamos el algoritmo de Tarjan para obtener las CFC (atractores) de la red y 3) comparamos con aquellos atractores que buscamos. Afortunadamente, por su amplia aplicación, este algoritmo lo encontramos ya programado en Python dentro del paquete PyBoolNet.

Aproximación basada en Álgebra Lineal

El enfoque expuesto en el apartado anterior, quizás por su antigüedad, es el más empleado en la actualidad a la hora de tratar con redes booleanas. No obstante, a principios de este siglo se introdujo un revolucionario modo de concebir la lógica booleana de la mano de Cheng, 2001. Con el producto (izquierdo) del Semi-Tensor (STP) es posible representar cualquier función booleana mediante productos matriciales, lo mismo ocurre con cualquier red booleana, que viene dada por una expresión matricial.

Específicamente, el STP de dos matrices A y B se define con la expresión 4.10:

$$A \ltimes B = (A \otimes I_{\frac{\alpha}{n}})(A \otimes I_{\frac{\alpha}{p}}) \tag{4.10}$$

Donde $A \in M_{mxn}$, $B \in M_{pxq}$, α es el mínimo común múltiplo de n y p y $\otimes$ denota al producto de Kronecker. El STP es, en esencia, una generalización multidimensional del producto de matrices. Si se dan las restricciones dimensionales adecuadas, el STP es equivalente al producto matricial convencional. También existe un producto derecho del semi-tensor, no obstante, dadas sus limitadas propiedades se emplea mucho menos que el STP.

Es de recalcar la belleza de este enfoque. No solo porque permite entender la lógica booleana desde una región del Álgebra Lineal, lo que en sí es digno de admiración, sino porque acerca la lógica booleana. Salva la división conceptual existente entre la Lógica y el resto de las Matemáticas. El caso es que el autor, Daizhan Cheng, ha desarrollado ampliamente esta forma de concebir la lógica con multitud de algoritmos y herramientas recogidas en su libro de 2011 (Cheng y col., 2011). A parte, ha desarrollado completamente un *toolbox* para MATLAB con todos estos algoritmos, en múltiples versiones.

Así pues, emerge ante nosotros otro mecanismo con el que validar la existencia de los atractores. Podemos construir la matriz que representa la red booleana para, con el producto del Semi-Tensor, comprobar si dicha matriz verifica la existencia de los atractores. Importante, sin llegar a obtener los atractores propios de la red. Que a priori no nos interesan, es decir, lo relevante para el cribado es que entre las soluciones del sistema se encuentren las que nosotros hemos fijado, en un principio el resto de soluciones no nos interesan.

Es este, en opinión del autor de este trabajo, otro de los puntos principales del proyecto porque, aunque sí que es cierto que las redes booleanas están caracterizadas plenamente dentro del Álgebra Lineal con el producto del Semi-Tensor, durante el desarrollo del trabajo no se ha encontrado ninguna referencia, ni bibliográfica ni en *software*, que asegure su empleo para esta tarea.

Pueden ser muchos los inconvenientes de este enfoque. El principal de ellos probablemente sea el coste computacional. Como asegura Akutsu, 2018, los algoritmos basados en el STP implican un alto coste computacional dado el gran tamaño de las matrices originadas. En el presente trabajo se ha tenido oportunidad de comprobar la veracidad de esta afirmación. No obstante, como muy bien indica Akutsu, 2018, estas matrices se encuentran prácticamente vacías. Por lo tanto, como pone de manifiesto Cheng y col., 2011, precisamente por la propia naturaleza de la matriz, se puede expresar de forma condensada mediante vectores numéricos. Por ejemplo, acorde con la nomenclatura de Cheng y col., 2011:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \delta_8^{4,7,1} \tag{4.11}$$

En esta nomenclatura, el número inferior se corresponde con las dimensiones de la matriz identidad de las que se extraen las columnas indicadas por los números en el superíndice.

Al principio de este trabajo se comenta la naturaleza markoviana de los sistemas representados. Es a través del STP donde se pone de manifiesto. Con el sistema del ejemplo, con 4 variables, el vector donde se representan los distintos estados del sistema pasa a ser un vector con 16 entradas (2^4 estados del sistema). La red booleana se representa mediante una matriz de transición de 16x16. Al multiplicar el vector que simboliza el estado del sistema por la matriz que representa el sistema en sí, transicionamos hacia el siguiente estado. Es decir, estamos realmente ante un sistema que se puede representar mediante una matriz markoviana determinista con la que empleamos el STP, que como se ha expuesto anteriormente, es una generalización del producto convencional. Y, dado que las restricciones dimensionales se cumplen, se comporta como un producto matricial convencional. Como resultado, las redes booleanas son sistemas markovianos.

Empleando la nomenclatura basada en δ, es posible representar estas matrices de forma compacta y efectuar los productos muy rápidamente. Sin llegar a construir la matriz. Actualmente, de todos los procesos descritos en el proyecto. La parte más costosa es la obtención de los atractores con el algoritmo de Tarjan. En consecuencia, una programación eficiente de los algoritmos planteados en Cheng y col., 2011 podría disminuir considerablemente el coste computacional de todo el procedimiento mediante la validación de los atractores con el STP, a partir de la construcción de la representación matricial de la red booleana.

Capítulo 5

Caso de aplicación: transición epitelio-mensénquima

En este capítulo se muestra un caso de aplicación del software desarrollado en este trabajo, la transición epitelio-mesenquima. Se expone no solo la expresión del modelo, sino su comparación en términos de extensión y naturaleza de los atractores. Así mismo, exponemos la novedad de nuestra aproximación con respecto a otras soluciones, el enfoque deductivo.

5.1 Introducción

Son numerosos los procedimientos de los que depende la evolución de un tumor. Sin duda, uno de los más temidos es la metástasis. En el sistema TNM, el mecanismo de clasificación que emplea la SEOM[1] para el NMIBC, la M es el marcador que indica la presencia tumoral en regiones lejanas a la fuente original. En otras palabras, la M es el marcador que se emplea para la metástasis. Conforme al *National Institutes of Health* (NIH) estadounidense, la metástasis es la *diseminación de células cancerosas desde el lugar donde se formó el cáncer por primera vez hasta otra parte del cuerpo.*

No obstante, ¿puede una célula tumoral moverse? La respuesta es que depende de su clasificación histológica. Tomemos el símil de la piel. A grandes rasgos, la piel se divide en dos regiones diferentes. Por un lado el parénquima, la región histológica donde se encuentra el tejido epitelial, y por otro lado el mesénquima, donde se genera y se soporta ese tejido epitelial. Esta distinción la podemos extrapolar a otras regiones donde encontremos tejido epitelial, como en las glándulas o la vejiga. Así, una célula epitelial no se puede mover, ya que existen estructuras que, a modo de costuras, impiden que las células epiteliales se separen, lo que descompondría el epitelio. Por lo tanto, una célula epitelial, debido a estas estructuras no puede moverse. Por el contrario, esto no ocurre con ciertos tipos celulares del mesénquima, que sí que tienen movilidad debido a su morfología (no tienen polaridad) y a que carecen de estas estructuras de cohesión.

Esto es lo que ocurre con las células madre pluripotenciales (PSC), una variedad celular que podemos encontrar en el mesénquima. Células de escasa diferenciación, por tanto con alto potencial

[1] Sociedad Española de Oncología Médica.

reproductivo y que carecen de anclajes. Justo lo opuesto a las células epiteliales. En estos tejidos es en el mesénquima donde las PSC proliferan y se diferencian, en un proceso progresivo hasta formar las células epiteliales. En la actualidad es posible revertir dicho proceso de diferenciación hasta devolver las células epiteliales a su estado de PSC. Así, estás células revertidas reciben el nombre de *induced pluripotent stem cells* (iPSC), es decir, PSC inducidas. Por otro lado, este proceso no ha de ser necesariamente artificial. Por el contrario, se da de forma natural durante la evolución del cáncer y constituye uno de los pilares de su propagación. En el melanoma, o en el caso que nos atañe, el NMIBC, si no se produjera esta transformación sería imposible que el tumor primario, de origen epitelial y por tanto inmóvil, colonizara otras regiones.

Consecuentemente, este fenómeno es un punto de inflexión en la evolución del tumor y el proceso sobre el que se ha decidido aplicar el procedimiento descrito en páginas anteriores. Concretamente, hemos reproducido el experimento publicado en Joo y col., 2018. En esta publicación, se parte de la información proporcionada en Lu y col., 2013 para aplicar combinatoria centrada en el dominio de las NCBF. Esta combinatoria es validada por expertos hasta conseguir el modelo que se presenta en la publicación. Nosotros, al igual que ellos, hemos partido del grafo ofrecido en Lu y col., 2013 solo que le vamos a aplicar el procedimiento de inferencia descrito en este trabajo. Por las limitaciones del mismo, no podemos aplicar una validación por expertos. La idea es comparar su enfoque, empírico, con el que se ofrece en este proyecto, deductivo. Nosotros, en lugar de buscar el resultado que mejor explica los resultados experimentales, hemos deducido el resultado que es biológicamente más coherente, dado el conocimiento a priori del que disponemos: el grafo y los atractores.

5.2 Datos iniciales

El grafo que introducimos en el *software* es el descrito en la Figura 5.1 (Lu y col., 2013).

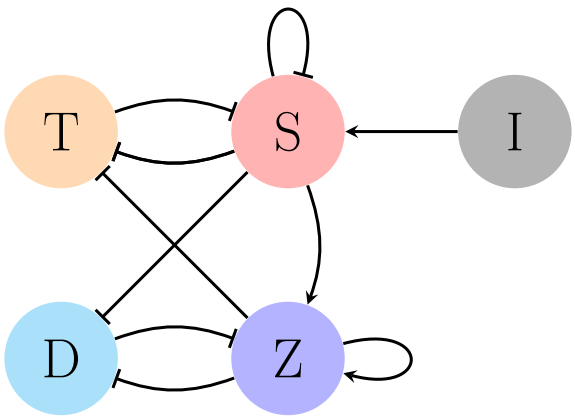


Figura 5.1: Grafo original de la EMT extraído de Lu y col., 2013. Una flecha acabada en punta indica una relación de activación mientras que una acabada con la forma −| indica una relación de inhibición. Por ejemplo, el nodo I activa al nodo S, mientras que el nodo S inhibe al nodo D. Los nombres han sido cambiados con respecto a los que encontramos en el artículo original. I representa el input del sistema, S es *Snail*, Z es *Zeb*, T es μ34 y D es μ200.

En el sistema descrito, se representa la EMT regulada por dos grandes familias de factores de transcripción, *Snail* (S) y *Zeb* (Z), combinadas con otras dos familias de μRNA, μ200 (D) y μ34

(T). El input (I) simboliza una amalgama más amplia de diferentes señales celulares, como los factores de transcripción TFG-β o HGF. En esencia, el *input* es una señal que indica el comienzo de la EMT.

Experimentalmente se ha demostrado que existen 3 fenotipos celulares en el proceso de la EMT: epitelial, híbrido y mesenquimal. Cualquier célula que sufra la EMT transita a través de ellos. Por tanto, sabemos que cada uno de estos fenotipos celulares se corresponde con el atractor, en este caso simple, de una cuenca de atracción. Así, el sistema se divide en 3 cuencas de atracción. No obstante, desconocemos la naturaleza de esos atractores. Investigando en la bibliografía (Lu y col., 2013; Joo y col., 2018), obtenemos los siguientes enunciados.

1. μ34 y *Snail* forman un bucle de realimentación negativa.
2. μ200 y *Zeb* forman un bucle de realimentación negativa.
3. *Snail* y *Zeb* forman un bucle de realimentación positiva.
4. μ200 es un marcador epitelial.
5. *Zeb* es un marcador mesenquimal.

Así pues, ya tenemos el enunciado de nuestro problema. Partiendo de los *pathways* de la Figura 5.1 tenemos que ejecutar el proceso de inferencia, para obtener una red con 3 atractores simples. De igual manera, dichos atractores deben ser compatibles con las restricciones anteriores. Entre otras, que el atractor epitelial tiene que manifestar $\mu 200 = 1$ y el atractor mesenquimal $Zeb = 1$.

5.3 Resultados

El *software* analiza un total de 243 533 redes, de las cuales solo 1207 convergen para el proceso de inferencia con atractores simples. Recordemos que si cada nodo depende de 5 variables, tenemos un espacio de $2^{128} = 3{,}4 \cdot 10^{38}$ redes posibles. En otras palabras, solo por emplear el marco teórico que hemos descrito reducimos la combinatoria en 33 órdenes de magnitud. De igual manera, el proceso de inferencia constituye otro filtro, el cual, antes de aplicar la validación por atractores reduce el total de redes en 3 órdenes de magnitud adicionales. Tras la validación por atractores, nos quedamos con 71 redes de 3 atractores que cumplen parcialmente las restricciones. Concretamente las condiciones 4 y 5. De esas 71 redes, hacemos un subgrupo con las 6 más representativas para terminar eligiendo la que mostramos en la Expresión 5.1.

$$\begin{aligned}
I_{next} &= I \\
S_{next} &= (I \wedge \neg T) \vee (I \wedge Z \wedge D) \vee (\neg S \wedge \neg T) \vee (I \wedge Z \wedge \neg S) \vee (Z \wedge D \wedge \neg S) \vee (Z \wedge D \wedge \neg T) \\
T_{next} &= (\neg I \wedge \neg S) \vee (\neg I \wedge \neg Z) \vee (\neg S \wedge \neg T) \vee (\neg S \wedge \neg Z) \vee (\neg T \wedge \neg Z) \vee (\neg Z \wedge \neg D) \\
Z_{next} &= (S \wedge \neg D) \vee (I \wedge S \wedge T) \vee (I \wedge S \wedge Z) \vee (I \wedge T \wedge Z) \vee (I \wedge T \wedge \neg D) \vee (I \wedge Z \wedge \neg D) \\
D_{next} &= (I \wedge S \wedge T) \vee (\neg S \wedge \neg Z) \vee (I \wedge T \wedge \neg D)
\end{aligned} \tag{5.1}$$

Los atractores de la red expuesta en la Figura 5.1 son los estados 00101, 10101 y 11010, para las variables I, S, T, Z y D respectivamente[2]. El 00101 se corresponde con el fenotipo epitelial, el 10101 con el híbrido y el 11010 con el mesenquimal. El 50 % de los estados de la red pertenecen a

[2]Por ejemplo, para el primer atractor tenemos que I=0, S=0, T=1, Z=0 y D=1.

la cuenca de atracción epitelial, el 3.12 % pertenecen a la cuenca del estado híbrido y el 46.88 % a la cuenca del estado mesenquimal. La topografía completa de la red, así como el listado final de *pathways* se encuentran en los anexos.

5.4 Discusión

El objetivo en este apartado es validar el modelo que aportamos en este proyecto mediante la comparación con los presentes en la bibliografía consultada, Lu y col., 2013; Joo y col., 2018.

5.4.1 Cumplimiento de las condiciones iniciales

La primera validación consiste en comprobar que nuestro modelo cumple las restricciones fijadas con anterioridad. Ya que han sido extraídas directamente de la bibliografía. En lo que se refiere a la primera condición, es inmediato comprobar que $\mu 34$ hace de inhibidor sobre *Snail*. De igual manera, *Snail* inhibe a $\mu 34$. Esto se manifiesta en que tanto *Snail* en la expresión de $\mu 34$ como $\mu 34$ en la expresión de *Snail* son variables que aparecen negadas. Se ha obtenido lo que se pretendía, un bucle de realimentación negativa.

De forma análoga, observamos que la segunda condición se cumple, esta vez entre $\mu 200$ y *Zeb*. Así mismo, obtenemos el bucle de realimentación positiva entre *Snail* y *Zeb*. En este caso porque las variables no aparecen negadas en sus respectivas expresiones. Por otro lado, si nos fijamos en los atractores de mayor peso, 00101 y 11010, podemos comprobar que los valores de $\mu 200$ y *Zeb* se corresponden con los valores de las condiciones 4 y 5 de forma que indican que el primero de estos dos atractores es el epitelial, y el segundo el mesenquimal. Por tanto, la red es capaz de representar adecuadamente ambos fenotipos, ya que coincide con los resultados de Joo y col., 2018 y Lu y col., 2013. Además, cada fenotipo muestra el valor del marcador que ha de mostrar, $Zeb = 1$ para mesenquimal y $\mu 200 = 1$ para epitelial. En consecuencia, se cumplen las condiciones 4 y 5.

5.4.2 Otros modelos

El modelo que ha servido de referencia durante el desarrollo de este proyecto es el que se muestra en Joo y col., 2018. El autor, como nosotros, parte de Lu y col., 2013 para desarrollar su aproximación al problema. En nuestro caso, el modelo que aportamos es muy similar al suyo. La principal diferencia radica en la cuenca del atractor híbrido[3]. Nuestro atractor híbrido es 10101 cuando el suyo es 11011. Además, la cuenca de este atractor está constituida por 5 estados mientras que la del atractor en nuestro modelo está constituida por 1 estado, únicamente el propio atractor.

En lo demás, ambas soluciones son muy similares ya que obtienen los mismos atractores con pesos parecidos[4]. No obstante, es de resaltar varios puntos que impiden que vayamos a obtener la misma solución que se obtiene en Joo y col., 2018.

1. El primero de ellos es la matriz de prioridades. Como se indica en apartados anteriores, nuestro procedimiento se cimenta sobre la disposición de información experimental con la que dirimir los conflictos en la matriz de prioridades. Por ausencia de información, las prioridades

[3] La cuenca de un atractor es el conjunto de estados de la red que conducen a ese atractor.
[4] El peso de un atractor es la fracción de estados de la red booleana que pertenecen a la cuenca de dicho atractor.

las generamos aleatoriamente. Como resultado, nuestra matriz va a ser total o parcialmente incorrecta, y con ella parte del proceso de inferencia.

2. Por otro lado, en Joo y col., 2018 se define el atractor híbrido conforme a una interpretación propia de la información que podemos encontrar en Lu y col., 2013. Si bien sí que en Lu y col., 2013 se marcan las pautas para deducir la forma de los atractores epitelial y mesenquimal, las indicaciones para deducir el atractor híbrido son más ambiguas. Como resultado, la definición de dicho atractor está más sujeta al criterio del autor que la definición de los otros dos atractores. Así, las diferencias en este atractor no son relevantes siempre que la dinámica global del sistema sea coherente.
3. Finalmente, acorde con el criterio expuesto en Joo y col., 2018, el grafo original de Lu y col., 2013 es modificado incluyendo un bucle de realimentación positiva en el nodo $\mu 200$. Dicha modificación facilita la representación de los resultados experimentales en la red resultante. En consecuencia, el modelo que obtienen ha sido calculado a partir de otro grafo, de manera que nuestro resultado ha de ser diferente ya que nosotros no introducimos dicha modificación.

Es decir, existe una base razonable para asumir las diferencias existentes entre nuestra solución y la expuesta en Joo y col., 2018. De igual manera, no solo el modelo, sino la solución en su conjunto que obtenemos en este proyecto, tiene una ventaja fundamental con respecto a la ofrecida en Joo y col., 2018: el enfoque deductivo.

Utilización del enfoque deductivo

Nuestra solución no se basa únicamente en la combinatoria. Como se ha indicado anteriormente, con la utilización del marco teórico expuesto en este trabajo reducimos drásticamente el número de posibilidades que habría que revisar en el caso de emplear únicamente combinatoria. No obstante, eso es solo una consecuencia de emplear un enfoque deductivo. Todos los modelos se obtienen mediante un procedimiento razonado, la resolución de los conflictos. Por tanto, cualquier relación descrita puede ser objeto de debate, y no solo hemos de emplear la similitud con los resultados experimentales para validar los modelos obtenidos.

Entendemos que esto es de gran importancia dado que pueden existir muchos modelos capaces de representar adecuadamente los resultados experimentales, pero solo uno capaz de explicar la naturaleza real del fenómeno a modelar. Si lo que se pretende es explicar dicha naturaleza, es este el modelo que hemos de buscar. Creemos que en ese aspecto, el enfoque deductivo que empleamos en este trabajo supone un gran avance con respecto el enfoque expuesto en Joo y col., 2018. Dado que en este artículo, aunque se emplee combinatoria acotada dentro de las NCBF, no deja de ser combinatoria. Como resultado, no justifican las interacciones internas que han conducido a su modelo. Únicamente justifican la calidad del modelo en base a la comparación con los datos experimentales.

5.5 Conclusión

En este trabajo exponemos un método para la inferencia de redes booleanas basado en el razonamiento lógico de las relaciones internas dentro del grafo. Este enfoque tiene dos ventajas con respecto las metodologías anteriores.

1. Este procedimiento es notablemente más rápido debido a que la búsqueda del modelo se reduce a aquellas expresiones que son biológicamente coherentes con el criterio dado (matriz de prioridades). Así, al reducir el número de redes a evaluar, aceleramos el proceso de validación y, por tanto, el tiempo dedicado al conjunto de la inferencia.
2. A diferencia de otras metodologías, el procedimiento de razonamiento, el enfoque deductivo, implica que junto con el propio modelo se obtienen todas las relaciones internas, ocultas dentro del grafo. Como resultado, la evaluación de los modelos adquiere una nueva dimensión. De forma que la validación de las redes no ha de realizarse únicamente conforme a los resultados finales sino en base a las hipótesis asumidas por el *software*. De igual manera, todas estas relaciones son conocimiento inferido directamente del grafo. Es decir, no solo se están representando resultados experimentales previos sino que se establecen hipótesis que pudieran ser empleadas en futuros trabajos.

Así mismo, este procedimiento ha sido validado satisfactoriamente mediante la comparativa con el trabajo de Joo y col., 2018. Por ende, aunque aún existe un amplio terreno por cubrir, podemos afirmar que la primera experiencia con esta metodología ha sido satisfactoria. En consecuencia, este proyecto puede contribuir al estado del arte en inferencia de redes booleanas por la potencia y originalidad de su planteamiento.

Capítulo 6

Manual del programador

En este capítulo describimos el software desarrollado a lo largo del proyecto para su uso en inferencia de redes booleanas. Al igual que los algoritmos, los teoremas y las demostraciones, todo el software descrito en este capítulo es un producto original de este trabajo. A excepción de librerías empleadas para la simplificación de expresiones lógicas y parte del código perteneciente al área del STP, como se indica en sus respectivas secciones.

6.1 Introducción

Todo el *software* se encuentra documentado. En este capítulo así como en el propio código, donde se puede acceder a las estructuras de datos empleadas y los objetivos de cada función mediante el comando `help()` de Python. A excepción de algunas funciones que no se encuentran documentadas. La razón es que no son funciones independientes sino fragmentos de código integrados en otros algoritmos que sí están descritos. El motivo de este aislamiento es la obtención de una ejecución eficaz a través de una gestión eficiente de la memoria. Este capítulo es en esencia una síntesis de toda la documentación que podemos encontrar en los archivos.

Para estructurar la exposición de todo el código vamos a definir su objetivo, arquitectura y, finalmente, todas las funciones y clases con las que debería interactuar el programador a través de los ejemplos ubicados en la carpeta *examples*.

6.2 Objetivo

El *software* parte de un grafo, la representación abstracta de un sistema biológico, que en el caso tratado es un proceso patológico. Junto con el grafo, el usuario ha de introducir una serie de estados del sistema, es decir, combinaciones de variables que se corresponden con estados prototípicos del mismo. En nuestro ejemplo manifestaciones clínicas. Así, con el grafo y las combinaciones de variables (atractores), el *software* reconstruye todo el modelo matemático (red booleana) que describe al sistema biológico.

La importancia del modelo radica en que, de ser introducida en él cualquier combinación de variables (estado de la red booleana), es posible predecir hacia cuál de estos escenarios prototípicos

va a evolucionar el sistema. Tomando el caso aplicado, el modelo pretende ser un apoyo en la prognosis médica. De tal forma que dada una serie de síntomas (estado inicial del sistema), sea posible anticipar el destino del paciente, es decir, indicar hacia qué manifestación clínica se dirige (estado prototípico). Concretamente, con el caso aplicado lo que se pretende es predecir si un tumor está desarrollando capacidad para metastatizar o no.

Así mismo, todas las decisiones tomadas por el *software*, al fundamentarse sobre los algoritmos y teoría descritos en capítulos anteriores, tienen sentido biológico. Tal significado queda registrado junto con el modelo final. Por lo que supone una gran mejora con respecto a la combinatoria dado que no solo la reduce, sino que facilita la evaluación por parte de expertos mediante el análisis de las decisiones tomadas por el código.

6.3 Arquitectura

El *software* está escrito en Python y organizado en 4 *packages* más la carpeta *examples* con los ejemplos descritos en la memoria: el caso de la transición epitelio-mesénquima (EMT) y el posprocesado sobre la red que se emplea de ejemplo para introducir la teoría. No se ha aplicado dicho posprocesado sobre la red de la EMT debido a que el coste computacional necesario sobrepasa con mucho los recursos de los que se dispone para realizar este trabajo.

- *Package* stp: recoge todas las funciones necesarias para la implementación de la validación por atractores mediante la obtención de la forma matricial de la red booleana. Es una adaptación del *toolbox* para MATLAB disponible en `http://lsc.amss.ac.cn/~dcheng/`. Aunque los algoritmos son tomados de los descritos en este *toolbox*, expuestos también en Cheng y col., 2011, no son una copia directa. Han sido adaptados a la naturaleza de Python con arquitectura y tipos de datos específicos. Como consecuencia, el tiempo de ejecución de estas librerías adaptadas es de 10 a 100 veces inferior en Python que el de sus homólogas en MATLAB. En el momento de presentar este trabajo no existe ninguna librería en Python dedicada al STP. Podría ser el comienzo de la primera librería en Python dedicada al producto del semi-tensor. *Modules: classes.py*, *utils.py* y *algorithms.py*.
- *Package* logic: esta es la librería que junto con SymPy se emplea para la simplificación de las expresiones lógicas. La diferencia con SymPy radica en su algoritmo, inspirado en los mapas de Karnaugh, cuya implementación es un 50 % más eficiente que la implementación en SymPy del algoritmo de Quine-McCluskey, aunque notablemente menos robusta. Todo el código de este paquete ha sido tomado de `https://github.com/zhcHoward/Kmap`. *Modules: Kmap.py* y *utils.py*.
- *Package* graphs: recoge todas las clases y métodos necesarios para la creación y ajuste inicial del objeto grafo. *Modules: conflicts_theory.py* y *graph.py*.
- *Package* base: recoge el resto de elementos necesarios. La estructura de datos de los resultados, las excepciones definidas a propósito y todas las clases y funciones que controlan la generación de NCBF así como la estructura de la validación, que se integra con el objeto grafo. *Modules: hibrids.py*, *exceptions.py*, *ncbf.py* y *validation.py*.

6.4 Ejemplo de inferencia de red

Este apartado discurre sobre el ejemplo expuesto en *inference_example.py*. El objetivo es, a partir del grafo, realizar todo el proceso de generación de las NCBF, inferencia de los *pathways* y finalmente validación de las redes obtenidas. El programador solo ha de introducir el grafo y una lista de parámetros iniciales de cara a ajustar el proceso de simulación a los recursos computacionales existentes.

6.4.1 Algoritmo de ejecución

La explicación de todas las fases que atraviesa la ejecución del *script* .

Introducción de los datos iniciales

En esencia, la introducción del grafo y de los parámetros que van a gobernar el desempeño de la simulación. Por claridad en la explicación, dichos parámetros serán introducidos al final de la sección. El grafo debe introducirse en forma tabulada, para el caso que nos atañe, la EMT, el grafo habría de ser representado como se muestra en la Tabla 6.1.

Tabla 6.1: Representación tabulada del grafo de la Figura 5.1. En la columna Nodo se indica el nodo sobre el cual ejercen su acción los ubicados en las otras dos columnas, activación e inhibición respectivamente. El nodo A no depende de otros nodos de la red, es por tanto considerado como una variable de entrada del sistema (INPUT).

Nodos	Activadores	Inhibidores
A		
B	A	B, C
C		B, D
D	B, D	E
E		B, D

La introducción de esta tabla se realiza en forma de *DataFrame*, formato de datos característico de la librería Pandas. Así mismo, los atractores que sabemos que presenta nuestro sistema son introducidos en una lista de strings. Esto se describe en el fragmento de código a continuación.

```
import pandas as pd

# Introduce the data
index = ['A', 'B', 'C', 'D', 'E']
columns = ['activators', 'inhibitors']
content = [[[''], ['']], [['A'], ['B', 'C']], [[''], ['B', 'D']], [['B', 'D'], ['E']], [[''], ['B', 'D']]]
initial_data = pd.DataFrame(data=content, index=index, columns=columns)
attractors = ['00101', '11011', '11010']  # Introduce each node in alphabetical order
```

Despliegue de la red

Una vez que hemos introducido los datos comienza el proceso de inferencia. En esta fase definimos las estructuras sobre las que se va a desarrollar el *software* . Básicamente, creamos la base del objeto grafo. Para seguir en orden cronológico la ejecución, no solo de este apartado sino de todo el proceso de inferencia, es adecuado observar el constructor de la clase Graph (`def __init__`) en el fichero `graphs/graph.py`. Esto se debe a que un proceso de inferencia se corresponde con un objeto de esta clase. Los tres métodos de la clase Graph, en orden cronológico, que desarrollan este apartado son:

1. `def get_nodes(self)`: primer método de relevancia en ejecución con el constructor. Con la representación tabulada del grafo genera todos los objetos de la clase nodo (`class Node`, mismo fichero) asociados.
2. `def get_space(self)`: función para calcular todo el espacio booleano (combinaciones de variables) en forma de lista de diccionarios sobre el que evaluar las funciones lógicas.
3. `def set_adjustment(self)`: establece todas las relaciones entre los nodos conforme indican los datos tabulados.

Cálculo de las variantes

Una vez que hemos establecido las relaciones entre nodos realizamos los preparativos para calcular las NCBF. Fundamentalmente, en este apartado definimos matemáticamente el significado de activador e inhibidor. Conforme a Li y col., 2013, en la expresión de una NCBF, cada variable tiene un valor canalizante (responsable de la acción) y un valor canalizado (efecto sobre la función). Se ha asumido que los pares canalizante-canalizado que podría manifestar un nodo inhibidor son $\{0,1\}$ y $\{1,0\}$. Que representarían simbólicamente que la ausencia de inhibidor estimula la expresión del gen inhibido y vice versa. Por el contrario, los pares mediante los que se representa el efecto de activación son $\{1,1\}$ y $\{0,0\}$ que simbolizarían respectivamente que la presencia de activador estimula la expresión del gen activado, o que para la expresión del gen activado es necesaria la presencia del activador.

En consecuencia, como solo podemos asignar un par por variable. Se han de realizar todas las posibles combinaciones de significados, no solo dentro de la expresión de un nodo (obteniendo múltiples expresiones), sino dentro de todas las expresiones de los nodos de la red. Es, el conjunto de significados agrupados, lo que en el *software* se ha llamado *roles set*. Por lo tanto, antes de calcular las redes es necesario generar todos los *roles sets* posibles. Es así como surge el objeto *Variant*, variante (`class Variant`, `graphs/graph.py`). Se genera de esta forma un objeto *variant* por cada *roles set*. El método responsable de la generación es `def get_variants(self, limit=None)`. Es posible fijar un límite al número de variantes generadas, aunque por defecto dicho límite no existe.

Generación de las redes

Una vez que hemos definido el significado de activador e inhibidor y preparado las conexiones entre los distintos nodos, ejecutamos el algoritmo descrito en la Sección 4.3.2. De esta forma, generamos por cada variante un conjunto de redes. Para el ejemplo que se viene desarrollando, el total de variantes generadas asciende a 1024, con la generación de redes aumentamos esa cifra hasta las 24336 modelos para analizar. El método del objeto Graph (recordemos que todos los métodos comentados hasta ahora son del objeto Graph) responsable de la generación de las redes a partir de las variantes es `def get_networks(self)`. Es así como imponemos el atributo *networks* sobre el objeto Graph, una lista de objetos *network* (`class Network`, `graphs/graph.py`). Cada objeto *network* contiene los datos, el *roles set* y la estructura de su NCBF asociada. Al igual que las variantes, es único.

Ejecutamos la simulación

Una vez que tenemos todas las redes procedemos con la simulación. Cada simulación consta de dos partes, por un lado la inferencia de los *pathways* con la resolución de los conflictos, y por otro la validación con los atractores. Antes de continuar hemos de recordar que el cálculo de la prioridad se produce en base a una matriz de prioridades que, por carecer de datos experimentales, es calculada aleatoriamente. Por otro lado, en la resolución de conflictos, cuando se ha de elegir un minitérmino de la suma de productos que encontramos en el miembro derecho del *pathway* impuesto, como no es posible conocer de antemano cuál de todos ellos es el más adecuado, en el código dicha elección es aleatoria.

Por tanto, por la generación de prioridades y el propio proceso de resolución de conflictos, cada inferencia es única. Para profundizar en el conjunto de alternativas, hemos fijado un número de repeticiones por simulación. Con el ejemplo que hemos desarrollado, se ha marcado un total de 20 repeticiones por objeto *network*. Lo que eleva el total de objetos a analizar hasta 486720.

Como se ha indicado anteriormente, la simulación como tal consta de dos partes bien diferenciadas:

1. Inferencia de los *pathways* y resolución de conflictos.
2. Validación mediante los atractores.

El primer paso se realiza dentro del objeto validación (`class Validation`, `base/validation.py`) asociado al objeto graph. Concretamente dentro del método `def third_validation(self)`, que es el que rige el algoritmo descrito en la Sección 4.5. Dentro de este método se genera un objeto *manager* (`class ConflictsManager`, `graphs/conflicts_theory.py`) que es el que resuelve el algoritmo de resolución de conflictos (Subsección 4.5.1). Concretamente mediante el método `def launch_first_algorithm(self)` en el constructor de dicho objeto. Al resultado se accede mediante el método `def get_solution(self)`.

Por otro lado tenemos el segundo apartado. Como se ha indicado con anterioridad, la implementación en Python de los algoritmos para la obtención de la representación matricial de la red booleana es de 10 a 100 veces más rápida dependiendo de la región del código que ejecutemos y de la complejidad de la red que estemos analizando. Por otro lado, este proceder sigue siendo en el mejor de los escenarios 10 veces más lento que el cálculo de los atractores con el algoritmo de Tarjan.

En consecuencia, aunque todas las funciones del *package* stp han sido validadas y son plenamente operativas, al final nos hemos decantado por emplear el algoritmo de Tarjan codificado en la librería *PyBoolNet*. La validación consiste en, una vez obtenidos los atractores a partir de las expresiones, cotejar el resultado con el conjunto de atractores que deberíamos obtener y en función de la comparación clasificar por similitud el resultado obtenido. Esta validación se produce dentro del método `def third_validation(self)` después de la resolución de conflictos. Al final creamos un objeto resultado (`class Result`, `base/hibrids.py`) con la intención de almacenar los resultados de forma compacta.

Así pues, aquí acaba el proceso de inferencia. Obtenemos por tanto nuestra red de *pathways* compatibles a ser aplicada sobre el mapa precalculado. Son las funciones y métodos comentados en este apartado los puntos clave del código.

Parámetros de la simulación

En este apartado introducimos los parámetros mencionados al principio de la sección. Aunque se fijan antes de ejecutar la simulación, por claridad en la explicación los introducimos al final.

1. `simulations`: número de repeticiones que queremos que se efectúen de cada inferencia.
2. `variants_limit`: número máximo de variantes a calcular.
3. `max_local_iterations`: número máximo de iteraciones fijado para el algoritmo de resolución de conflictos antes de declarar que el procedimiento no converge.
4. `max_global_iterations`: número máximo de iteraciones fijado para el algoritmo de inferencia de *pathways* antes de declarar que el procedimiento no converge.
5. `roles_sets`: lista con *roles sets*. Solo se generarán las redes de aquellas variantes con estos *roles_sets*.
6. `structures`: solo se realizarán inferencias de las redes con las estructuras recogidas en esta lista.
7. `filter_kernel`: estructura en la que integrar los dos parámetros anteriores.
8. `imposed_roles_sets`: *roles sets* que queremos introducir arbitrariamente en la inferencia aunque carezcan de sentido biológico.

6.5 Ejemplo de posprocesado

La filosofía que subyace bajo el posprocesado es diferente a la que se encuentra bajo el proceso de inferencia. Como la introducción de nuevos nodos no viene dada por ninguna regla sino por el criterio del programador, ha de ser manual. Así mismo, el cómputo de las NCBF asociadas a los nodos originales e insertados se produce por separado. Son estas las mayores diferencias dado que en el posprocesado no existe inferencia de *pathways*, solo hay generación de NCBF.

6.5.1 Algoritmo de ejecución

Introducción de los datos iniciales

Primero introducimos los nuevo nodos en la representación tabulada de los nodos originales del grafo. Como se indica en el código a continuación.

```
import pandas as pd

# Original graph
index = ['A', 'B', 'C', 'D']
columns = ['activators', 'inhibitors']
content = [[['D', 'F'], ['']], [['A'], ['E']], [['E'], ['A']], [['B'], ['C']]]
original_data = pd.DataFrame(data=content, index=index, columns=columns)
```

De igual manera, indicamos en forma tabular de qué variables depende cada uno de los nuevos nodos. Dado que no conocemos ni su estructura ni rol, es la única información a priori que podemos introducir.

```
import pandas as pd

# Unfixed conflicts graphs
index = ['E', 'F']
tag = 'variables'
columns = [tag]
content = [[['A', 'E']], [['A', 'D', 'E', 'F']]]
unfixed_conflicts_data = pd.DataFrame(data=content, index=index, columns=columns)
```

Generación del conjunto de redes a analizar

Hasta ahora, para los nodos originales hemos introducido las variables de las que dependen y la estructura de la red, por lo que calculamos las NCBF que responden a ambos parámetros con la función `def ncbfCalc(data=original_data, tags=tags)`. Así obtenemos un conjunto de subredes constituidas únicamente por los nodos originales. No podemos repetir la operación para los nuevos nodos dado que solo introdujimos las variables, carecemos de suficiente información. Por tanto, con la función `def conflict_ncbfCalc(variables=unfixed_conflicts_data, tag=tag)` primero calculamos todas las posibles estructuras que pudieran manifestar los nuevos nodos, para después calcular todas las posibles NCBF por estructura. Y así es como generamos el conjunto de subredes que representan a estos nodos.

Una vez que tenemos los dos conjuntos de subredes, el de los nodos originales y el de los nuevos, los combinamos todos con todos y fusionamos el total de redes. Así obtenemos el conjunto de redes a analizar, cada una de ellas única y constituida con las estructuras de todos los nodos.

Posprocesado

Así, con el conjunto definitivo de redes llamamos a la función `def post_process(*args)`. Esta función lo único que hace es inferir las expresiones de cada una de las redes, calcular los atractores y comparar con los que estamos buscando. El problema radica en el hecho de que los atractores que nosotros buscamos están sobre 4 nodos mientras que estamos analizando redes con 6 nodos. Por ello, en esta función se implementa el proceso de proyección que describimos en la teoría, es decir, no realizamos la comparación directamente.

Obtenemos la lista de redes cuya proyección se ajusta a los atractores que hemos fijado. A diferencia de en el ejemplo anterior, dado que no hemos generado un gran número de parámetros, no hemos definido un objeto resultado. Es por tanto una lista de diccionarios que contienen la estructura de la red y los atractores (extendidos) de las redes que han pasado la criba.

Parte II

Presupuesto

Capítulo 7

Presupuesto

En este capítulo exponemos un resumen de las cuentas detrás de este proyecto. No obstante, dado su carácter teórico, se han tenido que adaptar todos los elementos de un documento presupuesto al uso. Así pues, el presente escrito es equiparable a otros proyectos de distinta naturaleza.

7.1 Cuadro de precios de mano de obra

El salario por hora, tanto del ingeniero biomédico júnior como del profesor supervisor se han calculado en base a la Ley de Presupuestos para el ejercicio 2019, publicada en la página web de la *Universitat Politècnica de València*. El sueldo de un ingeniero biomédico júnior se equipara al de ayudante. De la misma manera se ha supuesto que un mes tiene 20 días laborales con una jornada de 8 horas.

Tabla 7.1: Salario por hora del profesor supervisor y del ingeniero biomédico júnior.

Unidad	Descripción	Precio (€)
h	Ingeniero biomédico	8.88
h	Profesor supervisor	50.14

7.2 Cuadro de precios de maquinaria

No se ha empleado maquinaria en este proyecto. Entendiendo por maquinaria cualquier tecnología que requiera de personal especializado o de un aprendizaje previo. Como se expone en el apartado materiales, únicamente hemos empleado un ordenador común así como *software* de uso general como lo es MATLAB, PyCharm y RStudio.

7.3 Cuadro de precios de materiales

En este apartado describimos el precio de todo el *software* y la bibliografía consultada durante el proyecto. Hay que destacar que la licencia de MATLAB es la licencia académica anual. Así mismo, para Python se ha empleado la versión mantenida por la comunidad de PyCharm, de ahí que su coste sea 0 €. Similar es lo que ocurre con RStudio, al ser *software* gratuito su coste es 0 €. Por otro lado, las tres últimas entradas se corresponden con los manuales publicados por Tatsuya Akutsu (Akutsu, 2018) y Daizhan Cheng, Hongsheng Qi y Zhiqiang Li (Cheng y col., 2011) respectivamente. Finalmente, dado el número de artículo leídos, optamos por dar un precio medio de los artículos leídos de la editorial Elsevier, el resto, por ser de revistas *Open Access* no han supuesto coste alguno. Así mismo, el único componente del equipo informático que ha condicionado el trabajo ha sido el procesador, AMD FX(tm)-8350 *Eight Core*. El total del equipo es valorado en 800 €.

Tabla 7.2: Cuadro de precios de materiales

Unidad	Descripción	Precio (€)	Cantidad	Factor de amortización	Coste (€)
ud	Equipo informático	800	1	0	0
ud	Licencia de MATLAB	250	1	6/12	125
ud	Licencia de PyCharm	0	1	6/12	0
ud	Licencia de RStudio	0	1	6/12	0
ud	*Algorithms for Analysis, Inference, and Control of Boolean Networks*	97.87	1	1	97.87
ud	*Analysis and Control of Boolean Networks*	117.89	1	1	117.89
ud	Artículo de Elsevier	39.95	3	1	159.8
				Total	500.56

Para calcular los factores de amortización, se ha partido del hecho de que el desarrollo del proyecto ha sido de forma intermitente a lo largo de 6 meses. Por tanto, para el *software*, el factor de amortización es de 6/12. Este enfoque es coherente dado que la licencia empleada es la licencia académica anual de MATLAB. Por otro lado, el factor de amortización para los libros se ha fijado en 1, dado que se han adquirido y empleado durante el proyecto. Este tipo de contenidos en un proyecto de ingeniería no se encontrarían en este apartado, aunque dada la naturaleza de investigación de este proyecto es correcto ubicarlos en esta sección. En sentido estricto forman algunos de los pilares sobre los que se ha desarrollado todo el proyecto.

Finalmente, el factor de amortización del equipo informático empleado se ha fijado en 0, dado que el procesador, el único componente de relevancia dado que no se ha empleado la tarjeta gráfica para los cálculos, fue lanzado al mercado en septiembre del 2014. Asumiendo un tiempo de amortización para componentes electrónicos de 5 años, dicho procesador está completamente amortizado en el momento en el que comenzó este proyecto.

7.4 Cuadro de precios unitarios

7.4.1 Definición del proyecto

Tabla 7.3: Reuniones dedicadas para fijar el alcance del proyecto así como el modo de trabajo a lo largo del mismo.

Unidad	Descripción	Precio
€	Reunión inicial	59.02
€	Reuniones de control	531.18
€	Reuniones para el diseño de algoritmos	531.18

7.4.2 Investigación bibliográfica

Tabla 7.4: Tareas de investigación y aprendizaje de todo el conocimiento necesario para completar el proyecto.

Unidad	Descripción	Precio
€	Investigación del estado del arte	53.28
€	Investigación en la Teoría de Redes Booleanas	142.08
€	Investigación en la Teoría de NCBF	106.56
€	Investigación en Cálculo lógico y STP	106.56

7.4.3 Desarrollo de algoritmos

Tabla 7.5: Tiempo dedicado al desarrollo teórico de los distintos algoritmos empleados durante el proyecto.

Unidad	Descripción	Precio
€	Algoritmo para la definición de redes	17.76
€	Algoritmo para la creación de redes	53.28
€	Algoritmo para la validación de redes	17.76

7.4.4 Implementación en software

Tabla 7.6: Tiempo dedicado a la codificación en *software* de los algoritmos mencionados anteriormente.

Unidad	Descripción	Precio
€	Algoritmo para la definición de redes	88.8
€	Algoritmo para la creación de redes	177.76
€	Algoritmo para la validación de redes	444
€	Validación del *software*	177.6

7.4.5 Análisis de redes biológicas

Tabla 7.7: Tareas de análisis de los resultados obtenidos así como comparación con la bibliografía existente.

Unidad	Descripción	Precio
€	Análisis de una red de ejemplo	88.8
€	Investigación sobre la EMT	177.6
€	Análisis de la red de la EMT	177.6

7.4.6 Redacción y defensa del TFG

Tabla 7.8: Tareas destinadas a la redacción de este documento.

Unidad	Descripción	Precio
€	Redacción de la memoria	541.68
€	Redacción del documento presupuesto	53.28
€	Creación de la presentación	53.28
€	Preparación de la presentación	17.76

7.5 Cuadro de precios descompuestos

7.5.1 Definición del proyecto

Reunión inicial

Tabla 7.9: Reunión realizada al principio del proyecto con la intención de marcar el alcance del mismo así como la forma de trabajar de ambas partes.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1
h	Profesor supervisor	horas totales:	1

Reunión de control

Tabla 7.10: Reunión periódica destinada a controlar la evolución del proyecto.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1
h	Profesor supervisor	horas totales:	1

Reunión de diseño

Tabla 7.11: Reunión periódica destinada a fijar la creación de algoritmos así como la estructura del *software*.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1
h	Profesor supervisor	horas totales:	1

7.5.2 Investigación bibliográfica

Investigación de la Teoría de Redes Booleanas

Tabla 7.12: Tiempo destinado al aprendizaje de la naturaleza y construcción de esta variante de modelado matemático.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Investigación de las NCBF

Tabla 7.13: Tiempo destinado al aprendizaje de la naturaleza de esta familia de funciones lógicas.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Investigación del Cálculo lógico y el STP

Tabla 7.14: Tiempo destinado al aprendizaje de las herramientas empleadas en cálculo lógico, concretamente el STP.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Investigación del Estado del arte

Tabla 7.15: Tiempo destinado a la investigación de otras herramientas alternativas de modelado de redes, así como de los casos previos de implementación de redes booleanas. También destinado a la información sobre la situación actual en la clasificación del cáncer de vejiga.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

7.5.3 Desarrollo de algoritmos

Algoritmo de definición de redes

Tabla 7.16: Tiempo destinado a la creación del algoritmo dedicado a la definición, nodo a nodo, de cuáles de los nodos vecinos son activadores y cuáles son inhibidores.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Algoritmo de creación de redes

Tabla 7.17: Tiempo destinado a la creación del algoritmo dedicado a la definición de las distintas estructuras que pudiera presentar una nodo, dados un conjunto de nodos activadores e inhibidores.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Algoritmo de validación de redes

Tabla 7.18: Tiempo destinado a la creación del algoritmo de validación de las redes a través de los atractores previamente conocidos. Primero destinado al desarrollo del algoritmo propio basado en el STP y posteriormente a la implementación del algoritmo de Tarjan.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

7.5.4 Implementación de software

Algoritmo de definición de redes

Tabla 7.19: Tiempo destinado a la implementación del algoritmo de definición de redes descrito con anterioridad.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Algoritmo de creación de redes

Tabla 7.20: Tiempo destinado a la implementación del algoritmo de creación de redes definido con anterioridad.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Algoritmo de validación de redes

Tabla 7.21: Tiempo destinado a la implementación del algoritmo de validación de redes descrito con anterioridad.

Unidad	Descripción		Medición
h	Ingeniero biomédic	horas totales:	1

Validación del software

Tabla 7.22: Tiempo destinado a *testear* el *software* mediante ejemplos simplificados.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

7.5.5 Análisis de redes biológicas

Desarrollo y análisis del ejemplo

Tabla 7.23: Tiempo dedicado a la búsqueda de un ejemplo representativo del procedimiento desarrollado.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Investigación bibliográfica sobre el modelo de la EMT

Tabla 7.24: Tiempo dedicado a la búsqueda de información sobre el modelo de la EMT. El segundo caso práctico.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Análisis comparativo del modelo obtenido de la EMT

Tabla 7.25: Tiempo dedicado a la comparación entre los resultados obtenidos para el modelo de la EMT y los encontrados en la bibliografía.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

7.5.6 Redacción y defensa del TFG

Redacción de la memoria

Tabla 7.26: Tiempo dedicado a la redacción de la memoria.

Unidad	Descripción	Medición	
h	Ingeniero biomédico	horas totales:	1

Redacción del resto de documentos

Tabla 7.27: Tiempo dedicado a la redacción del resto de documentos.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Creación de la presentación

Tabla 7.28: Tiempo dedicado a la creación de la presentación.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

Preparación de la presentación

Tabla 7.29: Tiempo dedicado al ensayo de la presentación.

Unidad	Descripción		Medición
h	Ingeniero biomédico	horas totales:	1

7.6 Mediciones

7.6.1 Definición del proyecto

Tabla 7.30: Reuniones dedicadas para fijar el alcance del proyecto así como el modo de trabajo a lo largo del mismo.

Unidad	Descripción		Medición
h	Reunión inicial	horas totales:	1
h	Reuniones de control	horas totales:	9
h	Reuniones para el diseño de algoritmos	horas totales:	9

7.6.2 Investigación bibliográfica

Tabla 7.31: Tareas de investigación y aprendizaje de todo el conocimiento necesario para completar el proyecto.

Unidad	Descripción		Medición
h	Investigación del estado del arte	horas totales:	6
h	Investigación en la Teoría de Redes Booleanas	horas totales:	16
h	Investigación en la Teoría de NCBF	horas totales:	12
h	Investigación en Cálculo lógico y STP	horas totales:	12

7.6.3 Desarrollo de algoritmos

Tabla 7.32: Tiempo dedicado al desarrollo teórico de los distintos algoritmos empleados durante el proyecto.

Unidad	Descripción		Medición
h	Algoritmo para la definición de redes	horas totales:	2
h	Algoritmo para la creación de redes	horas totales:	6
h	Algoritmo para la validación de redes	horas totales:	2

7.6.4 Implementación en software

Tabla 7.33: Tiempo dedicado a la codificación en *software* de los algoritmos mencionados anteriormente.

Unidad	Descripción		Medición
h	Algoritmo para la definición de redes	horas totales:	10
h	Algoritmo para la creación de redes	horas totales:	20
h	Algoritmo para la validación de redes	horas totales:	50
h	Validación del *software*	horas totales:	20

7.6.5 Análisis de redes biológicas

Tabla 7.34: Tareas de análisis de los resultados obtenidos así como comparación con la bibliografía existente.

Unidad	Descripción		Medición
h	Análisis de una red de ejemplo	horas totales:	10
h	Investigación sobre la EMT	horas totales:	20
h	Análisis de la red de la EMT	horas totales:	20

7.6.6 Redacción y defensa del TFG

Tabla 7.35: Tareas destinadas a la redacción de este documento.

Unidad	Descripción		Medición
h	Redacción de la memoria	horas totales:	61
h	Redacción del documento presupuesto	horas totales:	6
h	Creación de la presentación	horas totales:	6
h	Preparación de la presentación	horas totales:	2

7.7 Presupuestos parciales

Con los datos de los apartados anteriores obtenemos.

Tabla 7.36: Presupuestos parciales

Unidad	Descripción	Medición	Coste (€)
h	Definición del proyecto	1	1121.38
h	Investigación bibliográfica	1	408.48
h	Desarrollo de algoritmos	1	88.8
h	Implementación en *software*	1	888
h	Análisis de redes biológicas	1	444
h	Redacción y defensa del TFG	1	666
		Total	3616.66

7.8 Presupuesto de ejecución por contrata

Al final obtenemos el presupuesto de ejecución por contrata. El beneficio industrial se ha fijado en base al Reglamento General de la Ley de Contratos de las Administraciones Públicas (Real Decreto 1098/2001).

Tabla 7.37: Presupuesto de ejecución por contrata

Descripción	Coste (€)
Presupuesto de ejecución material	3616.66
Beneficio industrial (6 %)	217
Gastos (13 %)	470.17
Total antes de impuestos	4303.83
IVA (21 %)	903.8
Final	5207.63

Parte III

Anexos

Demostraciones

En este anexo exponemos las demostraciones de las afirmaciones sobre las que se han construido el software y los algoritmos empleados. Al haber sido deducidas y aplicadas durante el desarrollo del proyecto, son ***una aportación original de este trabajo****. Hasta donde conoce el autor, ninguna de dichas demostraciones* ***ha sido publicada*** *con anterioridad en el momento de entregar esta memoria. Por tanto, merecen formar parte de ella. Es por esta razón que se les ha dedicado un anexo con la intención de desarrollarlas en profundidad.*

Teoremas sobre sobre la construcción y modificación de funciones

7.8.1 Teorema 1

Enunciado

Toda función booleana puede ser expresada como una combinación de NCBF.

Demostración

Nosotros partimos de la base de que toda función booleana se puede expresar mediante producto de maxitérminos, lo que se conoce como *conjunctive normal form* (CNF), o mediante suma de minitérminos, es decir, la *disjunctive normal form* (DNF).

Tabla 7.38: Mapa de Karnaugh de una función f.

AB \ CD	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	0	1	1
10	1	1	1	1

Tomando como ejemplo la función expresada en la Tabla 7.38. Su CNF (7.1) y DNF (7.2) se muestran a continuación.

$$f = (A \vee C) \wedge (\neg B \vee C) \wedge (A \vee D) \tag{7.1}$$

$$f = A \wedge \neg B \vee C \wedge D \vee A \wedge C \tag{7.2}$$

En otras palabras, toda función booleana se puede expresar como suma de productos o como producto de sumas. Esta afirmación es inmediata, una vez que se tiene la tabla de verdad de la propia función o, como en este caso, su mapa de Karnaugh.

Ahora volvamos a las NCBF mediante la Expresión 4.6:

$$f(x_1, x_2, ...x_n) = M_1(M_2(...(M_{r-1}(M_r \oplus 1) \oplus 1)... \oplus 1) \oplus b$$

Ciñéndonos a su definición, expuesta de forma más exhaustiva en el Capítulo 4, podemos concluir que las distintas M_i son productorios de la forma $(x_{i_j} \oplus a_{i_j})(x_{i_j+1} \oplus a_{i_j+1})...(x_{i_n} \oplus a_{i_n})$.

De forma intuitiva, basta con mantener en la cabeza que el valor con el que se realiza la operación módulo 2 en cada uno de los factores del productorio, es el valor que queremos que manifieste la variable en cuestión para obtener en f el valor marcado por la capa, por M_i. El valor que muestra f, como se ha expuesto en capítulos anteriores, viene marcado por la primera capa (M_1). Luego, al ser la expresión un conjunto de negaciones anidadas, en las capas internas el valor de b va implícito dentro de la propia expresión. Es decir, si $b = 1$ para M_1, aunque no se declare explícitamente, $b = 0$ para M_2, $b = 1$ para M_3 y así sucesivamente. Es esta idea uno de los pilares sobre los que se construye el algoritmo para construir de forma eficiente NCBF.

Tomemos ahora un minitérmino cualquiera, no expresado en notación de algebra de módulo 2 sino con la notación que venimos empleando a lo largo del trabajo.

$$m = x_1 \wedge \neg x_2 \wedge x_3 \tag{7.3}$$

En la Expresión 7.3 podemos apreciar que el minitérmino, la función m, depende de 3 variables y que, en el caso de que X_1 o X_3 presenten el valor de 0 o X_2 el de 1, m valdrá 0. En otras palabras, es posible expresar m en notación de algebra de módulo 2 como:

$$m = (x_1 \oplus 0)(x_2 \oplus 1)(x_3 \oplus 0) \oplus 0$$

Esta formulación es extensible a cualquier minitérmino, lo único que variará serán los valores canalizantes (Capítulo 3) de cada variable. Es inmediato percatarse de que esta expresión se corresponde con la Ecuación 4.6. Donde solo hay una única M_i, $M_1 = (x_1 \oplus 0)(x_2 \oplus 1)(x_3 \oplus 0)$ y $b = 0$. Por lo tanto, cualquier minitérmino es una NCBF.

Centremos la atención ahora en un maxitérmino cualquiera:

$$M = x_1 \vee \neg x_2 \vee x_3$$

Análogamente a como hemos procedido con el minitérmino, nos damos cuenta de que podemos expresarlo de la forma:

$$M = (X_1 \oplus 1)(X_2 \oplus 0)(X_3 \oplus 1) \oplus 1$$

Donde una vez más tenemos una única capa, M_1, con un valor de $b = 1$.

Como resultado, observamos que tanto los maxitérminos como los minitérminos son NCBF. Llegados a este punto, si cualquier función se puede expresar como combinación de minitérminos (o maxitérminos). Hemos demostrado lo que se pretendía: *cualquier función booleana se puede expresar como combinación de NCBF*. Dado que maxitérminos y minitérminos son NCBF.

7.8.2 Teorema 2

Enunciado

Dadas dos funciones booleanas f y Q, existe una transformación K, tal que $Q = K(f)$. Donde K es de la forma:

$$Q = K(f) = R \wedge f \vee D$$

Donde, $R = \bigwedge_i \bigwedge_j \neg M_{ij}$, $D = \bigvee_m \bigvee_n M_{mn}$ y M_{ij} y M_{mn} son NCBF. Recordemos que en esta expresión, $\bigwedge$ y $\bigvee$ son los equivalentes lógicos de $\prod$ y $\sum$. Es decir, $\bigwedge$ representa un productorio lógico ($\bigwedge_{i=1}^{k} = x_1 \wedge x_2 \wedge x_3 ... \wedge x_i$) y $\bigvee$ un sumatorio lógico ($\bigvee_{i=1}^{k} = x_1 \vee x_2 \vee x_3 ... \vee x_k$).

Demostración

Tomemos dos funciones f y Q. Representadas en la Tabla 7.39. Cuyas expresiones se dan en las Ecuaciones 7.4 y 7.5.

Tabla 7.39: Mapas de las funciones lógicas de ejemplo.

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	1	0	0
11	0	1	1	1
10	1	1	1	0

(a) f

AB \ CD	00	01	11	10
00	1	1	1	0
01	0	1	1	0
11	1	1	0	1
10	0	1	1	1

(b) Q

$$f = (\neg C \wedge D) \vee (A \wedge D) \vee (A \wedge \neg B \wedge \neg D) \vee (A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg D) \vee (\neg A \wedge \neg B \wedge C) \quad (7.4)$$

$$Q = (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge D) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge \neg D) \vee (A \wedge \neg B \wedge D) \vee (A \wedge \neg B \wedge D) \vee (A \wedge \neg B \wedge C) \quad (7.5)$$

Dada una función cualquiera S es inmediato que $S \vee 1 = 1$, independientemente del valor de S. De igual manera, $S \vee 0 = S$. Esto es así porque una adición, OR lógico, entre funciones equivale

a realizar un OR casilla a casilla entre los mapas de ambas funciones. O lo que es lo mismo, al realizar un OR entre dos funciones, el *support* de la función resultante es la unión de los *support* de las dos funciones por separado: $supp(f \vee Q) = supp(f) \cup supp(Q)$ [1].

Volviendo al concepto de minitérmino que introdujimos en apartados anteriores, ¿cómo sería el mapa de Karnaugh de un minitérmino? Observemos el ejemplo del minitérmino $A \wedge \neg B \wedge C \neg D$ en el mapa de la Tabla 7.40.

Tabla 7.40: Mapa de Karnaugh del minitérmino $A \wedge \neg B \wedge C \neg D$.

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	1

De esta forma, la operación $r = f \vee (A \wedge \neg B \wedge C \wedge \neg D)$, sobre los mapas se representa en la Figura 7.1.

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	1	0	0
11	0	1	1	1
10	1	1	1	0

(a) f

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	1

(b) $A \wedge \neg B \wedge C \neg D$

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	1	0	0
11	0	1	1	1
10	1	1	1	1

(c) $f \vee (A \wedge \neg B \wedge C \neg D)$

Figura 7.1: Adición representada sobre los mapas de Karnaugh.

Por otro lado, cuando empleamos el producto (AND lógico) tenemos que $S \wedge 0 = 0$ y $S \wedge 1 = S$. Los papeles se invierten. En otras palabras, el suplemento de un producto de funciones es la intersección de los suplementos de los factores: $supp(f \wedge Q) = supp(f) \cap supp(Q)$. No obstante, la razón no está tan clara como en apartados anteriores. Si cogemos un minitérmino cualquiera, por ejemplo $\neg A \wedge B \wedge \neg C \wedge \neg D$, sobre los mapas logramos el efecto que se muestra en la Figura 7.2.

De forma que, análogamente a como lo expusimos para el caso de la adición tenemos lo que se muestra en la Figura 7.3.

Es así como, valiéndonos de la adición de minitérminos, podemos introducir en una función todos los 1 que sean necesarios. De la misma manera, empleando productos con minitérminos negados, podemos introducir todos los 0 que deseemos. Como se ha demostrado en apartados

[1] Recordemos que definimos el *support* de una función f como el conjunto de combinaciones de los argumentos de entrada que consiguen que $f = 1$.

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	0

(a) $\neg A \wedge B \wedge \neg C \wedge \neg D$

AB \ CD	00	01	11	10
00	1	1	1	1
01	0	1	1	1
11	1	1	1	1
10	1	1	1	1

(b) $\neg(\neg A \wedge B \wedge \neg C \wedge \neg D)$

Figura 7.2: Efecto de la negación de un minitérmino sobre su mapa de Karnaugh

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	1	0	0
11	0	1	1	1
10	1	1	1	0

(a) f

AB \ CD	00	01	11	10
00	1	1	1	1
01	0	1	1	1
11	1	1	1	1
10	1	1	1	1

(b) $\neg(\neg A \wedge B \wedge \neg C \wedge \neg D)$

AB \ CD	00	01	11	10
00	0	1	1	1
01	0	1	0	0
11	0	1	1	1
10	1	1	1	0

(c) $f \wedge \neg(\neg A \wedge B \wedge \neg C \wedge \neg D)$

Figura 7.3: Producto representado sobre los mapas de Karnaugh.

anteriores, todos los minitérminos son NCBF. De tal manera que para lograr el resultado anterior nos es indiferente hablar de minitérminos que de NCBF, con salvedad de que las NCBF nos permiten definir de forma precisa las regiones que queremos modificar.

Así es como llegamos a la siguiente expresión:

$$Q = K(f) = (\bigwedge_i \bigwedge_j \neg M_{ij}) \wedge f \vee (\bigvee_m \bigvee_n M_{mn})$$

Donde M_{ij} y M_{mn} son NCBF y f y Q dos funciones lógicas cualquiera.

Esto es así dado que valiéndonos de NCBF podríamos definir completamente el mapa de una función sin importar el sustrato sobre el que la desarrolláramos. Más aún, si eliminamos el primer sumando de la expresión, lo que sería equivalente a multiplicar la función f por los cuatro minitérminos negados de A, B, C y D, nos encontramos con el Teorema 1: toda función booleana se puede expresar como combinación de NCBF. Es así como queda demostrado el enunciado que introducíamos al principio de este apartado: *dadas dos funciones booleanas f y Q, existe una transformación K, tal que $Q = K(f)$. Donde K es de la forma descrita en la Ecuación 7.8.2.*

Demostración de por qué no es posible obtener una red compuesta de no NCBF a partir de la proyección de una red compuesta únicamente por NCBF.

Al final, el objetivo de este proyecto es obtener redes que sean capaces de representar la información introducida en grafo de una manera biológicamente coherente. Estas redes se encuentran constituidas a partir de nodos, que a su vez son definidos por funciones booleanas. Las cuales pueden ser NCBF o no. Aunque como ya se ha venido argumentando a lo largo del trabajo, existe un alto número de probabilidades de que lo sean, o de que sean funciones muy similares a NCBF.

Así mismo, este trabajo surge de la intención de hibridar la teoría expuesta en Layek, 2012 con la existente de NCBF (Li y col., 2013). Una manera de introducir los conflictos, tal como los concibe Layek, 2012, es mediante nodos externos responsables de estos conflictos. Basta con observar las funciones que obtiene Layek, 2012 para percibir que esas funciones no tienen porqué ser NCBF. Al mismo tiempo, si introducimos los conflictos mediante nodos añadidos, vamos a obtener una red integrada por más agentes de los inicialmente introducidos. Lo que no tiene porqué ser erróneo ni carecer de interés, aunque no es lo ideal.

Es por eso que es necesario realizarnos la pregunta, ¿sería posible obtener una red constituida de funciones no NCBF a partir de la proyección de una red constituida por NCBF? La respuesta es que no. Es en este apartado donde demostramos el motivo. Lo primero que tenemos que fijar es el significado del término proyección. La proyección de una red A, con n nodos, se define como la transformación que tiene que sufrir A para convertirse en otra red B de p nodos donde $p < n$. En B, las expresiones de todos los nodos han de mantenerse inalteradas salvo por el hecho de que han desaparecido las dependencias de los nodos eliminados. Por sustitución con otras variables o con valores constantes. En las expresiones de la Figura 7.4 se desarrolla un ejemplo de proyección de red.

$$\begin{aligned} A &= F \wedge B \wedge D \vee C \\ B &= E \wedge A \\ C &= A \wedge E \vee C \wedge F \\ D &= F \wedge A \vee B \\ E &= A \wedge C \vee D \\ F &= E \vee A \end{aligned}$$

(a) Expresión sin proyectar.

$$\begin{aligned} A &= F \wedge B \wedge D \vee C \\ B &= E \wedge A \\ C &= A \wedge E \vee C \wedge F \\ D &= F \wedge A \vee B \\ E &= A \wedge C \vee D \\ F &= 1 \end{aligned}$$

(b) Expresión con valores constantes.

$$\begin{aligned} A &= B \wedge D \vee C \\ B &= (A \wedge C \vee D) \wedge A \\ C &= A \wedge (A \wedge C \vee D) \vee C \\ D &= A \vee B \end{aligned}$$

(c) Expresión proyectada.

Figura 7.4: Proceso de proyección de una red de 6 nodos en otra de 4.

Con este procedimiento es posible obtener múltiples redes, no todas ellas biológicamente coherentes, como se desarrollará más adelante. No obstante, la pregunta que nos hacemos es sobre la naturaleza de estas redes, sobre si se ajustan matemáticamente o no al esquema fijado por las NCBF. Una vez más, recurrimos a la Expresión 4.6.

$$f(x_1, x_2, ...x_n) = M_1(M_2(...(M_{r-1}(M_r \oplus 1) \oplus 1)... \oplus 1) \oplus b$$

Para facilitar la exposición fijamos un ejemplo $f = M_1[M_2[M_3[M_4 \oplus 1] \oplus 1] \oplus 1] \oplus 1$. Donde $M_1 = (x_1 \oplus 0)$, $M_2 = (x_2 \oplus 0)(x_3 \oplus 1)$, $M_3 = (x_4 \oplus 1)(x_5 \oplus 0)(x_6 \oplus 1)$ y $M_4 = (x_7 \oplus 0)(x_8 \oplus 1)$. Con este ejemplo explicamos los posibles casos que nos podemos encontrar y cómo pueden influir en la Expresión 4.6.

Anulación de una capa

Supongamos que $x_4 = 1$ o $x_5 = 0$ o $x_6 = 1$. En cualquiera de estos supuestos Tenemos que $M_3 = 0$. Y por lo tanto, si sustituimos en nuestro ejemplo:

$$f = M_1[M_2[0 \oplus 1] \oplus 1] \oplus 1 = M_1[M_2 \oplus 1] \oplus 1 \tag{7.6}$$

Es decir, en el supuesto de que sustituyamos una variable por un valor constante que se encuentre entre los valores canalizantes de la capa (siguiendo la terminología expuesta en Li y col., 2013) vamos a conseguir eliminar de la expresión la capa en la que se encuentra esa M_i así como todas las que se encuentran por debajo. En cualquier caso, como demuestra la Expresión 7.6, seguimos teniendo una NCBF.

Imposición de valores no canalizantes

Siguiendo con el ejemplo anterior supongamos que $x_4 = 0, x_5 = 1$ y $x_6 = 0$. En este caso ninguna de las variables presenta el valor canalizante, en consecuencia $M_3 = 1$ y por lo tanto tenemos que:

$$f = M_1[M_2[[M_4 \oplus 1] \oplus 1] \oplus 1] \oplus 1 = M_1[M_2M_4 \oplus 1] \oplus 1 = M_1[M_{24} \oplus 1] \oplus 1$$

Es decir, M_2 y M_4 pasan a tener la misma prioridad y, por lo tanto, a constituir una nueva capa M_{24}. Esto tiene sentido dado que M_2 y M_4 tienen el mismo valor canalizado (b) y ha desaparecido M_3, la capa intermedia de valor canalizado contrario. De manera que la transformación que algebraicamente se deduce rápidamente, tiene sentido dentro de la teoría que venimos exponiendo en los capítulos anteriores.

Ahora supongamos que no llenamos toda la capa con valores no canalizantes, sino que únicamente $x_4 = 0$. En este caso, $x_4 \oplus 1 = 1$ y como resultado $M_3 = (x_5 \oplus 0)(x_6 \oplus 1)$. Simplemente hemos eliminado una de las variables pero la estructura de la función permanece intacta. Por lo tanto, sustituyendo con valores no canalizantes siempre vamos a acabar en una NCBF. Podemos concluir que mediante sustitución de valores constantes no vamos a conseguir funciones no NCBF a partir de una NCBF. Podemos modificar su estructura en tanto que fusionemos o eliminemos capas. Aunque en ningún caso nos vamos a salir del marco fijado en la Expresión 4.6. No obstante, ¿qué ocurriría si sustituimos expresiones? ¿Qué ocurriría si simplificáramos el conjunto?

Sustitución de NCBF en NCBF

Supongamos dos NCBF A y B cuyas expresiones encontramos a continuación.

$$A = (x_1 \oplus 0)[(x_2 \oplus 1) \oplus 1] \oplus 1$$
$$B = (x_3 \oplus 1)[(A \oplus 1) \oplus 1] \oplus 1$$

Basta con sustituir la expresión de A en B para obtener:

$$B = (x_3 \oplus 1)[(x_1 \oplus 0)[(x_2 \oplus 1) \oplus 1] \oplus 1] \oplus 1$$

Esta expresión es inmediata y surge de la anulación de dos negaciones anidadas consecutivas. Sin embargo, ahora supongamos el otro caso posible, en el que no hay dos negaciones consecutivas, para ello cambiamos la estructura de A de tal forma que:

$$A = (x_1 \oplus 0)[(x_2 \oplus 1) \oplus 1]$$

Si sustituimos en B y simplificamos obtenemos la siguiente expresión:

$$B = (x_3 \oplus 0)[((x_1 \oplus 0)[(x_2 \oplus 1) \oplus 1] \oplus 1) \oplus 1] \oplus 1$$
$$B = (x_3 \oplus 1)(x_1 \oplus 0)(x_2 \oplus 1) \oplus 0$$

Como se puede apreciar en estos dos casos, no podemos en ningún supuesto obtener un función no NCBF mediante sustitución entre NCBF. Directamente o simplificando, llegamos a la expresión que pone de manifiesto que la función resultante se ajusta al esquema fijado por Li y col., 2013, de forma que siempre el resultado es otra NCBF.

Así pues, hemos demostrado lo que pretendíamos al principio de esta sección, por sustitución, de expresiones o de constantes, no es posible obtener funciones no NCBF a partir de otra NCBF. Por lo tanto, la proyección de NCBF, tal y como está definida en líneas anteriores, no puede dar lugar a funciones no NCBF. No obstante, tras llevar tiempo trabajando esta idea, queremos dejar abierta la puerta a otras definiciones de proyección, tal vez más sofisticadas, que consigan la obtención de funciones no NCBF a partir de la proyección de NCBF. Una idea que resulta de interés dado el conjunto de procedimientos expuestos en este proyecto.

Resultado del *software*

En este apartado exponemos todos los datos, obtenidos por el propio software, que condujeron a la representación de la red expuesta en el Capítulo 5.

Topografía de la red

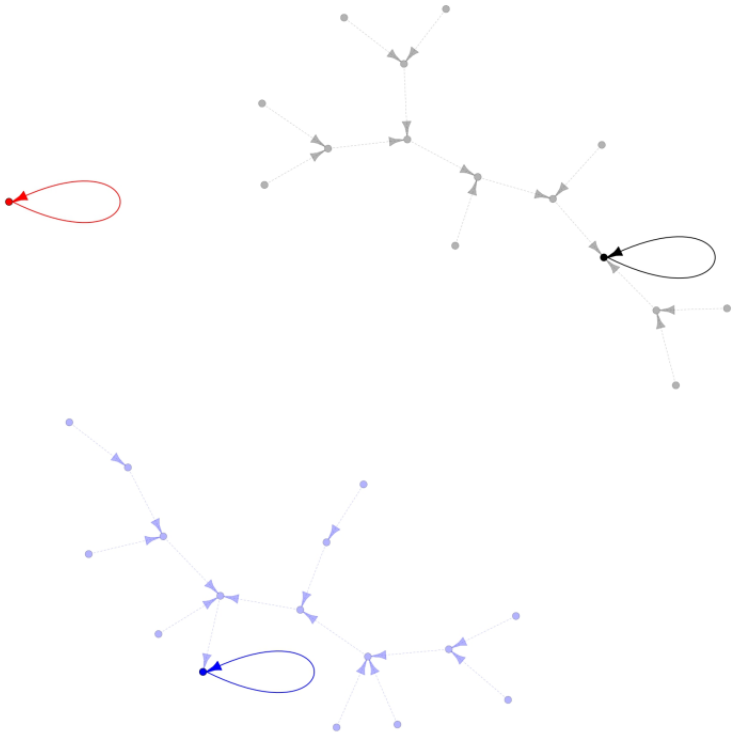


Figura 7.5: Topografía de la red resultado. Cada punto de color se corresponde con un estado de la red booleana (una combinación de valores para las variables de la red). Los estados en azul forman la cuenca del atractor 00101 (epitelial), los estados en negro la cuenca del atractor 11010 (mesenquimal) y los estados en rojo forman la cuenca del atractor 10101 (híbrido).

Listado final de *pathways*

El conjunto final de *pathways* sin conflictos inferido por el *software*.

$$\neg T \wedge I \xrightarrow{1:1,1} S$$
$$I \wedge S \wedge T \wedge Z \wedge D \xrightarrow{1:1,1} S$$
$$\neg S \wedge I \wedge T \wedge Z \xrightarrow{1:1,1} S$$
$$\neg T \wedge I \wedge S \wedge Z \xrightarrow{1:1,1} S$$
$$\neg T \wedge D \wedge I \wedge S \wedge Z \xrightarrow{1:1,1} S$$
$$\neg I \wedge Z \wedge D \wedge \neg T \xrightarrow{1:1,1} S$$
$$\neg T \wedge Z \wedge D \xrightarrow{1:1,1} S$$
$$Z \wedge D \wedge \neg S \xrightarrow{1:1,1} S$$
$$\neg I \wedge T \wedge S \xrightarrow{1:1,0} S$$
$$\neg Z \wedge T \wedge S \wedge \neg D \xrightarrow{1:1,0} S$$
$$T \wedge S \wedge \neg D \xrightarrow{1:1,0} S$$
$$\neg I \wedge T \wedge \neg D \xrightarrow{1:1,0} S$$
$$\neg Z \wedge T \xrightarrow{1:1,0} S$$
$$I \wedge S \wedge T \wedge Z \wedge \neg D \xrightarrow{1:1,0} S$$
$$\neg Z \wedge \neg I \wedge T \xrightarrow{1:1,0} S$$
$$\neg I \wedge S \wedge \neg D \xrightarrow{1:1,0} S$$
$$\neg Z \wedge \neg I \wedge S \xrightarrow{1:1,0} S$$
$$\neg I \wedge \neg Z \wedge T \wedge S \xrightarrow{1:1,0} S$$
$$\neg I \wedge \neg Z \xrightarrow{1:1,1} T$$
$$\neg Z \wedge \neg S \xrightarrow{1:1,1} T$$
$$\neg Z \wedge \neg T \xrightarrow{1:1,1} T$$
$$\neg Z \wedge \neg D \xrightarrow{1:1,1} T$$
$$\neg D \wedge \neg Z \wedge I \wedge S \wedge T \xrightarrow{1:1,1} T$$
$$I \wedge S \wedge T \wedge Z \wedge D \xrightarrow{1:1,0} T$$
$$Z \wedge \neg I \wedge S \xrightarrow{1:1,0} T$$
$$S \wedge T \wedge Z \xrightarrow{1:1,0} T$$
$$Z \wedge I \wedge S \wedge T \xrightarrow{1:1,0} T$$
$$I \wedge S \wedge T \wedge D \xrightarrow{1:1,0} T$$
$$Z \wedge \neg T \wedge S \xrightarrow{1:1,0} T$$
$$Z \wedge I \wedge T \xrightarrow{1:1,0} T$$
$$I \wedge T \wedge Z \wedge D \xrightarrow{1:1,0} T$$
$$I \wedge T \wedge S \xrightarrow{1:1,1} Z$$
$$\neg D \wedge \neg I \wedge S \wedge T \wedge Z \xrightarrow{1:1,1} Z$$
$$Z \wedge I \wedge T \xrightarrow{1:1,1} Z$$
$$Z \wedge \neg D \wedge S \xrightarrow{1:1,1} Z$$
$$I \wedge S \wedge Z \xrightarrow{1:1,1} Z$$
$$S \wedge T \wedge Z \wedge \neg D \xrightarrow{1:1,1} Z$$
$$Z \wedge \neg D \wedge I \xrightarrow{1:1,1} Z$$
$$I \wedge S \wedge T \wedge Z \wedge D \xrightarrow{1:1,1} Z$$
$$\neg I \wedge D \xrightarrow{1:1,0} Z$$
$$\neg T \wedge \neg Z \wedge D \xrightarrow{1:1,0} Z$$
$$\neg I \wedge \neg Z \wedge D \xrightarrow{1:1,0} Z$$
$$D \wedge \neg T \wedge \neg S \xrightarrow{1:1,0} Z$$
$$\neg I \wedge \neg T \wedge \neg S \xrightarrow{1:1,0} Z$$
$$\neg S \wedge \neg Z \wedge D \xrightarrow{1:1,0} Z$$
$$\neg I \wedge \neg S \xrightarrow{1:1,0} Z$$
$$\neg I \wedge D \wedge \neg T \xrightarrow{1:1,0} Z$$
$$\neg Z \wedge \neg T \wedge \neg S \xrightarrow{1:1,0} Z$$
$$\neg Z \wedge I \wedge T \xrightarrow{1:1,1} D$$
$$I \wedge T \wedge \neg D \xrightarrow{1:1,1} D$$
$$\neg D \wedge \neg S \wedge I \wedge T \wedge Z \xrightarrow{1:1,1} D$$
$$\neg D \wedge I \wedge T \wedge Z \xrightarrow{1:1,1} D$$
$$I \wedge S \wedge T \wedge Z \wedge D \xrightarrow{1:1,1} D$$
$$I \wedge S \wedge T \xrightarrow{1:1,1} D$$
$$S \wedge T \wedge Z \wedge I \xrightarrow{1:1,1} D$$
$$S \wedge T \wedge Z \wedge \neg D \wedge I \xrightarrow{1:1,1} D$$
$$Z \wedge \neg I \wedge \neg S \xrightarrow{1:1,0} D$$
$$\neg T \wedge Z \wedge \neg I \xrightarrow{1:1,0} D$$
$$\neg T \wedge Z \wedge D \xrightarrow{1:1,0} D$$
$$Z \wedge \neg T \xrightarrow{1:1,0} D$$
$$\neg T \wedge Z \wedge D \wedge S \xrightarrow{1:1,0} D$$
$$\neg I \wedge Z \wedge D \xrightarrow{1:1,0} D$$
$$\neg I \wedge Z \wedge D \wedge S \xrightarrow{1:1,0} D$$
$$\neg I \wedge S \xrightarrow{1:1,0} D$$
$$\neg T \wedge I \wedge S \wedge Z \wedge D \xrightarrow{1:1,0} D$$
$$\neg T \wedge S \xrightarrow{1:1,0} D$$
$$Z \wedge D \wedge \neg S \xrightarrow{1:1,0} D$$

Bibliografía

Karnaugh, M. (1953). The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, *72*(5), 593-599.

Tarjan, R. E. (1972). Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.*, *1*, 146-160.

Cheng, D. (2001). Semi-Tensor Product of Matrices and Its Application to Morgen ' s Problem, *44*(3), 18.

Kauffman, S., Peterson, C., Samuelsson, B. & Troein, C. (2003). Random Boolean Network Models and the Yeast Transcriptional Network. *Proceedings of the National Academy of Sciences*, *100*(25), 14796-14799. https://doi.org/10.1073/pnas.2036429100

Kauffman, S., Peterson, C., Samuelsson, B. & Troein, C. (2004). Genetic Networks with Canalyzing Boolean Rules Are Always Stable. *Proceedings of the National Academy of Sciences*, *101*(49), 17102-17107. https://doi.org/10.1073/pnas.0407783101

Cheng, D., Qi, H. & Li, Z. (2011). *Analysis and Control of Boolean Networks*. London, Springer London. https://doi.org/10.1007/978-0-85729-097-7

Layek, R. (2012). Pathways, Networks and Therapy: A Boolean Approach to Systems Biology.

Li, Y., Adeyeye, J. O., Murrugarra, D., Aguilar, B. & Laubenbacher, R. (2013). Boolean Nested Canalizing Functions: A Comprehensive Analysis. *Theoretical Computer Science*, *481*, 24-36. https://doi.org/10.1016/j.tcs.2013.02.020

Lu, M., Jolly, M. K., Levine, H., Onuchic, J. N. & Ben-Jacob, E. (2013). MicroRNA-based regulation of epithelial–hybrid–mesenchymal fate determination. *Proceedings of the National Academy of Sciences*, *110*(45), https://www.pnas.org/content/110/45/18144.full.pdf, 18144-18149. https://doi.org/10.1073/pnas.1318192110

Robeva, R., Kirkwood, B. & Davies, R. (2013). Mechanisms of Gene Regulation: Boolean Network Models of the Lactose Operon in Escherichia Coli, En *Mathematical Concepts and Methods in Modern Biology*. Elsevier. https://doi.org/10.1016/B978-0-12-415780-4.00001-6

Murrugarra, D. & Dimitrova, E. S. (2015). Molecular Network Control through Boolean Canalization. *J Bioinform Sys Biology*, *2015*(1), 9. https://doi.org/10.1186/s13637-015-0029-2

Hedegaard, J., Lamy, P., Nordentoft, I., Algaba, F., Høyer, S., Ulhøi, B. P., Vang, S., Reinert, T., Hermann, G. G., Mogensen, K., Thomsen, M. B. H., Nielsen, M. M., Marquez, M., Segersten, U., Aine, M., Höglund, M., Birkenkamp-Demtröder, K., Fristrup, N., Borre, M., ... Dyrskjøt, L. (2016). Comprehensive Transcriptional Analysis of Early-Stage Urothelial Carcinoma. *Cancer Cell*, *30*(1), 27-42. https://doi.org/10.1016/j.ccell.2016.05.004

Lerner, S. P., McConkey, D. J., Hoadley, K. A., Chan, K. S., Kim, W. Y., Radvanyi, F., Höglund, M. & Real, F. X. (2016). Bladder Cancer Molecular Taxonomy: Summary from a Consensus Meeting. *BLC*, *2*(1), 37-47. https://doi.org/10.3233/BLC-150037

Zhou, J. X., Samal, A., d'Hérouël, A. F., Price, N. D. & Huang, S. (2016). Relative Stability of Network States in Boolean Network Models of Gene Regulation in Development. *Biosystems*, *142-143*, 15-24. https://doi.org/10.1016/j.biosystems.2016.03.002

Choi, W., Ochoa, A., McConkey, D. J., Aine, M., Höglund, M., Kim, W. Y., Real, F. X., Kiltie, A. E., Milsom, I., Dyrskjøt, L. & Lerner, S. P. (2017). Genetic Alterations in the Molecular Subtypes of Bladder Cancer: Illustration in the Cancer Genome Atlas Dataset. *European Urology*, *72*(3), 354-365. https://doi.org/10.1016/j.eururo.2017.03.010

Yuan, R., Zhang, S., Yu, J., Huang, Y., Lu, D., Cheng, R., Huang, S., Ao, P., Zheng, S., Hood, L. & Zhu, X. (2017). Beyond Cancer Genes: Colorectal Cancer as Robust Intrinsic States Formed by Molecular Interactions. *Open Biol.*, *7*(11), 170169. https://doi.org/10.1098/rsob.170169

Akutsu, T. (2018). *Algorithms for Analysis, Inference, and Control of Boolean Networks*. WORLD SCIENTIFIC. https://doi.org/10.1142/10801

Joo, J. I., Zhou, J. X., Huang, S. & Cho, K.-H. (2018). Determining Relative Dynamic Stability of Cell States Using Boolean Network Model. *Sci Rep*, *8*(1), 12077. https://doi.org/10.1038/s41598-018-30544-0

Marzouka, N.-d., Eriksson, P., Rovira, C., Liedberg, F., Sjödahl, G. & Höglund, M. (2018). A Validation and Extended Description of the Lund Taxonomy for Urothelial Carcinoma Using the TCGA Cohort. *Sci Rep*, *8*(1), 3737. https://doi.org/10.1038/s41598-018-22126-x

Das, H., Deshpande, A. & Layek, R. K. (2019). A Linear Formulation of Asynchronous Boolean Networks. *IEEE Control Syst. Lett.*, *3*(2), 284-289. https://doi.org/10.1109/LCSYS.2018.2869045

Müller Bark, J., Kulasinghe, A., Chua, B., Day, B. W. & Punyadeera, C. (2020). Circulating Biomarkers in Patients with Glioblastoma. *Br J Cancer*, *122*(3), 295-305. https://doi.org/10.1038/s41416-019-0603-6