UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Puntos fijos de redes biológicas discretas

Denise Ayelén Galarza Rial

Directora: Alicia Dickenstein

17 de diciembre 2024

A Haydée

Agradecimientos

No sé por dónde arrancar a agradecer a todos los que me acompañaron y me acompañan, y que gracias a ellos esto fue posible.

A Alicia, por aceptarme para trabajar en este proyecto hermoso, por la paciencia y por estar siempre dispuesta a ayudar en lo que sea, incluso desde antes de ser mi directora.

A Juliana y Mercedes por incorporarme en su trabajo. Gracias, Mercedes, por darme feedback en la app y ayudarme a revisar cuentas :)

A los jurados, por ser mis jurados con todo lo que eso implica. También, Santiago por hacer que la aplicación se pueda usar en Windows y Eda, por sus correcciones y comentarios.

A mamá, papá y Nahue por estar a mi lado apoyándome en todas mis decisiones desde siempre.

A tíos, primos, abuelas y padrinos, que siempre están a mi lado.

Al CBC por darme la oportunidad de ser docente por más de diez años.

A mis amigos, a los que están desde la niñez, a los que conocí en la facu, a los que perdí contacto pero fueron muy importantes en mi vida y los que aún están a pesar de la distancia.

A mis suegros, Mica y Tomy, que siempre me reciben con los brazos abiertos.

Y por último a mi compañero de vida, Fran. Gracias por siempre empujarme para que pueda conseguir mis objetivos y acompañarme en todas las locuras. Todo es más fácil a tu lado. Te amo.

Gracias a todos por recordarme que todo se puede.

Índice general

Introducción

Las redes booleanas modelan la dinámica de un número finito de componentes que pueden tomar los posibles valores "sí" y "no", o 1 y 0. Pueden ser vistas como funciones sobre un conjunto de cadenas binarias de una longitud dada, descriptas por reglas lógicas. Se introdujeron como modelos dinámicos en biología, en particular como modelos lógicos de redes reguladoras intracelulares que involucran genes, proteínas y metabolitos y han sido usadas extensamente en ciencias de la computación, ingeniería y física. Fueron introducidas en la biología por Stuart Kauffman [16], como una clase modelo para redes reguladoras intracelulares, vistas como redes de conmutación lógica más que como redes de reacciones bioquímicas. Esto fue probado que es muy útil, en parte porque su especificación es intuitiva desde el punto de vista biológico y hay numerosas publicaciones de modelos de redes booleanas. Para modelos con un pequeño número de variables, es posible realizar una simulación exhaustiva del modelo, mientras que para modelos más grandes, un método común es el muestreo del espacio de estados del modelo. Una característica interesante a entender es cómo las dinámicas de una red booleana pueden tener alta complejidad, incluso estando compuestas de simples unidades elementales [15].

El científico belga René Thomas realizó contribuciones clave en genética, biología matemática y sistemas dinámicos. Hay mucho trabajo en redes booleanas atribuido a Thomas, quien argumentó que la intuición de los investigadores no era suficiente para entender las intrincadas redes regulatorias biológicas, y por ende propuso formalizar los modelos con las operaciones del álgebra booleana, donde las variables estaban prendidas o apagadas, y se usaban los operadores booleanos AND, OR y NOT [21]. Thomas también argumentó que el caso binario puede resultar demasiado simple para describir muchos sistemas biológicos, pues en muchos casos variables con múltiples acciones requieren diferentes niveles para producir efectos distintos [22]. Sin embargo, existen pocos estudios que aborden modelos multivaluados.

El análisis de estados estables (o puntos fijos) de redes booleanas es computacionalmente complejo, y aumentar el número de valores de las variables más allá de 2 aumenta en general sustancialmente esta complejidad. Sin embargo, en las aplicaciones de este marco de modelado en biología, a menudo resulta útil poder suponer que ciertas variables tienen más de dos valores posibles. Para la regulación génica, por ejemplo, un gen determinado puede tener varios modelos de acción, dependiendo de sus niveles de expresión. Para captar esta complejidad, una variable podría tener que adoptar varios valores diferentes, como "0 (desactivado), 1 (expresión media), 2 (expresión alta)". Las funciones que gobiernan dichas redes se pueden expresar en términos de lógica multivaluada, y las denominamos *redes multivaluadas*.

En [17] los autores estudian funciones multilineales representando funciones de

red $f : X_m^n \to X_m$ en redes multivaluadas pero el abordaje no es general. En [18] se estudia el abordaje de redes multivaluadas mediante diagramas de decisión lógicos, que usualmente requieren mucho espacio para expresarlos. La principal contribución de esta tesis, basada en el trabajo conjunto [14], es proporcionar un algoritmo para calcular los estados estables de una red multivaluada, usando operaciones de la lógica multivaluada [7, 9], que proporcionan una representación intuitiva de una red biológica. Más aún, usamos herramientas para computar puntos enteros en polítopos racionales, aprovechando el área de la combinatoria algebraica como fuente de algoritmos combinatorios para el análisis de las redes. Asimismo, proporcionamos una implementación del algoritmo para llevar a cabo dicho análisis de una manera computacionalmente eficiente.

Primero, hagamos el problema más preciso. Dadas n especies biológicas que interactúan, cuyos valores pueden describirse mediante $m+1$ estados, podemos establecer una biyección entre el conjunto de valores y el conjunto

$$X_m = \Big\{0, \frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}, 1\Big\}.$$

En particular, si $m = 1$, tenemos una red booleana. Nuestro objetivo es estudiar la dinámica de una función $f : X_m^n \to X_m^n$ con actualización sincrónica de las variables.

Cualquier función de n variables definida sobre un conjunto finito X de cardinal una potencia de un número primo (por ejemplo, en el caso booleano) puede ser pensado como una función sobre un cuerpo finito. Más aún, puede ser expresado como una función polinomial con coeficientes en un cuerpo, y esto abre el uso de herramientas de la geometría algebraica computacional como las bases de Gröbner [26]. La ventaja es que no es necesario expresar la función a través de una tabla completa.

Vamos a mostrar que, con las operaciones de la lógica multivaluada, que son expresadas en términos de desigualdades lineales, podemos recuperar la mayoría de las propiedades interesantes de las redes booleanas. Más aún, el cómputo de puntos fijos puede ser realizado algorítmicamente para cualquier m usando herramientas para encontrar puntos enteros en polítopos racionales y, en muchos casos, la complejidad para encontrar estos puntos fijos es esencialmente la misma que en el contexto booleano. Esto es una alternativa a traducir redes multivaluadas en redes booleanas donde aumenta el número de nodos y donde las funciones están definidas solo parcialmente (por ejemplo, ver [24]).

En la sección 2.1 introducimos los operadores lógicos $\oplus, \odot, \mathrm{neg}$ junto con sus propiedades principales y damos, en el Teorema 2.12, una forma constructiva de transformar datos de una tabla en una expresión de función en términos de $\odot, \mathrm{neg}$ y funciones constantes. Cuando modelamos redes biológicas es útil tener en mente el comportamiento de distintas pequeñas redes que aparecen frecuentemente como subredes de las más grandes, llamados *motifs*. También mostramos en esta sección el comportamiento de algunos pequeños motifs.

En la sección 2.2, mostramos cómo traducir algorítmicamente cualquier red en una red $\odot$-neg y mostramos en el Teorema 2.28 cómo recuperar sus puntos fijos. Incluso si trabajamos con actualizaciones sincrónicas de los nodos (es decir, si consideramos la dinámica dada por una función que describe simultáneamente las ac-

tualizaciones de todos los nodos), mostramos en el ejemplo 2.29 una pequeña red extraída del libro fundacional [23], donde el entorno multivaluado sincrónico puede modelar las características de un modelo asincrónico, ya que las redes se vuelven más *expresivas*. Notar que el orden en la cual las variables individualmente son actualizadas, ya sea sincrónico o asincrónicamente, no cambia los estados estables del sistema.

En la sección 3.2 proponemos muchas posibles reducciones del número de variables (inspiradas en [27] del contexto booleano) sin incrementar el indegree, es decir, sin incrementar la máxima cantidad de variables de las que depende cada variable (ver 3.5). Presentamos en la sección 3.2.1 un ejemplo, que es la traducción a nuestro contexto de una interesante red extraída de [35], donde el número de nodos eventualmente decrece. En los casos más interesantes, el cómputo es muy pesado para realizar manualmente y usamos nuestra implementación, disponible públicamente y descripta en este trabajo [11].

En la sección 3.3, nos concentramos en el cálculo de puntos fijos para redes $\odot$-neg. Como mencionamos, esto nos da los puntos fijos de cualquier red $F : X_m^n \to X_m^n$ por el Teorema 2.28. Un hecho crucial algorítmico para redes $\odot$-neg es el Teorema 3.6 que nos permite automatizar los cómputos sin simulaciones, que serían muy caros en general. Presentamos nuestro pseudo-código en 2 y discutimos el costo de su cómputo. Cuando las hipótesis del Teorema 3.11 se cumplen, hay solución única; cuando no se cumplen, es necesario contar la cantidad de puntos de coordenadas enteras en un polítopo racional, esto es, todas las soluciones enteras de un sistema de desigualdades dado por las formas lineales con coeficientes enteros definidos por una región acotada. Barvinok propuso en [3] un algoritmo que cuenta estos puntos en tiempo polinomial cuando la dimensión del polítopo está fija. La base teórica fue dada por Brion en el trabajo de [5], donde usa la relación de polítopos racionales con la K-teoría equivariante de las variedades tóricas asociadas.

Ejemplificamos el uso de nuestra aplicación para el cómputo de puntos fijos en nuestra red introducida en la sección 3.2.1.

En la sección 4.1 presentamos un modelo multivaluado para la red de denitrificación en *Pseudomonas aeruginosa* basado en tiempo discreto y el modelo multivaluado determinístico introducido en [1]. *P. aeruginosa* puede hacer una denitrificación completa, un proceso respiratorio que eventualmente produce N_2 atmosférico. Los parámetros externos en el modelo y algunas variables son tratadas como booleanas (en estado bajo o alto), y otras como ternarias (bajo, medio, o alto). Las reglas de actualización son formuladas en [1] en términos de MIN, MAX y NOT (que corresponden a AND, OR y NOT en el contexto booleano). De todas formas, las regulaciones de unas pocas variables que no se pueden expresar en términos de estos operadores (marcados con * en la columna de "Reglas de actualización" de su Tabla 3). Como enunciado en el Teorema 2.12, cualquier función puede ser escrita en términos de constantes y los operadores lógicos $\oplus, \odot,$ neg que proponemos usar en este trabajo, por lo que representamos sus reglas de actualización de esta manera, y encontramos los puntos fijos en este trabajo, como ya fue mostrado en [14].

Además de hacer una breve mención de las herramientas disponibles, hablando

de programas como LattE y Barvinok, y notaciones estándar de poliedros, también presentamos en este trabajo, en el Capítulo 5, una breve experimentación en el espacio de funciones. Sobre el final de este capítulo mostraremos en el Teorema 5.15 la cantidad exacta de funciones $\odot -$ neg para cada elección de m y n.

Al final de la tesis, incluimos en el Capítulo 6 conclusiones y algunas propuestas de trabajo futuro.

1 Motivación: Redes Booleanas

Motoyosi Sugita [20], físico japonés, introdujo en 1963 el análisis funcional de sistemas químicos in vivo usando circuitos lógicos, explorando el concepto de autómatas moleculares. De manera similar, el científico estadounidense Stuart Kauffman [16] utilizó redes genéticas aleatorias para estudiar la estabilidad metabólica y la epigénesis, contribuyendo significativamente a la biología teórica. René Thomas, en su artículo de 1973 [21] sobre formalización booleana de circuitos de control genético, impulsó el uso del álgebra booleana para modelar las complejidades de las redes biológicas, que representan de forma efectiva las dinámicas genéticas.

El álgebra booleana, que emplea variables binarias (0/APAGADO o 1/ENCENDIDO) y operadores lógicos simples como AND, OR y NOT, resultó especialmente adecuada para formalizar el proceso de razonamiento genético. Así, se podían formular fácilmente afirmaciones como "este gen estará activado sólo si el factor regulador (activador) está presente y el otro factor (inhibidor) está ausente". La modelización lógica de redes reguladoras, introducida por Thomas y sus colaboradores, tuvo un impacto significativo en el estudio de redes que controlan procesos biológicos variados, como la segmentación del embrión de Drosophila, el ciclo celular en eucariotas y la activación y diferenciación de linfocitos T. Investigaciones como las de Bardet, Faugère, Salvy y Spaenlehauer [2] han logrado complejidades en el caso booleano con n nodos de $O(2^{0.841n})$ bajo ciertas hipótesis, en contraste con la búsqueda exhaustiva que tiene una complejidad de $O(2^n \log(n))$.

En el caso de modelos booleanos sincrónicos, autores como Laubenbacher, Murrugarra y Veliz-Cuba (por ejemplo, en [26], [27], [28]) interpretaron funciones $f : \{0,1\}^n \to \{0,1\}^n$ en términos de polinomios en $F_2[x_1, \ldots, x_n]$, es decir con coeficientes en el cuerpo con dos elementos, y emplearon bases de Gröbner para calcular los puntos fijos de f mediante la solución del sistema polinomial cuadrado $f(x) - x = 0$. Para modelos con variables de más de dos valores, existen trabajos de Laubenbacher y colaboradores que usan polinomios sobre el campo F_{m+1}, cuando $m + 1$ es primo. Sin embargo, estas operaciones polinomiales carecen de una interpretación biológica natural y presentan detalles aritméticos innecesarios.

1.1. Operaciones básicas

En la lógica booleana tenemos los operadores de

- Conjunción: $x \wedge y = 1$ si $x = y = 1$, $x \wedge y = 0$ en caso contrario.
- Disyunción: $x \vee y = 0$ si $x = y = 0$, $x \vee y = 1$ en caso contrario.

- Negación: $\neg x = 0$ si $x = 1$, $\neg x = 1$ en caso contrario.

Estos operadores tendrán sus operadores correspondientes en nuestra propuesta de modelo de redes multivaluadas, y coincidirá con estas definiciones en el caso que estemos utilizando nuestro modelo multivaluado en una red booleana.

Proposición 1.1. *Las operaciones booleanas cumplen las siguientes propiedades:*

(i) Asociatividad de $\vee$:
$$x \vee (y \vee z) = (x \vee y) \vee z.$$

(ii) Asociatividad de $\wedge$:
$$x \wedge (y \wedge z) = (x \wedge y) \wedge z.$$

(iii) Conmutatividad de $\vee$:
$$x \vee y = y \vee x.$$

(iv) Conmutatividad de $\wedge$:
$$x \wedge y = y \wedge x.$$

(v) Distributividad de $\wedge$ sobre $\vee$:
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

(vi) Identidad para $\vee$:
$$x \vee 0 = x.$$

(vii) Identidad para $\wedge$:
$$x \wedge 1 = x.$$

(viii) Anulador para $\wedge$:
$$x \wedge 0 = 0.$$

Las siguientes leyes se cumplen en el álgebra booleana, pero no en el álgebra ordinaria:

(i) Anulador para $\vee$:
$$x \vee 1 = 1.$$

(ii) Idempotencia de $\vee$:
$$x \vee x = x.$$

(iii) Idempotencia de $\wedge$:
$$x \wedge x = x.$$

(iv) Absorción 1:
$$x \wedge (x \vee y) = x.$$

(v) Absorción 2:

$$x \vee (x \wedge y) = x.$$

(vi) Distributividad de $\vee$ sobre $\wedge$:

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z).$$

Muchas de estas propiedades se extenderán a nuestra propuesta de modelo de redes multivaluadas en el Capítulo 2, aunque no todas.

Otra propiedad que también podremos extender en nuestro caso multivaluado son las leyes de De Morgan: Si x e y son dos variables booleanas, valen las siguientes igualdades:
$\neg(x \vee y) = \neg x \wedge \neg y$,
$\neg(x \wedge y) = \neg x \vee \neg y$.
Las extensiones que haremos, coincidirán en el caso en el que estemos usando nuestro modelo multivaluado para una red booleana.

1.2. Las funciones AND-NOT

En el trabajo sobre redes booleanas de Veliz-Cuba [28] se transforman todas las funciones booleanas a las funciones AND-NOT de la Definición 1.3. El objetivo es poder representar de modo biyectivo la función booleana y el diagrama de conexiones (*wiring diagram*), que es un grafo dirigido naturalmente asociado a una función AND-NOT. Para esto, es necesario reemplazar las operaciones OR mediante el uso de las leyes de De Morgan y la eventual introducción de nuevas variables.

Definición 1.2. *Para un grafo dirigido $G = (V_G, E_G)$ con signos $+, -$, denotamos $I_i^+ = \{j : (j, i, s) \in E_G\}$, $I_i^- = \{j : (j, i, -s) \in E_G\}$. Es decir, I_i es el conjunto de todos los ejes incidentes al nodo i, I_i^+ es el subconjunto de ejes con signo positivo e I_i^- el subconjunto de ejes con signo negativo.*
Vamos a representar gráficamente:

- $i \longrightarrow j$ *cuando i es un activador de j (es decir, el eje es positivo para j),*
- $i \longdashv j$ *cuando i es un represor de j (es decir, el eje es negativo para j).*

Definición 1.3. *Una función AND-NOT es una función Booleana, $h : \{0,1\}^n \to \{0,1\}$, tal que h puede escribirse en la forma*

$$h(x_1, \ldots, x_n) = \bigwedge_{j \in P} x_j \wedge \bigwedge_{j \in N} \neg x_j, \tag{1.2.1}$$

donde $P \cap N = \emptyset$. Si $P = N = \emptyset$, entonces h es la función constante 1. Si $i \in P$ (o $i \in N$, respectivamente) decimos que i o x_i es un regulador positivo (negativo) de h o que es un activador (represor).

Una red AND-NOT es una red Booleana (BN), $f = (f_1, \ldots, f_n) : \{0,1\}^n \to \{0,1\}^n$, tal que f_i es una función AND-NOT para todos $i = 1, \ldots, n$. Las redes AND-NOT también se llaman redes conjuntivas con signos.

Definición 1.4. *El diagrama de conexiones (o wiring diagram) de una red AND-NOT está definido por un grafo $G = (V_G, E_G)$ con vértices $V_G = \{1, \dots, n\}$ (o $\{x_1, \dots, x_n\}$) y ejes E_G dados como sigue: $(i, j, +) \in E_G$ ($(i, j, -) \in E_G$, respectivamente) si x_i es un regulador positivo (negativo, respectivamente) de f_j.*

Nótese que los nodos correspondientes a funciones constantes tienen grado de entrada cero. Además, el diagrama de conexiones de una red AND-NOT contiene toda la información sobre la red; es decir, solo necesitamos especificar el diagrama de conexiones para definir una red AND-NOT.

Ejemplo 1.5. Consideremos la red Booleana $f = (f_1, \dots, f_6) : \{0,1\}^6 \to \{0,1\}^6$ dada por

$$f(x) = (x_2 \wedge x_4 \wedge \neg x_5, x_1 \wedge x_6 \wedge \neg x_3 \wedge \neg x_5, 1, x_6 \wedge \neg x_1 \wedge \neg x_5, x_6 \wedge \neg x_1, 1). \quad (1.2.2)$$

Es fácil ver que f es una red AND-NOT. Su diagrama de conexiones se muestra en la Figura 1.2.

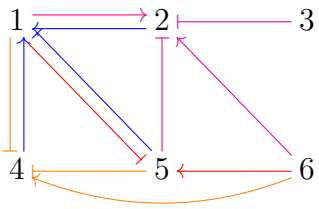


Figura 1.1: Diagrama de conexiones de la red AND-NOT en el Ejemplo 1.5.

Hay una buena cantidad de reducciones definidas en [27], con el objetivo de eliminar redundancia en redes booleanas en la búsqueda de puntos fijos. Por ejemplo, la primera reducción que propone [27] consiste en eliminar nodos sin aristas salientes, ya que dichos nodos no contribuyen a la cantidad de puntos fijos. En [27], podemos ver ejemplos de uso del modelado en redes booleanas. Trabajan sobre la diferenciación de Th-linfocitos, definiendo una función con 26 variables. Mencionan que el tamaño del espacio trabajado es exponencial, y muestran que con las reducciones propuestas logran llegar a una función con solo 4 estados, 3 de los cuales son estables.

En la sección 2.2 extendemos la definición de funciones AND-NOT a las funciones que llamamos $\odot$-neg.

1.3. Cuestiones que motivaron este trabajo

Si bien existe una extensa literatura sobre redes booleanas dinámicas (tanto sincrónicas como asincrónicas), los estudios sobre redes multivaluadas son escasos.

¿Cómo podemos representar funciones de manera que sean más cercanas a interpretaciones biológicas y que permitan cálculos sin recurrir a búsquedas exhaustivas? Además, buscamos condiciones que garanticen una complejidad computacional

comparable a la de las redes booleanas. Incluso para quienes buscan descripciones lógicas, es evidente que un enfoque puramente binario es, en muchos casos, insuficiente. ¿Cuándo es necesario emplear variables con más de dos valores? La respuesta surge al considerar que cuando una acción genera múltiples efectos, es probable que se necesiten diferentes niveles de activación para capturarlos. Por ejemplo, en un sistema donde un represor tiene efectos distintos como reprimir otros genes y activar su propia síntesis, puede requerirse una variable de tres niveles para representar adecuadamente estos umbrales de activación. En general, si una variable tiene n acciones distintas, se le pueden asignar n umbrales y tratarla como una variable de $n+1$ niveles, permitiendo una modelización más precisa y rica en el análisis de redes reguladoras biológicas.

2 Propuesta de modelo multivaluado

En este capítulo introducimos las definiciones necesarias para el resto del trabajo. En particular, las operaciones básicas de lógica multivaluada, la forma de usarlas para representar las funciones en modelos biológicos. Mostraremos cómo traducir algorítmicamente cualquier red en una red $\odot-$neg y mostraremos cómo recuperar sus puntos fijos. Cuando modelamos redes biológicas es útil tener en mente el comportamiento de distintas pequeñas redes que aparecen frecuentemente como subredes de las más grandes, llamados motifs. También mostraremos el comportamiento de algunos pequeños motifs.

2.1. Operaciones básicas

Fijado un número natural m, consideramos redes con $m+1$ valores en el conjunto

$$X_m = \left\{0, \frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}, 1\right\} \subset [0,1]. \tag{2.1.1}$$

Las redes Booleanas son un caso particular cuando $m = 1$. Como X_m es un subconjunto de $\mathbb{R}$, podemos considerar las operaciones estándar de suma/resta $(+/-)$ y multiplicación $(.)$ en la recta real. Sin embargo, introducimos las operaciones $\odot$, $\oplus$ que vienen de la lógica multivaluada [7, 9], la cual es más intuitiva y cercana a las interpretaciones biológicas.

2.1.1. Operaciones multivaluadas

Las 3 operaciones básicas en lógica multivaluada que usamos para construir todas las funciones que vamos a considerar son las siguientes:

Definición 2.1. *Consideramos las siguientes operaciones en X_m:*

$$\text{neg} : X_m \to X_m, \qquad \textit{donde } \text{neg}(x) = 1 - x, \tag{2.1.2}$$

$$\oplus : X_m \times X_m \to X_m, \qquad \textit{donde } x \oplus y = min\{1, x + y\}, \tag{2.1.3}$$

$$\odot : X_m \times X_m \to X_m, \qquad \textit{donde } x \odot y = max\{0, x + y - 1\}. \tag{2.1.4}$$

Ejemplo 2.2. Cuando $m = 2$, las operaciones (2.1.3) y (2.1.4) tienen la siguiente representación en la tabla:

$\oplus$	0	$\frac{1}{2}$	1
0	0	$\frac{1}{2}$	1
$\frac{1}{2}$	$\frac{1}{2}$	1	1
1	1	1	1

$\odot$	0	$\frac{1}{2}$	1
0	0	0	0
$\frac{1}{2}$	0	0	$\frac{1}{2}$
1	0	$\frac{1}{2}$	1

.

Ejemplo 2.3. Cuando $m = 1$, se cumple que $x \oplus y = x \wedge y, x \odot y = x \vee y$.

Observación 2.4. Las siguientes relaciones entre $\oplus$ y $\odot$, son formalmente iguales a las leyes de De Morgan entre AND y OR en el caso Booleano:

$$x \odot y = \text{neg}(\text{neg}(x) \oplus \text{neg}(y)), \tag{2.1.5}$$
$$x \oplus y = \text{neg}(\text{neg}(x) \odot \text{neg}(y)). \tag{2.1.6}$$

La siguiente proposición lista otras propiedades útiles, cuya demostración es directa

Demostración de (2.1.5). Notar que, para todo $x, y \in X_m$

$$\begin{aligned} x \odot y = \text{neg}(\text{neg}(x) \oplus \text{neg}(y)) \Leftrightarrow \\ \text{neg}(x \odot y) = \text{neg}(x) \oplus \text{neg}(y) \Leftrightarrow \\ 1 - \text{máx}\{0, x + y - 1\} = \text{mín}\{1, 2 - (x + y)\} \end{aligned} \tag{2.1.7}$$

Consideramos dos casos:

- Si $x + y < 1$, entonces en (2.1.7), tenemos $1 - 0 = 1$.
- Si $x + y \geq 1$, entonces en (2.1.7), tenemos $1 - (x + y - 1) = 2 - (x + y)$.

□

Demostración de (2.1.6). Notar que, para todo $x, y \in X_m$

$$\begin{aligned} x \oplus y = \text{neg}(\text{neg}(x) \odot \text{neg}(y)) \Leftrightarrow \\ \text{neg}(x \oplus y) = \text{neg}(x) \odot \text{neg}(y) \Leftrightarrow \\ 1 - \text{mín}\{1, x + y\} = \text{máx}\{0, 1 - (x + y)\} \end{aligned} \tag{2.1.8}$$

Consideramos dos casos:

- Si $x + y < 1$, entonces en (2.1.8), tenemos $1 - (x + y) = 1 - (x + y)$.
- Si $x + y \geq 1$, entonces en (2.1.8), tenemos $1 - 1 = 0$.

□

Proposición 2.5. *Dado m y X_m como en (2.1.1), considerando las operaciones definidas en la Definición 2.1. Se siguen las siguientes propiedades:*

(i) $\text{neg}(0) = 1$ *y* $\text{neg}(1) = 0$.

(ii) El producto de varias variables puede ser expresado como

$$x_1 \odot \cdots \odot x_n = \text{máx}\{0, x_1 + \cdots + x_n - (n - 1)\}.$$

(iii) *Analogamente,*
$$x_1 \oplus \cdots \oplus x_n = \min\{1, x_1 + \cdots + x_n\}.$$

(iv) *El producto* $\odot$ *es asociativo y conmutativo.*

(v) $0 \odot x = 0$ *y* $1 \odot x = x$ *para todo* $x \in X_m$.

(vi) *Para todo conjunto finito de índices* I, $\operatorname{neg}(\bigoplus_{i\in I} a_i) = \bigodot_{i\in I} \operatorname{neg}(a_i)$.

(vii) $\max\{x, y\} = (x \odot \operatorname{neg}(y)) \oplus y$.

(viii) $\min\{x, y\} = (x \oplus \operatorname{neg}(y)) \odot y$.

(ix) $x \leq y$ *si y sólo si* $x \odot \operatorname{neg}(y) = 0$.

(x) $x \odot y = 0$ *si y sólo si* $x + y \leq 1$. *Más en general,* $x_1 \odot \cdots \odot x_n = 0$ *si y sólo si* $x_1 + \cdots + x_n \leq (n-1)$.

Demostración.

(i) $\operatorname{neg}(0) = 1 - 0 = 1$,
$\operatorname{neg}(1) = 1 - 1 = 0$.

(ii) and (iii) Se demuestran por inducción.

(iv) De ii) es inmediato que valen la asociatividad y conmutatividad.

(v) Por definición $0 \odot x = \max\{0, x-1\} = 0$ dado que $0 \leq x \leq 1$.
Por definición $1 \odot x = \max\{0, 1+x-1\} = max\{0, x\} = x$ dado que $0 \leq x \leq 1$.

(vi) Con $n \in I$, si $|I| \geq 2$

$$\operatorname{neg}(\bigoplus_{i\in I} a_i) = \bigodot_{i\in I} \operatorname{neg}(a_i)$$
$$\operatorname{neg}\left(a_n \oplus \bigoplus_{i\in I, i\neq n} a_i\right) = a_n \odot \bigodot_{i\in I, i\neq n} \operatorname{neg}(a_i)$$

Y esto vale por (2.1.6) y iv).

(vii) Para todo $x, y \in [0, 1]$, queremos ver que

$$\begin{aligned} \max\{x, y\} &= (x \odot \operatorname{neg}(y)) \oplus y \\ &= \min\{1, \max\{0, x - y\} + y\} \end{aligned}$$

Consideramos dos casos:

- Si $x \geq y$, entonces la ecuación de arriba es equivalente a $x = \text{mín}\{1, x\}$, con $x \in [0, 1]$, que siempre es cierto.
- Si $x < y$, entonces la ecuación es equivalente a $y = \text{mín}\{1, y\}$, con $y \in [0, 1]$, que siempre es cierto.

(viii)

$$\begin{aligned}\text{mín}\{x, y\} &= (x \oplus \text{neg}(y)) \odot y \\ &= \text{máx}\{0, \text{mín}\{1, 1 + x - y\} + y - 1\}\end{aligned}$$

Consideramos dos casos:

- Si $x \geq y$, entonces tenemos $y = \text{máx}\{0, 1 + y - 1\}$, con $y \in [0, 1]$
- Si $x < y$, entonces tenemos $x = \text{máx}\{0, 1 + x - y + y - 1\}$, con $x \in [0, 1]$

(ix)

$$\begin{aligned}x \leq y \iff \\ x - y \leq 0 \iff \\ \text{máx}\{0, x - y\} = 0 \iff \\ x \odot \text{neg}(y) = 0.\end{aligned}$$

(x)

$$\begin{aligned}x \odot y = 0 \iff \\ \text{máx}\{0, x + y - 1\} = 0 \iff \\ x + y - 1 \leq 0 \iff \\ x + y \leq 1.\end{aligned}$$

$$\begin{aligned}x_1 \odot \cdots \odot x_n = 0 \iff \\ \text{máx}\{0, x_1 + \cdots + x_n - (n-1)\} = 0 \iff \\ x_1 + \cdots + x_n - (n-1)\} \leq 0 \iff \\ x_1 + \cdots + x_n \leq (n-1).\end{aligned}$$

□

Observación 2.6. La asociatividad nos permite obviar escribir paréntesis en algunos casos. Es decir, escribimos:

$$\begin{aligned}s_1(x_1) \odot s_2(x_2) \odot s_3(x_3) &= s_1(x_1) \odot \Big(s_2(x_2) \odot s_3(x_3)\Big) \\ &= \Big(s_1(x_1) \odot s_2(x_2)\Big) \odot s_3(x_3)\end{aligned}$$

donde $s_i(x_i) = x_i$ o $s_i(x_i) = \text{neg}(x_i)$.

Observación 2.7. Las operaciones que definimos no satisfacen distributividad. En general, $(x \oplus y) \odot z \neq (x \odot z) \oplus (y \odot z)$. Por ejemplo, si $m = 5, x = \frac{3}{5}, y = \frac{2}{5}, z = \frac{1}{5}$,

$$(x \oplus y) \odot z = \frac{1}{5}$$
$$(x \odot z) \oplus (y \odot z) = 0.$$

Definición 2.8. *También definimos la resta, exponenciación y multiplicación por un entero $k \in \mathbb{N}$ como:*

$$x \ominus y := x \odot \operatorname{neg}(y) \tag{2.1.9}$$
$$= \operatorname{máx}\{0, x - y\} \tag{2.1.10}$$
$$= \begin{cases} x - y & si\ x - y \geq 0 \\ 0 & en\ caso\ contrario \end{cases} \tag{2.1.11}$$
$$x^k := \underbrace{x \odot \cdots \odot x}_{k\ times} = \operatorname{máx}\{0, kx - (k-1)\} \tag{2.1.12}$$
$$kx := \underbrace{x \oplus \cdots \oplus x}_{k\ times} = \operatorname{mín}\{1, kx\}. \tag{2.1.13}$$

Observación 2.9. Notar que $x \odot y \leq x$ para todo $x, y \in X_m$, y se cumple la igualdad solo cuando $y = 1$ o $x = 0$.

En particular, $x^k \leq x$ para todo $x \in X_m$ y todo $k \in \mathbb{N}$, con igualdad solo cuando $x = 0, 1$ o $k = 1$.

Observación 2.10. Es interesante notar que $\operatorname{neg}(kx) = \operatorname{neg}(x)^k$, en efecto:

$$\operatorname{neg}(kx) = \operatorname{neg}(\underbrace{x \oplus \cdots \oplus x}_{k\ \text{times}}) = \operatorname{neg}(\operatorname{neg}(\underbrace{\operatorname{neg}(x) \odot \cdots \odot \operatorname{neg}(x)}_{k\ \text{times}})) = \operatorname{neg}(x)^k.$$

2.1.2. Representabilidad de las funciones

Mostramos como representar la función $f : X_m^n \to X_m$, en términos de los operadores $\odot$ y neg de una manera constructiva. Empecemos con un ejemplo.

Ejemplo 2.11. Sea $m = 2$, tal que $X_m = X_2 = \{0, \frac{1}{2}, 1\}$. Definimos la siguiente red

$$f_0(x) = \operatorname{neg}(x)^2$$
$$f_1(x) = x^2,$$
$$f_{\frac{1}{2}}(x) = \operatorname{neg}(f_0(x)) \odot \operatorname{neg}(f_1(x)).$$

Entonces,

x	$f_0(x)$	$f_{\frac{1}{2}}(x)$	$f_1(x)$
0	1	0	0
$\frac{1}{2}$	0	1	0
1	0	0	1

Esto significa que $f_0, f_{\frac{1}{2}}, f_1$ son interpoladores y esto nos permite escribir cualquier $f : X_2 \to X_2$ en términos de $\oplus, \odot, \operatorname{neg}$ y constantes. Más aun, deducimos de

las identidades en la observación 2.4 que podemos traducir cualquier operador $\oplus$ en operadores que sólo involucren operaciones $\odot$ y neg.

En el siguiente resultado, daremos una prueba basada en interpolación de funciones como en el ejemplo 2.11.

Teorema 2.12. *Dado* $X_m = \{0, \frac{1}{m}, \ldots, \frac{m-1}{m}, 1\}$ *y* $n \in \mathbb{N}$. *Toda función* $f : X_m^n \longrightarrow X_m$ *con variables* $x_1, \ldots, x_n$, *se puede expresar en términos de* $\odot$, neg *y funciones constantes.*

Demostración. Toda función $f : X_m^n \to X_m$ puede ser representada como

$$f(x_1, \ldots, x_n) = \bigoplus_{(i_1, i_2, \ldots, i_n) \in X_m^n} f(i_1, i_2, \ldots, i_n) \odot \ell_{i_1}(x_1) \odot \cdots \odot \ell_{i_n}(x_n),$$

donde las funciones $\ell_{\frac{j}{m}}$ son definidas de la siguiente manera.

Sea $\ell_0(x) = (\mathrm{neg}(x))^m$ y $\ell_1(x) = x^m$.

Para $1 \leq j \leq m-1$, definimos

$$g_j(x) := (x \oplus \frac{m-j}{m})^m = \begin{cases} 0 & \text{si } x < \frac{j}{m} \\ 1 & \text{si } x \geq \frac{j}{m} \end{cases}$$

$$h_j(x) := (\mathrm{neg}(x) \oplus \frac{j}{m})^m = \begin{cases} 0 & \text{si } x > \frac{j}{m} \\ 1 & \text{si } x \leq \frac{j}{m} \end{cases}$$

$$\ell_{\frac{j}{m}}(x) = g_j(x) \odot h_j(x) = \begin{cases} 0 & \text{si } x \neq \frac{j}{m} \\ 1 & \text{si } x = \frac{j}{m} \end{cases}$$

De esto se deduce que

$$\ell_{i_1}(x_1) \odot \cdots \odot \ell_{i_n}(x_n) = \begin{cases} 0 & \text{si } (x_1, \ldots, x_n) \neq (i_1, \ldots, i_n) \\ 1 & \text{si } (x_1, \ldots, x_n) = (i_1, \ldots, i_n) \end{cases}$$

Ahora simplemente iteramos aplicando las igualdades de la observación 2.4 para deshacernos del operador $\oplus$. □

De hecho, podemos expresar cualquier función en términos de $\oplus$, neg y funciones constantes.

Por ejemplo, los interpoladores del ejemplo 2.11 son

$$f_0(x) = \mathrm{neg}(x \oplus x)$$
$$f_1(x) = \mathrm{neg}(\mathrm{neg}(x) \oplus \mathrm{neg}(x))$$
$$f_{\frac{1}{2}}(x) = \mathrm{neg}(f_0(x) \oplus f_1(x)).$$

Pero en general vamos a usar la versión sin $\oplus$ para imitar la elección estándar en los casos booleanos. Por otro lado,

Es importante notar que si no permitimos funciones constantes en el teorema 2.12, el resultado no es cierto. Por ejemplo, con $m = 2$, la función constante $\frac{1}{2}$ no puede ser expresada en términos de $\oplus, \odot$ y neg.

Observación 2.13. Diferentes funciones sobre $\mathbb{R}$ pueden coincidir en X_m y puede existir más de una expresión $\odot$-neg para la misma función.

(i) Por ejemplo,

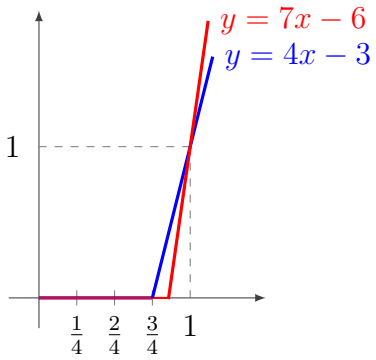
- Con $s \in \mathbb{N}$,

$$x^s = \text{máx}\{0, sx - (s-1)\}$$

Entonces para todo $s \geq m$ si $x \in X_m$ vale que

$$x^s = x^m = \begin{cases} 1 & \text{si } x = 1 \\ 0 & \text{en caso contrario} \end{cases} \tag{2.1.14}$$

Pero son distintas en $\mathbb{R}$: si $x \in \left[\frac{s-1}{s}, +\infty\right) - \{1\}$, $x^s \neq x^t \ \forall s, t : m < s < t$ (ver Figura 2.1)



x	0	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{3}{4}$	1
x^4	0	0	0	0	1
x^7	0	0	0	0	1

Figura 2.1: Comparando x^4 y x^7 cuando $m = 4$. Como función con valores reales, $max\{0, 7x-6\}$ y $max\{0, 4x-3\}$ son diferentes pero coinciden en $X_m = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\}$.

- Análogamente, para todo $s \geq m$ si $x \in X_m$ vale que

$$\text{neg}(x)^s = \begin{cases} 1 & \text{si } x = 0, \\ 0 & \text{en caso contrario.} \end{cases}$$

Demostración. Para demostrar (2.1.14), comenzamos observando que:

$$x^s = \text{máx}\{0, sx - (s-1)\}.$$

En particular si $1 + s(x-1) > 0$, entonces:

$$\frac{-1}{s} < x - 1,$$

y así,

$$x > 1 - \frac{1}{s}.$$

Por lo tanto,

$$x > \frac{s-1}{s}. \tag{2.1.15}$$

Ahora analizamos los casos:

- Si $x = 1$, entonces $1^s = 1$.
- Si $x < 1$, entonces $x \leq \frac{m-1}{m}$. Reemplazando en la expresión (2.1.15), obtenemos:

  $$\frac{s-1}{s} < \frac{m-1}{m}.$$

  De aquí se sigue que

  $$\frac{1}{m} < \frac{1}{s},$$

  lo cual implica que $m > s$.

  Por lo tanto si $s \geq m$ no se cumple que $1 + s(x-1) > 0$ por lo que $x^s = \text{máx}\{0, sx - (s-1)\} = 0$ si $x \leq 1$.

□

(ii) Para $m = 3$, la igualdad $x^2 \odot \frac{1}{3} = x \odot \frac{1}{3}$ vale para todo $x \in X$.

$$x^2 \odot \frac{1}{3} = \begin{cases} \frac{1}{3} & \text{si } x = 1 \\ 0 & \text{en caso contrario} \end{cases}$$

Más en general, dado $m \in \mathbb{N}$, para cada $2 \leq p \leq m$ la igualdad $x \odot \frac{1}{m} = x^p \odot \frac{1}{m} = \frac{1}{m}$ vale para todo $x \in X$.

Este tipo de funciones nos da una forma más intuitiva de ver lo que sucede biológicamente.

Ejemplo 2.14. En la figura 2.2, que presenta un ejemplo de dos nodos que interactúan entre si, con tres valores cada uno, donde la segunda variable actúa como represora de la primera. La función de actualización del primer nodo, f_1, se representa tanto como un polinomio sobre el cuerpo con 3 elementos, lo que no da ninguna pista sobre su comportamiento, como en términos de las operaciones lógicas, cuya interpretación es transparente.

$x_1 \setminus x_2$	0	1	2
0	0	0	0
1	1	0	0
2	2	1	0

(a) $f_1 : \mathbb{F}_3^2 \to \mathbb{F}_3$
$f_1(x_1, x_2) = x_1 + 2x_1x_2 + x_1^2x_2 + x_1x_2^2 + x_1^2x_2^2$

$x_1 \setminus x_2$	0	$\frac{1}{2}$	1
0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0
1	1	$\frac{1}{2}$	0

(b) $f_1 : \{0, 1/2, 1\}^2 \to \{0, 1/2, 1\}$
$f_1(x_1, x_2) = x_1 \ominus x_2$.

Figura 2.2: (a) Los nodos toman valores $\{0, 1, 2\}$ y el polinomio $f_1 \in \mathbb{F}_3[x_1, x_2]$ resume los valores de la tabla.
(b) Los nodos toman valores $\{0, 1/2, 1\}$ y la tabla es la traducción de la primera; la función $f_1 = x_1 \ominus x_2$ tiene en cuenta la represión mostrada en la tabla.

Vamos a describir cómo podemos construir una función *suavizada* sin alterar sus puntos fijos. Los puntos fijos van a ser relevantes en nuestro trabajo, y empezaremos a entrar más en detalle sobre ellos en la sección 3.1.

Siguiendo las ideas en [6], introducimos una función para *preservar continuidad*, lo que significa que cada nodo va a cambiar a lo sumo $\frac{1}{m}$ en cada paso temporal. Para esto vamos a tener en cuenta el estado previo del nodo correspondiente, y vamos a agregar un ciclo de autorregulación a cada nodo de la red. El valor futuro de la variable regulada bajo continuidad se computa como sigue:

Definición 2.15. *Dado $X_m = \{0, \frac{1}{m}, \dots, 1\}$ definimos la función $h : X_m^2 \to X_m$ como*

$$h(x, y) = \begin{cases} x \oplus \frac{1}{m} & si\ y > x \\ x & si\ y = x \\ x \ominus \frac{1}{m} & si\ y < x \end{cases} \tag{2.1.16}$$

Dada $F = (f_1, \dots, f_n) : X_m^n \to X_m^n$, la versión continua de cada función coordenada $f_i : X_m^n \to X_m$ es escrita como $f_{i,cont}$ y es definida como

$$f_{i,cont}(x) = h(x_i, f_i(x)), \tag{2.1.17}$$

para $i = 1, \dots, n$. Denotamos $F_{cont} = (f_{1,cont}, \dots, f_{n,cont})$.

El siguiente lema es una consecuencia inmediata de la definición previa:

Lema 2.16. *Una función $F : X_m^n \to X_m^n$ y su función asociada F_{cont} en la definición 2.15 tienen los mismos puntos fijos.*

Para todo m, la versión continua de todas las potencias x^k es la misma función, independientemente de k.

Lema 2.17. *Dado $k \in \mathbb{N}_{\geq 2}$, entonces*

$$x_{cont}^k = \begin{cases} x \ominus \frac{1}{m} & si\ x < 1 \\ 1 & si\ x = 1 \end{cases}$$

Demostración. Es directo chequear que $x \geq x^k$ para todo $x \in X$ y que la igualdad se alcanza solo si $x = 0$ o $x = 1$. □

2.1.3. Motifs

Representamos en esta sección cuatro simples *motifs*. Estas son pequeñas redes que aparecen con frecuencia como subredes de otras mas grandes. Son construidas usando operaciones de lógica multivaluada para redes con $n \geq 3$ interacciones de especies biológicas las cuales pueden ser descriptas con $m + 1$ valores en $X_m = \{0, \frac{1}{m}, \ldots, \frac{m-1}{m}, 1\}$.

Motif 1: Son mecanismos, donde el nodo 1 es moderadamente activado por el nodo 2. Mostramos ejemplos de funciones resultantes que describen esta situación.

1. Una opción es considerar.

   $$f_1 = x_2^2 = \text{máx}\{0, 2x_2 - 1\}$$

   Por ejemplo, si $m = 5$, tenemos

x_2	f_1
0	0
1/5	0
2/5	0
3/5	1/5
4/5	3/5
1	1

2. Ahora presentamos la versión suavizada del ejemplo anterior, que describe como el nodo 1 es moderadamente activado por el nodo 2:

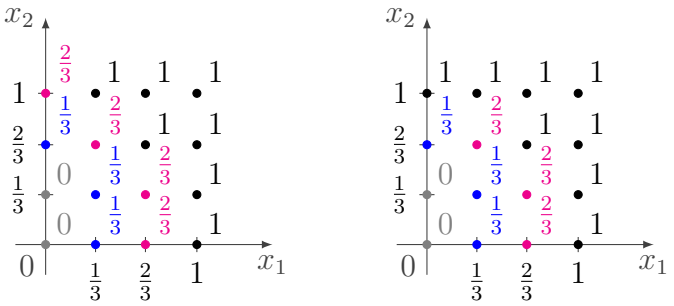
   $$f_1 = (x_2^2)_{cont}$$

   Por ejemplo, si $m = 5$, podemos comparar con el ejemplo anterior y ver en ambos casos como a medida que incrementa el valor del nodo 2 el valor del nodo 1 incrementa. En el caso continuo el incremento es de manera más suave.

x_2	x_2^2	f_1
0	0	0
1/5	0	0
2/5	0	1/5
3/5	1/5	2/5
4/5	3/5	3/5
1	1	1

3. Otra alternativa es cuando el nodo 1 es moderadamente activado por el nodo 2 pero a la vez tiene en cuenta el valor de x_1. Dos ejemplos de esto para $m = 3$

Motif 1a: $f_1 = x_1 \oplus (x_2 \ominus \frac{1}{m})$ Motif 1b: $f_1 = x_1 \oplus x_2^2$

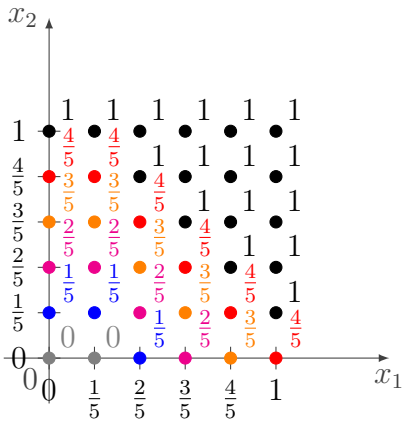


Motif 2: Nodo 2 activa al nodo 1, y el nodo 1 se degrada.

$$f_1 = x_2 \oplus \left(\frac{m-1}{m} \odot x_1\right) = x_2 \oplus \left(x_1 \ominus \frac{1}{m}\right)$$

Cuando $m = 5$, tenemos

$f_1 = x_2 \oplus (x_1 \ominus \frac{1}{5})$



Motif 3: En este motif el nodo 1 es dependiente de su estado previo, el nodo 2 actúa como activador y el nodo 3 actúa como activador débil.

$$\begin{aligned} f_1 &= x_1 \oplus x_2 \oplus x_3^2 \\ &= \text{mín}\{1, x_1 + x_2 + \text{máx}\{0, 2x_3 - 1\}\}. \\ &= \text{neg}(\text{neg}(x_1) \odot \text{neg}(x_2) \odot \text{neg}(x_3 \odot x_3)). \end{aligned}$$

Motif 4: En este motif la función de actualización del nodo 1 es dependiente de sí mismo, un nodo activador 2 y un nodo represor 3.

$$\begin{aligned} f_1 &= x_2 \oplus (x_1 \ominus x_3) \\ &= x_2 \oplus (x_1 \odot \text{neg}(x_3)) \\ &= \text{mín}\{1, x_2 + \text{máx}\{0, x_1 - x_3\}\}. \end{aligned}$$

Asumiendo $m = 3$, entonces:

x_2	x_3	x_1	f_1
0	0	1/3	1/3
1/3	0	1/3	2/3
2/3	0	1/3	1
1	0	1/3	1
0	0	2/3	2/3
1/3	0	2/3	1
2/3	0	2/3	1
1	0	2/3	1
0	1/3	2/3	1/3
1/3	1/3	2/3	2/3
2/3	1/3	2/3	1
1	1/3	2/3	1

x_2	x_3	x_1	f_1
0	0	1	1
1/3	0	1	1
2/3	0	1	1
1	0	1	1
0	1/3	1	2/3
1/3	1/3	1	1
2/3	1/3	1	1
1	1/3	1	1
0	2/3	1	1/3
1/3	2/3	1	2/3
2/3	2/3	1	1
1	2/3	1	1

Motif 5: Finalmente introducimos un mecanismo que considera el resultado de muchos activadores e inhibidores actuando simultáneamente en un nodo. Proponemos el siguiente motif que describe el efecto total sobre el nodo ℓ producido por los activadores $i \in A$ y los inhibidores $j \in B$, con $A, B \subseteq \{1, \dots, n\}$, $A \cap B = \emptyset$: $f_\ell = \oplus_{i\in A} x_i \ominus (\oplus_{j\in B} x_j)$, o, más generalmente, dados pesos positivos $w_k \in \mathbb{N}$:

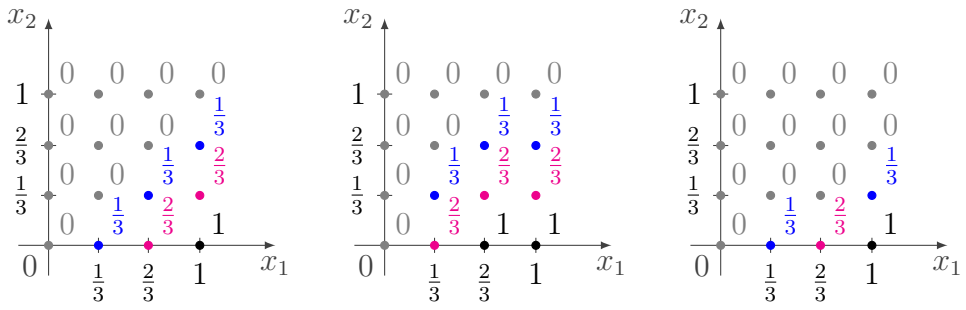
$$f_\ell = \bigoplus_{i\in A} w_i x_i \ominus \left(\bigoplus_{j\in B} w_j x_j\right) = \text{máx}\{0, \text{mín}\{1, \sum_{i\in A} w_i x_i\} - \text{mín}\{1, \sum_{j\in B} w_j x_j\}\}.$$

Observación 2.18. Si el efecto de los inhibidores es suficientemente grande, es decir $\sum_{j\in B} w_j x_j \geq 1$, $f_\ell = 0$

Observación 2.19. Si $\sum_{i\in A} w_i x_i \leq 1$ y $\sum_{j\in B} w_j x_j \leq 1$, $f_\ell > 0$ si y sólo si $\sum_{i\in A} w_i x_i > \sum_{j\in B} w_j x_j$

Esto imita las funciones de cotas booleanas (boolean threshold functions, o funciones indicadoras de $X \geq c$). Por ejemplo, mostramos 3 de estas funciones para $m = 3$:

$f_\ell = x_1 \ominus x_2$ $\quad$ $f_\ell = 2x_1 \ominus x_2$ $\quad$ $f_\ell = x_1 \ominus 2x_2$



2.2. Traducción a funciones $\odot$-neg

En las siguientes secciones nos vamos a enfocar en sistemas dinámicos sobre el conjunto X_m. Consideremos $n \in \mathbb{N}$ y una función $F : X_m^n \to X_m^n$, $F = (f_1, \dots, f_n)$

que describe la actualización sincrónica de un sistema biológico con n nodos con valores en el conjunto finito X_m como en (2.1.1).

Inspirados por la estructura en [28], proponemos nuestro propio método para computar los puntos fijos de la red. Las ecuaciones se pueden simplificar previamente usando las consideraciones de la sección 3.2.

Vamos a imitar el procedimiento que es útil en el contexto de redes booleanas y que mencionamos en el Capítulo 1. Recordemos que toda función booleana f se puede escribir en forma conjuntiva normal como

$$f = w_1 \wedge \cdots \wedge w_r$$

donde

$$w_j = s_1 x_{i_1} \vee \cdots \vee s_{n_j} x_{i_{n_j}}$$

for $j \in \{1, \ldots, r\}$, $\{i_1, \ldots, i_{n_j}\} \subseteq \{1, \ldots, n\}$ y $s_k \in \{id, \text{neg}\}$. Esto nos da la idea de una función AND-NOT: una función vectorial $F = (f_1, \ldots, f_n) : X_m^n \to X_m^n$ tal que para todo $f_i : X_m \to X_m$ es una función AND-NOT, tiene la misma información que su diagrama de conexiones (wiring diagram) [28, 27].

Para obtener propiedades similares en nuestro contexto definimos lo que llamamos funciones ⊙-neg.

Definición 2.20. *Una función $f : X_m^n \longrightarrow X_m$ es llamada una función* ⊙-neg *si se puede escribir como*

$$f(x_1, \ldots, x_n) = \bigodot_{j \in A} x_j^{p_j} \odot \bigodot_{j \in B} \text{neg}(x_j)^{q_j} \bigodot c$$

con $c \in X_m$, $A \cup B \subseteq \{1, \ldots, n\}$, $A \cap B = \emptyset$ y $\{p_j, q_j\} \subseteq \mathbb{N}$.

Una función vectorial $F = (f_1, \ldots, f_n) : X_m^n \longrightarrow X_m^n$ es llamada una función de red ⊙-neg *si f_i es una función* ⊙-neg *para todo $i \in \{1, \ldots, n\}$.*

La característica más importante de las redes ⊙-neg es que hay una correspondencia directa entre su diagrama de conexiones y las funciones de red. Toda la información sobre la dinámica de la red se puede leer del diagrama de conexiones y viceversa, lo que es definido por el grafo etiquetado $G = (V_G, E_G)$ con vértices

$$V_G = \{1, \ldots, n, C\}$$

($\{1, \ldots, n\}$ también se puede denotar $\{x_1, \ldots, x_n\}$) y aristas E_G como sigue. Si $F = (f_1, \ldots, f_n)$ y

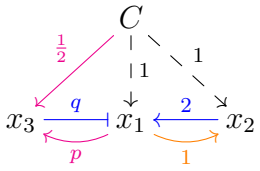
$$f_l(x_1, \ldots, x_n) = \bigodot_{i \in A_l} x_i^{p_{l_i}} \odot \bigodot_{j \in B_l} \text{neg}(x_j)^{q_{l_j}} \bigodot c_l,$$

entonces existen aristas etiquetadas de la forma

$$i \xrightarrow{p_{l_i}} l \text{ si } p_{l_i} > 0; \quad j \xrightarrow{q_{l_j}} l \text{ si } q_{l_j} > 0; \quad C \xrightarrow{c_l} l \,. \tag{2.2.1}$$

No toda función es una función ⊙-neg. Un ejemplo de esto es la función $f_1 = x_i \oplus x_j$ para cualquier i, j. Pero podemos traducirla a una función ⊙-neg usando la igualdad (2.1.6). Daremos la traducción general en el Teorema 2.28.

Ejemplo 2.21. Dada $F(x_1, x_2, x_3) = (x_2^2 \odot \text{neg}(x_3)^q, x_1, \frac{1}{2} \odot x_1^p)$, el diagrama de conexiones asociado es como sigue:



(Dado que multiplicar por 1 es no hacer nada, las flechas punteadas se pueden saltear.)

Proposición 2.22. *Dada $f \not\equiv 0$ una función* $\odot$-neg *en $f : X_m^n \to X_m$ escrita como en la definición 2.20, los conjuntos A, B se pueden elegir de manera tal que $A \cap B = \emptyset$, $c \neq 0$ y $1 \leq p_j, q_j \leq m$.*

Demostración. Primero notemos que si $A \cap B \neq \emptyset$ entonces $f = 0$ porque $x \odot neg(x) = 0$ para todo x por (2.1.9). La última parte de esta afirmación sigue de la observación 2.13. □

Antes de demostrar que cualquier función de red $F : X_m^n \to X_m^n$ se puede transformar en una función de red $\odot$-neg donde podemos rastrear los puntos fijos de F, introducimos una medida de complejidad que va a estimar el tamaño de la nueva red $\odot$-neg.

Definición 2.23. *Definimos por el algoritmo 1 una medida de complejidad de la función $f : X_m^n \to X_m$ llamada* profundidad, *denotada por $\mu(f)$. También definimos la función asociada $\overline{f}$ dependiendo de $\mu(f)$ nuevas variables.*

Algorithm 1 Como construir μ y $\overline{f}$

Input: $f : X_m^n \to X_m$.
Output: $\mu(f)$ una función red $(\overline{f}, \overline{f}_{u_1}, \ldots, \overline{f}_{u_{\mu(f)}}) : X_m^{n+\mu(f)} \to X_m^{1+\mu(f)}$.

```
μ(f) ← 0
f̄_0 ← f
k ← 1
while hay una expresión neg(h), con h una función ⊙-neg con dos o más
factores, en f̄_{k−1} do
    agregar una nueva variable u_k
    f̄_k ← la función resultante de reemplazar h en f̄_{k−1} con la nueva variable u_k
    f̄_{u_k} ← h
    μ(f) ← μ(f) + 1
    k ← k + 1
end while
f̄ ← f̄_k
```

Dada una función red $F = (f_1, \ldots, f_n) : X_m^n \to X_m^n$, aplicamos el algoritmo a todo f_i, y definimos $\mu(F) = \sum_{i=1}^n \mu(f_i)$.

$\mu(F)$ da una cota superior al total de nuevas variables que se necesitan agregar para poder escribir a cada F_i como una función ⊙-neg como en la definición 2.20.

Observación 2.24. El algoritmo 1 puede ser mejorado, por ejemplo, haciendo una lista de sustituciones. Se agrega una nueva variable solo si la nueva sustitución no está en la lista.

Observación 2.25. Para $f : X_m^n \to X_m$ y $1 \leq k \leq \mu(f)$, notemos que $\overline{f}_{u_k}$ solo depende de variables previamente definidas: $\overline{f}_{u_1} = \overline{f}_{u_1}(x_1, \ldots, x_n)$, y $\overline{f}_{u_k} = \overline{f}_{u_k}(x_1, \ldots, x_n, u_1, \ldots, u_{k-1})$, para $2 \leq k$.

Ejemplo 2.26. Si recordamos los motifs 3, 4 y 5 de 2.1.3, podemos transformar cada función en una función ⊙-neg reproduciendo los pasos del algoritmo 1.

Motif 3:

$$\begin{aligned} f_1 &= x_1 \oplus x_2 \oplus x_3^2 \\ &= \text{mín}\{1, x_1 + x_2 + \text{máx}\{0, 2x_3 - 1\}\} \\ &= \text{neg}(\text{neg}(x_1) \odot \text{neg}(x_2) \odot \text{neg}(x_3 \odot x_3)) \end{aligned} \quad \Rightarrow \quad \begin{aligned} \overline{f}_1 &= \text{neg}(u_2) \\ \overline{f}_{u_1} &= x_3 \odot x_3 \\ \overline{f}_{u_2} &= \text{neg}(x_1) \odot \text{neg}(x_2) \odot \text{neg}(u_1) \end{aligned}$$

Motif 4:

$$\begin{aligned} f_1 &= x_2 \oplus (x_1 \ominus x_3) \\ &= \text{mín}\{1, x_2 + \text{máx}\{0, x_1 - x_3\}\} \\ &= \text{neg}(\text{neg}(x_2) \odot \text{neg}(x_1 \odot \text{neg}(x_3))) \end{aligned} \quad \Rightarrow \quad \begin{aligned} \overline{f}_1 &= \text{neg}(u_2) \\ \overline{f}_{u_1} &= x_1 \odot \text{neg}(x_3) \\ \overline{f}_{u_2} &= \text{neg}(x_2) \odot \text{neg}(u_1) \end{aligned}$$

Motif 5:

$$\begin{aligned} f_\ell &= \textstyle\bigoplus_{i \in A} w_i x_i \ominus \left(\bigoplus_{j \in B} w_j x_j \right) \\ &= \text{máx}\{0, \text{mín}\{1, \sum_{i \in A} w_i x_i\} - \text{mín}\{1, \sum_{j \in B} w_j x_j\}\} \\ &= \text{neg}(\textstyle\bigodot_{i \in A} \text{neg}(x_i)^{w_i}) \odot (\bigodot_{j \in B} \text{neg}(x_j)^{w_j}) \end{aligned} \quad \Rightarrow \quad \begin{aligned} \overline{f}_\ell &= \text{neg}(u_1) \odot u_2 \\ \overline{f}_{u_1} &= \textstyle\bigodot_{i \in A} \text{neg}(x_i)^{w_i} \\ \overline{f}_{u_2} &= \textstyle\bigodot_{j \in B} \text{neg}(x_j)^{w_j} \end{aligned}$$

En los 3 casos el valor correspondiente de μ es 2 y podemos ver que esto está relacionado a la cantidad de cambios entre máximos y mínimos que ocurre en las funciones.

Observación 2.27. Notemos que por los ítems (ii) y (iii) en la proposición 2.5, operaciones $\oplus$ consecutivas (respectivamente $\odot$) pueden sen reemplazadas por un solo mínimo (respectivamente máximo). Entonces, dada cualquier función $f : X_m^n \to X_m$, $\mu(f)$ *mide* la cantidad de cambios entre operaciones $\oplus$ y $\odot$ en la expresión de f.

Podríamos haber escrito cualquier función en términos de $\oplus$, neg y funciones constantes, y propuesto un algoritmo similar a agregar una variable nueva para cada negación de una suma. Elegimos funciones ⊙-neg en vez ⊕-neg para generalizar [28].

Los diagramas de conexión de redes ⊙-neg codifican toda la información de la red. Los puntos fijos del sistema dinámico obtenido por la iteración de F están en biyección con los puntos fijos de $\overline{F}$.

Dada $F : X_m^{n_1} \to X_m^{n_1}$ y su correspondiente $\overline{F} : X_m^{n_2} \to X_m^{n_2}$ construida acorde a la definición 2.23, llamamos $x = (x_1, \ldots, x_{n_1})$, y definimos la función $U : X_m^{n_1} \to X_m^{n_2 - n_1}$ de manera recursiva (ver la observación 2.25):

$$\begin{cases} U_1(x) = \overline{f}_{u_1}(x), \\ U_k(x) = \overline{f}_{u_k}(x, U_1(x), \ldots, U_{k-1}(x)), \text{ para } 2 \leq k \leq n_2 - n_1. \end{cases} \tag{2.2.2}$$

Teorema 2.28. *Dada $F : X_m^{n_1} \to X_m^{n_1}$, el algoritmo de la definición 2.23 construye $\overline{F} : X_m^{n_2} \to X_m^{n_2}$ que consiste de funciones $\odot$-neg y la función $U : X_m^{n_1} \to X_m^{n_2-n_1}$ como en 2.2.2, donde $n_2 - n_1 \leq \mu(F)$. Más aún, el siguiente diagrama conmuta:*

$$\begin{array}{ccc} X_m^{n_1} & \xrightarrow{(Id,U)} & X_m^{n_2} \\ {\scriptstyle F}\downarrow & & \downarrow{\scriptstyle \overline{F}} \\ X_m^{n_1} & \xleftarrow[\pi(x,y)=x]{} & X_m^{n_2} \end{array} \tag{2.2.3}$$

y los puntos fijos de F están en biyección con los puntos fijos de $\overline{F}$ a través de la proyección en la primeras n_1 coordenadas.

Previo a la demostración exponemos un ejemplo:

Ejemplo 2.29. Este ejemplo está basado en Thomas y D'ari [23, Capítulo 4,§II]. Consideremos 3 genes $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ con sus respectivos productos denotados x, y, z. El gen $\mathcal{X}$ está expresado constitutivamente (no regulado), el gen $\mathcal{Y}$ está expresado solo en la ausencia del producto x, y el gen $\mathcal{Z}$ está expresado según estén el producto y o z presentes.

Las relaciones lógicas booleanas propuestas en [23] son

$$(x, y, z) \mapsto (1, \neg x, y \vee z),$$

con 2 estados estables: $(1, 0, 0)$ y $(1, 0, 1)$.

Los autores argumentan que empezar del estado inicial $(0, 0, 0)$ si el gen $\mathcal{X}$ está prendido (por ejemplo inmediatamente después de una infección por un virus) entonces el gen $\mathcal{Y}$ va a ser expresado hasta que el producto x aparezca, entonces apagándolo. El gen $\mathcal{Z}$ se va a prender solo si y aparece antes de que x apague al gen Y. Este modelo es asincrónico con 3 retrasos temporales (time delay) t_x, t_y, t_z. Si actualizamos primero a x antes que a y (es decir $t_x < t_y$), entonces $(0, 0, 0) \mapsto (1, 0, 0)$ y el sistema se mantiene en ese estado como el caso sincrónico booleano. Cuando $t_x > t_y$ entonces $(0, 0, 0) \mapsto (0, 1, 0)$. Si además $t_x > t_y + t_z$, entonces $(0, 1, 0) \mapsto (1, 1, 1) \mapsto (1, 0, 1)$ y el sistema permanece en $(1, 0, 1)$.

Proponemos, en cambio, el siguiente modelado *sincrónico* con $m = 3$. Suponemos que $\mathcal{X}$ apaga $\mathcal{Y}$ completamente solo si x está a nivel 1 (completamente activado). Consideramos entonces la siguiente función de red $F = (f_x, f_y, f_z)$:

$$F : X_m^3 \to X_m^3, \quad F(x, y, z) = \left(x \oplus \frac{1}{3}, \mathrm{neg}(x), y \oplus z\right).$$

La órbita de $(0, 0, 0)$ sobre F es: $(0, 0, 0) \mapsto (\frac{1}{3}, 1, 0) \mapsto (\frac{2}{3}, \frac{2}{3}, 1) \mapsto (1, \frac{1}{3}, 1) \mapsto (1, 0, 1) \mapsto (1, 0, 1)$. Entonces, $(1, 0, 1)$ es un punto fijo y, mientras $\mathcal{X}$ es activado en 3 pasos, $\mathcal{Z}$ se puede prender incluso si $\mathcal{Y}$ es eventualmente apagado. Notar que $(1, 0, z_0)$ es un punto fijo para cualquier valor de z_0 en z.

Para traducir F a una función $\odot -$ neg $\overline{F}$ necesitamos agregar 2 variables u_1, u_2. Entonces, tenemos que $\overline{F} = (\overline{f}_x, \overline{f}_y, \overline{f}_z, \overline{f}_{u_1}, \overline{f}_{u_2}) : X_m^5 \to X_m^5$ está definido por

$$\begin{aligned}
\overline{f}_x &= \mathrm{neg}(u_1)\\
\overline{f}_y &= \mathrm{neg}(x)\\
\overline{f}_z &= \mathrm{neg}(u_2)\\
\overline{f}_{u_1} &= \mathrm{neg}(x) \odot \tfrac{2}{3}\\
\overline{f}_{u_2} &= \mathrm{neg}(y) \odot \mathrm{neg}(z).
\end{aligned}$$

Sus puntos fijos están en biyección con los de F proyectando en las primeras 3 coordenadas. En efecto, si $(x^*, y^*, z^*, u_1^*, u_2^*)$ es un punto fijo de $\overline{F}$ entonces

$$\begin{aligned}
x^* &= \mathrm{neg}(u_1^*) = \mathrm{neg}(\mathrm{neg}(x^*) \odot \frac{2}{3}) = x^* \oplus \frac{1}{3} = f_x(x^*, y^*, z^*),\\
y^* &= \mathrm{neg}(x^*) = f_y(x^*, y^*, z^*),\\
z^* &= \mathrm{neg}(u_2^*) = \mathrm{neg}(\mathrm{neg}(y^*) \odot \mathrm{neg}(z^*)) = y^* \oplus z^* = f_z(x^*, y^*, z^*),
\end{aligned}$$

y (x^*, y^*, z^*) es punto fijo de F.

Recíprocamente, si (x^*, y^*, z^*) es un punto fijo de F, definimos $u_1^* = \mathrm{neg}(x^*) \odot \frac{2}{3}$ y $u_2^* = \mathrm{neg}(y^*) \odot \mathrm{neg}(z^*)$ y es directo chequear que $(x^*, y^*, z^*, u_1^*, u_2^*)$ es un punto fijo de $\overline{F}$.

Demostración del teorema 2.28. Del algoritmo 1 y la definición 2.23 es directo chequear que $n_2 - n_1 \leq \mu(F)$.

También es inmediato de la definición de $\overline{F}$ y U que $\pi(\overline{F}(x, U(x))) = F(x)$ para cualquier $x \in X^{n_1}$ y que el diagrama (2.2.3) conmuta.

Consideremos un punto fijo x^* de F y definamos $u^* = U(x^*)$. Entonces, $\overline{f}_i(x^*, u^*) = \overline{f}_i(x^*, U(x^*)) = f_i(x^*) = x_i^*$ para $1 \leq i \leq n_1$, y $\overline{f}_i(x^*, u^*) = u_i^*$ para $1 \leq i \leq n_2 - n_1$ por la definición de u^*. Luego, (x^*, u^*) es punto fijo de $\overline{F}$.

Análogamente, si (x^*, u^*) es un punto fijo de $\overline{F}$, por la definición recursiva 2.2.2 de U tenemos que $u^* = U(x^*)$. Entonces, $F(x^*) = \pi(\overline{F}(x^*, U(x^*))) = \pi(\overline{F}(x^*, u^*)) = \pi(x^*, u^*) = x^*$.

Concluimos entonces que x^* es un punto fijo de F. □

3 Puntos fijos de redes multivaluadas

Proponemos en esta sección algunas reducciones que pueden ser aplicadas a una red multivaluada $F : X_m^n \to X_m^n$, y su correspondiente red $\odot$-neg $\overline{F}$, que eventualmente producen una red $\odot$-neg $G : X_m^{n'} \to X_m^{n'}$ cuyos puntos fijos nos permiten reconstruir inmediatamente los puntos fijos de F. Usualmente $n' << n$, por ejemplo 3.2.1 ($n = 19$, $n' = 4$) o 4.1 ($n = 15$, $n' = 5$ en el peor caso).

Primero simplificamos las ecuaciones con la ayuda de algunas reducciones que pueden ser fácilmente vistas desde sus expresiones algebraicas.

Después transformamos la red reducida en una red $\odot$-neg, en la que nuevas variables son agregadas para ganar simplicidad al trabajo con los máximos y mínimos impuestos por las estructuras lógicas.

Después aplicamos más reducciones de red a la red $\odot$-neg antes de lidiar con las (usualmente pocas) ecuaciones de punto fijo.

3.1. Puntos fijos

Al estudiar las dinámicas de sistemas biológicos, un aspecto importante a considerar es el efecto a largo plazo del comportamiento del sistema, especialmente el equilibro del sistema dinámico.

En el caso de sistemas biológicos de tiempo continuo, es común estudiar los *estados estables* del sistema. En el contexto de tiempo discreto, los estados estables (o atractores) son llamados *puntos fijos* del sistema.

Más aún, si hay una cantidad finita de estados para cada nodo y el sistema dinámico está dado por $F : X_m^n \to X_m^n$, los puntos fijos son los los puntos $x \in X_m^n$ tal que $F(x) = x$ (i.e. $f_i(x) = x_i$ para todo $i \in \{1, \ldots, n\}$).

Estos puntos van a ser nuestro foco principal en las siguientes secciones.

Es sensible preguntarse cuáles son los puntos fijos de la función $F : X_m^n \to X_m^n$ con X_m un conjunto finito, dado que la mayoría de estas funciones tienen al menos uno. Al contar la cantidad de estas funciones sin puntos fijos surge el siguiente resultado:

Proposición 3.1. *Sea $q = (m+1)^n$, donde $\#X_m = m + 1$. La proporción de $F : X_m^n \to X_m^n$ con al menos un punto fijo equivale a*

$$\frac{q^q - (q-1)^q}{q^q} = 1 - \left(\frac{q-1}{q}\right)^q$$

Entonces, cuando $n \to +\infty$ o $m \to +\infty$ esta proporción está cerca de $1 - \frac{1}{e} \sim \frac{2}{3}$

Demostración. Como $\#X_m = m + 1$ entonces $\#X_m^n = q$ Entonces $\#\{F : X_m^n \to X_m^n\} = q^q$ pues para cada elemento del dominio tengo q opciones para la imagen y son q elementos en el dominio.

Ahora contemos cuantas funciones $F : X_m^n \to X_m^n$ no tienen ningún punto fijo. Para cada elemento del dominio tengo $q - 1$ opciones para la imagen, pues no puede valer lo mismo que si mismo ya que sino seria un punto fijo. Como hay q elementos en el dominio, entonces el cardinal del conjunto de funciones $\{F : X_m^n \to X_m^n$ sin puntos fijos$\}$ es igual a $(q-1)^q$. Por lo tanto,

$$\begin{aligned} &\#\{\text{funciones con al menos un punto fijo}\} \\ &= \#\{\text{funciones totales}\} - \#\{\text{funciones sin punto fijo}\} \\ &= q^q - (q-1)^q. \end{aligned}$$

□

3.2. Reducciones de red

Muchas reducciones de red son propuestas en [27] para redes booleanas AND-NOT. En nuestro contexto multivaluado algunas de estas reducciones aún son válidas.

Proveemos aquí una lista de reducciones con el objetivo de reducir la complejidad de la red. Cuando usamos estas reducciones la red cambia pero los puntos fijos se mantienen.

(i) Si $f_i = c$ con $c \in X_m$, entonces x_i puede ser *separado*, es decir ignorado respecto a las ecuaciones de punto fijo dado que no cambia el resultado.

(ii) Si $f_i = x_k$ o $f_i = \text{neg}(x_k)$ entonces x_i puede ser *separado* y podemos reemplazar $x_i = x_k$ o $x_i = \text{neg}(x_k)$ respectivamente en cada función.

(iii) Si $f_i = f_j$, entonces podemos reemplazar $f_i = x_j$ y *separamos* x_i.

(iv) Si x_i no aparece en ningún f_j, entonces x_i puede ser *separado*.

(v) Para todo x tenemos la identidad $x \odot \text{neg}(x) = 0$ por (2.1.9).

Más precisamente, si $f_i = x_k \odot \text{neg}(x_k) \odot w$, donde w es una función multivaluada, entonces podemos reemplazar $x_i = 0$, y x_i puede ser *separado*.

(vi) Si $f_i = x_j \odot x_k \odot w$, y $f_j = \text{neg}(x_k) \odot w'$, entonces, en un punto fijo, $f_i = x_k \odot \text{neg}(x_k) \odot w'' = 0$, donde w, w' y w'' son funciones multivaluadas.

Entonces, en un punto fijo, $x_i = 0$ y x_i puede ser *separado*.

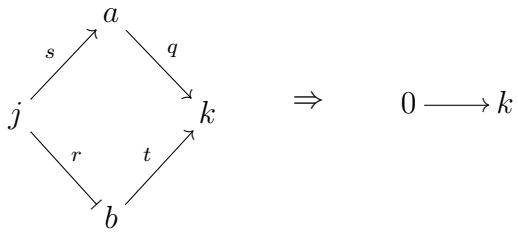
(vii) Por la observación 2.13, podemos reemplazar cualquier potencia $s > m$ por m.

Decimos que *separamos* una variable, cuando reemplazamos x_i por su correspondiente expresión de f_i en toda función f_j donde aparece x_i. Por ejemplo, si $f_i = x_k$, $i \neq k$, reemplazamos $x_i = x_k$ en toda f_j para $j \neq i$. En consecuencia, tanto x_i como f_i no están más involucrados en el nuevo sistema de ecuaciones a ser considerado. Guardamos afuera del sistema la ecuación $x_i = f_i$ para obtener el valor de x_i como punto fijo después de obtener los valores de las otras variables involucradas en f_i (ver el ejemplo 3.2.1).

Recordemos que hay una correspondencia directa entre el diagrama de conexiones de una red $\odot$-neg y la función red. Presentamos en la siguiente proposición una nueva reducción que puede ser fácilmente detectada en el diagrama de conexiones de dicha red, generalizando la observación (v).

Observación 3.2. Sea $F : X_m^n \to X_m^n$ una red $\odot$-neg, y sea x un punto fijo.

Consideremos el siguiente diagrama de conexiones de F y miremos una reducción posible:

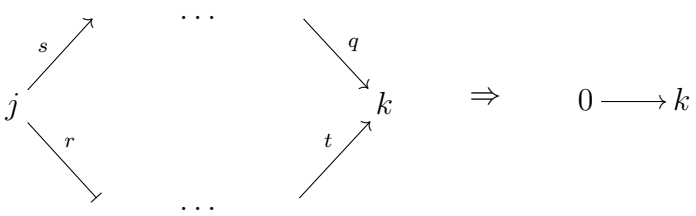


Como x es un punto fijo, tenemos que $f_k = x_a^q \odot x_b^t = x_k$, $f_a = x_j^s = x_a$ y $f_b = \text{neg}(x_j)^r = x_b$. Entonces, por el ítem (ii), $x_k = (x_j^s)^q \odot (\text{neg}(x_j)^r)^t$. Notemos que esto es de la forma $x_j \odot \text{neg}(x_j) \odot \omega$. Usando la reducción (v), entonces, $x_k = 0$.

Ahora veamos una generalización de dicha reducción:

Observación 3.3. Sea $F : X_m^n \to X_m^n$ una red $\odot$-neg, y sea x un punto fijo.

Consideremos el siguiente diagrama de conexiones de F y miremos una reducción posible:



Esto es, dos de los caminos que llegan a x_k

- Vienen de un mismo nodo x_j
- Uno de los caminos consiste de todos los nodos como activadores (es decir, se multiplica (con la operación $\odot$) por potencias de las variables).
- El otro camino consiste de x_j actuando como $\text{neg}(x_j)^r$ en la primera flecha, y los nodos subsiguientes como activadores.

Entonces, reemplazando los valores de las distintas coordenadas en un punto fijo x, tenemos que $x_k = f_k = x_j \odot neg(x_j) \odot \omega = 0$.

Sea $F = (f_1, \ldots, f_n) : X_m^n \to X_m^n$ una función red $\odot$-neg representada por un digrafo, con $n+1$ nodos $\{x_1, \ldots, x_n, C\}$ como en (2.2.1).

Definición 3.4. *Sea $F = (f_1, \ldots, f_n) : X_m^n \to X_m^n$ una función red $\odot$-neg representada por su digrafo asociado. Definimos $indeg(f_i) = \#\{$flechas que llegan a $x_i\}$ y $indeg(F) = \max\{indeg(f_i)\}$.*

Entonces el grado de entrada mide el máximo número de variables de la que cada función coordenadaf_i de F depende. En el caso Booleano, la mayoría de las reducciones descriptas en [27] no generan un incremento del grado de entrada. Aplicando cualquier secuencia de las reducciones que introducimos antes en esta sección, podemos transformar el cómputo de los puntos fijos de una función red $\odot$-neg F en el cómputo de los puntos fijos de otra red reducida, con menor cantidad de variables, que denotamos F_{red}.

Proposición 3.5. *Cualquier secuencia de reducciones descriptas para una función red $\odot$-neg F a una función red F_{red} no incrementa el grado de entrada.*

La prueba de la proposición 3.5 es inmediata.

3.2.1. Ciclo celular de los mamíferos

Vamos a analizar el ejemplo de Cell Collective que está descripto en [34].

Generalicemos la red Booleana descripta. Esta red modela la red de interacción de la tirosina quinasa transmembrana ERBB2. La sobreexpresión de esta proteína es un marcador pronóstico adverso en el cáncer de mama y se presenta en casi el 30 % de las pacientes. Usando las reducciones, pudimos reducir de las 19 variables originales a 4 variables y una constante. Aun así, en uno de los casos, el cómputo es muy largo para hacer a mano y usamos nuestra implementación [11], que nos da una respuesta inmediata en este caso.

Definimos el siguiente conjunto de variables:

$c = \text{EGF}$,	$x_7 = \text{cMYC}$,	$x_{14} = \text{p27}$,
$x_1 = \text{ErbB1}$,	$x_8 = \text{Akt1}$,	$x_{15} = \text{pRB}$,
$x_2 = \text{ERa}$,	$x_9 = \text{ErbB2}$,	$x_{16} = \text{ErbB2\_3}$,
$x_3 = \text{CycE1}$,	$x_{10} = \text{p21}$,	$x_{17} = \text{CDK6}$,
$x_4 = \text{CycD1}$,	$x_{11} = \text{ErbB1\_2}$,	$x_{18} = \text{CDK2}$,
$x_5 = \text{CDK4}$,	$x_{12} = \text{ErbB1\_3}$,	$x_{19} = \text{MEK1}$.
$x_6 = \text{ErbB3}$,	$x_{13} = \text{IGF1R}$,	

Traducimos la red booleana a nuestra configuración cambiando $\vee \rightsquigarrow \oplus$, $\wedge \rightsquigarrow \odot$ y $\neg \rightsquigarrow$ neg para todo valor de m:

$f_1 = \mathrm{c}$
$f_2 = x_8 \oplus x_{19}$
$f_3 = x_7$
$f_4 = (x_{19} \odot x_2 \odot x_7) \oplus (x_8 \odot x_2 \odot x_7)$
$f_5 = x_4 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$f_6 = c$
$f_7 = x_{19} \oplus x_2 \oplus x_8$
$f_8 = x_{12} \oplus x_{13} \oplus x_1 \oplus x_{11} \oplus x_{16}$
$f_9 = \mathrm{c}$
$f_{10} = x_2 \odot \mathrm{neg}(x_8) \odot \mathrm{neg}(x_7) \odot \mathrm{neg}(x_5)$
$f_{11} = x_1 \odot x_9$
$f_{12} = x_1 \odot x_6$
$f_{13} = (x_8 \odot \mathrm{neg}(x_{16})) \oplus (x_2 \odot \mathrm{neg}(x_{16}))$
$f_{14} = x_2 \odot \mathrm{neg}(x_{18}) \odot \mathrm{neg}(x_5) \odot \mathrm{neg}(x_7) \odot \mathrm{neg}(x_8)$
$f_{15} = (x_{18} \odot x_5 \odot x_{17}) \oplus (x_5 \odot x_{17})$
$f_{16} = x_9 \odot x_6$
$f_{17} = x_4$
$f_{18} = x_3 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$f_{19} = x_1 \oplus x_{13} \oplus x_{12} \oplus x_{16} \oplus x_{11}$

Siguiendo las reducciones de 3.2, empezamos reemplazando $x_1 = x_6 = x_9 = c$, $x_3 = x_7$, $x_{17} = x_4$ y eliminando f_1, f_3, f_6, f_9 y f_{17}.

Como x_{15} no está involucrada en ninguna ecuación, definimos $x_{15} = (x_{18} \odot x_5 \odot x_{17}) \oplus (x_5 \odot x_{17})$ y eliminamos f_{15}. Luego reemplazamos $x_{11} = x_{12} = x_{16} = c^2$, $x_{19} = x_8$ y eliminamos f_{11}, f_{12}, f_{16} y f_{19}. Más aún, como $x_{10} = x_2 \odot \mathrm{neg}(x_8) \odot \mathrm{neg}(x_7) \odot \mathrm{neg}(x_5)$ podemos reemplazar esta expresión en f_{14}. El resultado de aplicar estas reducciones es:

$f_2 = x_8 \oplus x_{19}$
$f_4 = (x_{19} \odot x_2 \odot x_7) \oplus (x_8 \odot x_2 \odot x_7)$
$f_5 = x_4 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$f_7 = x_{19} \oplus x_2 \oplus x_8$
$f_8 = x_{12} \oplus x_{13} \oplus c \oplus x_{11} \oplus x_{16}$
$f_{10} = x_2 \odot \mathrm{neg}(x_8) \odot \mathrm{neg}(x_7) \odot \mathrm{neg}(x_5)$
$f_{11} = c^2$
$f_{12} = c^2$
$f_{13} = (x_8 \odot \mathrm{neg}(x_{16})) \oplus (x_2 \odot \mathrm{neg}(x_{16}))$
$f_{14} = x_{10} \odot \mathrm{neg}(x_{18})$
$f_{16} = c^2$
$f_{18} = x_7 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$f_{19} = c \oplus x_{13} \oplus x_{12} \oplus x_{16} \oplus x_{11}$

$\rightsquigarrow$

$f_2 = x_8 \oplus x_8$
$f_4 = (x_2 \odot x_7 \odot x_8) \oplus (x_2 \odot x_7 \odot x_8)$
$f_5 = x_4 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$f_7 = x_2 \oplus x_8 \oplus x_8$
$f_8 = x_{13} \oplus c \oplus c^2 \oplus c^2 \oplus c^2$
$f_{10} = x_2 \odot \mathrm{neg}(x_8) \odot \mathrm{neg}(x_7) \odot \mathrm{neg}(x_5)$
$f_{13} = (x_8 \odot \mathrm{neg}(c^2)) \oplus (x_2 \odot \mathrm{neg}(c^2))$
$f_{14} = x_{10} \odot \mathrm{neg}(x_{18})$
$f_{18} = x_7 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$

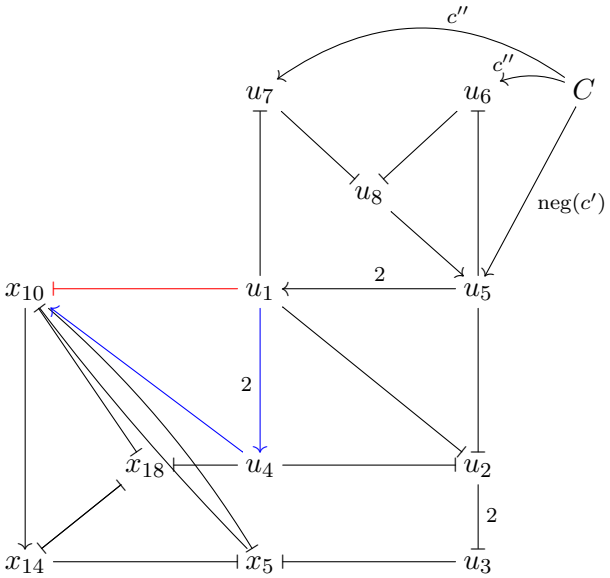
Tenemos la siguiente función red $\odot$-neg, donde llamamos $c' = c \oplus c^2 \oplus c^2 \oplus c^2$, $c'' = \mathrm{neg}(c^2)$:

$\overline{f}_2 = \mathrm{neg}(u_1)$
$\overline{f}_4 = \mathrm{neg}(u_3)$
$\overline{f}_5 = x_4 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$\overline{f}_7 = \mathrm{neg}(u_4)$
$\overline{f}_8 = \mathrm{neg}(u_5)$
$\overline{f}_{10} = x_2 \odot \mathrm{neg}(x_8) \odot \mathrm{neg}(x_7) \odot \mathrm{neg}(x_5)$
$\overline{f}_{13} = \mathrm{neg}(u_8)$
$\overline{f}_{14} = x_{10} \odot \mathrm{neg}(x_{18})$
$\overline{f}_{18} = x_7 \odot \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14})$
$\overline{f}_{u_1} = \mathrm{neg}(x_8)^2$
$\overline{f}_{u_2} = x_2 \odot x_7 \odot x_8$
$\overline{f}_{u_3} = \mathrm{neg}(u_2)^2$
$\overline{f}_{u_4} = u_1 \odot \mathrm{neg}(x_2)$
$\overline{f}_{u_5} = \mathrm{neg}(x_{13}) \odot \mathrm{neg}(c')$
$\overline{f}_{u_6} = x_8 \odot \ c''$
$\overline{f}_{u_7} = x_2 \odot \ c''$
$\overline{f}_{u_8} = \mathrm{neg}(u_6) \odot \mathrm{neg}(u_7)$

Nuevamente, siguiendo las reducciones de 3.2, reemplazamos $x_2 = \mathrm{neg}(u_1)$,$x_4 = \mathrm{neg}(u_3)$, $x_7 = \mathrm{neg}(u_4)$, $x_8 = \mathrm{neg}(u_5)$, $x_{13} = \mathrm{neg}(u_8)$ y eliminamos $\overline{f}_2$, $\overline{f}_4$, $\overline{f}_7$, $\overline{f}_8$, $\overline{f}_{13}$.

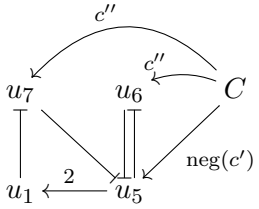
Llamamos g_i a las funciones obtenidas después de aplicar las reducciones a $\overline{f}_i$.

$$
\begin{aligned}
g_5 &= \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14}) \odot \mathrm{neg}(u_3)\\
g_{10} &= \mathrm{neg}(x_5) \odot \mathrm{neg}(u_1) \odot u_4 \odot u_5\\
g_{14} &= x_{10} \odot \mathrm{neg}(x_{18})\\
g_{18} &= \mathrm{neg}(x_{10}) \odot \mathrm{neg}(x_{14}) \odot \mathrm{neg}(u_4)\\
g_{u_1} &= u_5^2\\
g_{u_2} &= \mathrm{neg}(u_1) \odot \mathrm{neg}(u_4) \odot \mathrm{neg}(u_5)\\
g_{u_3} &= \mathrm{neg}(u_2)^2\\
g_{u_4} &= u_1^2\\
g_{u_5} &= u_8 \odot \mathrm{neg}(c')\\
g_{u_6} &= \mathrm{neg}(u_5) \odot\ c''\\
g_{u_7} &= \mathrm{neg}(u_1) \odot\ c''\\
g_{u_8} &= \mathrm{neg}(u_6) \odot \mathrm{neg}(u_7)
\end{aligned}
$$



Para poder reducir aún más las ecuaciones de punto fijo, con la ayuda de la observación 3.3 y las flechas pintadas con rojo y azul en el digrafo de red de la derecha, tenemos que $x_{10} = 0$ y, al reemplazar esta información en g_{14} también tenemos $x_{14} = 0$. Hacemos los reemplazos correspondientes y entonces eliminamos g_{10} y g_{14}. Luego proseguimos al reemplazar $x_5 = \mathrm{neg}(u_3)$ y $x_{18} = \mathrm{neg}(u_4)$, y eliminamos g_5 y g_{18}. Como u_3 no está involucrado en ninguna de las ecuaciones restantes, definimos $u_3 = \mathrm{neg}(u_2)^2$ y eliminamos g_{u_3}. Con el mismo razonamiento, reemplazamos $u_2 = \mathrm{neg}(u_1) \odot \mathrm{neg}(u_4) \odot \mathrm{neg}(u_5)$ y eliminamos g_{u_2}, y $u_4 = u_1^2$ y eliminamos g_{u_4}. Más aún reemplazamos $u_8 = \mathrm{neg}(u_6) \odot \mathrm{neg}(u_7)$, y eliminamos g_{u_8}. Llamamos h_i a las funciones obtenidas luego de aplicar las reducciones propuestas:

$$
\begin{aligned}
h_{u_1} &= u_5^2\\
h_{u_5} &= \mathrm{neg}(u_6) \odot \mathrm{neg}(u_7) \odot \mathrm{neg}(c')\\
h_{u_6} &= \mathrm{neg}(u_5) \odot\ c''\\
h_{u_7} &= \mathrm{neg}(u_1) \odot\ c''
\end{aligned}
\qquad (3.2.1)
$$



Nos vamos a referir al sistema (3.2.1) como el *sistema reducido*.

Hasta ahora tenemos:

$$
\begin{aligned}
x_1 = x_6 = x_9 &= c,\\
x_2 &= \mathrm{neg}(u_1),\\
x_4 = x_5 = x_{17} &= \mathrm{neg}(u_3),\\
x_3 = x_7 = x_{18} &= \mathrm{neg}(u_4),\\
x_8 = x_{19} &= \mathrm{neg}(u_5),\\
x_{10} = x_{14} &= 0,\\
x_{11} = x_{12} = x_{16} &= c^2,
\end{aligned}
$$

$$x_{13} = \text{neg}(u_8),$$
$$x_{15} = (x_{18} \odot x_5 \odot x_{17}) \oplus (x_5 \odot x_{17}),$$

y

$$u_2 = \text{neg}(u_1) \odot \text{neg}(u_4) \odot \text{neg}(u_5),$$
$$u_3 = \text{neg}(u_2)^2,$$
$$u_4 = u_1^2,$$
$$u_8 = \text{neg}(u_6) \odot \text{neg}(u_7).$$

Notar que $c^2 = \text{máx}\{0, 2c-1\}$. Entonces,

- Si $c \leq \frac{1}{2}$, entonces $c^2 = 0$ y $c' = c$, $c'' = 1$. Obtenemos entonces $u_6 = \text{neg}(u_5)$, $u_7 = \text{neg}(u_1)$ y podemos eliminar h_{u_6} y h_{u_7} y obtener $h_{u_1} = u_5^2$, $h_{u_5} = u_5 \odot u_1 \odot \text{neg}(c)$. Yendo un paso más y reemplazando $u_1 = u_5^2$ nos queda una sola ecuación de punto fijo:
$$u_5 = u_5^3 \odot \text{neg}(c).$$
  - Si $u_5 \leq \frac{2}{3}$, entonces $u_5^3 = 0$ y entonces necesariamente $u_5 = 0$. Esto da el siguiente punto fijo:
$$x_1 = x_6 = x_9 = c,$$
$$x_{10} = x_{11} = x_{12} = x_{14} = x_{16} = 0,$$
$$x_2 = x_3 = x_4 = x_5 = x_7 = x_8 = x_{13} = x_{15} = x_{17} = x_{18} = x_{19} = 1.$$
  - Si $u_5 > \frac{2}{3}$ entonces $u_5 = \text{máx}\{0, 3u_5 + (1-c) - 3\} = 3u_5 + (1-c) - 3 \Leftrightarrow \ 2u_5 = 2 + c \ \Leftrightarrow \ u_5 = 1 + \frac{c}{2}$. Esto solo puede suceder si $u_5 = 1$ y $c = 0$, y entonces, $c' = 0$, $c'' = 1$, y por substitución obtenemos $x_i = 0$ para $i \in \{1, \cdots, 19\}$.

- Si $c > \frac{1}{2}$, entonces $c^2 = 2c - 1$ y $c' = \text{mín}\{1, 7c-3\}$.
  - Más aún, si $c \geq \frac{4}{7}$, entonces $c' = 1$, $\text{neg}(c') = 0$ y esto da $u_5 = 0$.

    Deducimos entonces $u_1 = 0$, $u_6 = u_7 = c'' = \text{neg}(c^2) = 2 - 2c$ y $u_8 = c^4 = \text{máx}\{0, 4c-3\}$.

    Reemplazando estos valores obtenemos el único punto fijo:
$$x_1 = x_6 = x_9 = c,$$
$$x_{11} = x_{12} = x_{16} = 2c - 1,$$
$$x_{13} = \text{neg}(c^4),$$
$$x_{10} = x_{14} = 0,$$
$$x_2 = x_3 = x_4 = x_5 = x_7 = x_8 = x_{15} = x_{17} = x_{18} = x_{19} = 1.$$

- Si $c \in (\frac{1}{2}, \frac{4}{7})$, entonces $0 < c^2 = 2c - 1 < \frac{1}{7}$ y $c'' = \text{neg}(c^2) = 2(1-c)$.

  Es fácil chequear que $u_1 = u_5 = 0$, $u_6 = u_7 = 2(1-c)$ es un punto fijo del sistema reducido y, reemplazando, obtenemos:

$$x_1 = x_6 = x_9 = c,$$

$$x_{11} = x_{12} = x_{16} = 2c - 1,$$

$$x_{10} = x_{14} = 0,$$

$$x_2 = x_3 = x_4 = x_5 = x_7 = x_8 = x_{13} = x_{15} = x_{17} = x_{18} = x_{19} = 1.$$

  En efecto, chequeamos que es el único punto fijo en 2 casos. Con $m = 9$ y $c = 5/9$ (y entonces $c' = c'' = 8/9$), y con $m = 13$ y $c = 7/13$ (y $c' = 10/13, c'' = 12/13$), chequeado usando nuestra implementación [11].

  Por ejemplo para el caso de $m = 9, c = \frac{5}{9}$ la matriz que utilizamos para el programa para representar las ecuaciones de (3.2.1) es

$$\begin{pmatrix} 0 & 2 & 0 & 0 & -1 \\ 0 & 0 & -1 & -1 & \frac{1}{9} \\ 0 & -1 & 0 & 0 & \frac{8}{9} \\ -1 & 0 & 0 & 0 & \frac{8}{9} \end{pmatrix}$$

  El punto fijo que obtenemos es

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
$\frac{5}{9}$	1	1	1	1	$\frac{5}{9}$	1	1	$\frac{5}{9}$	0	$\frac{1}{9}$	$\frac{1}{9}$	1	0	1	$\frac{1}{9}$	1	1	1

  Más detalles al respecto podrán verse en los siguientes capítulos.

  En este pequeño ejemplo e incluso con todas estas reducciones, estos cómputos ya son grandes para hacerlos manualmente.

En la siguiente sección vamos a explicar cómo computar los puntos fijos sistemáticamente.

3.3. Algoritmo de puntos fijos

Dados $X_m = \{0, \frac{1}{m}, \ldots, \frac{m-1}{m}, 1\}$ con $m \geq 1$ arbitrario, y una red multivaluada F sobre X_m. Nos vamos a enfocar en encontrar efectivamente los puntos fijos del sistema.

Teorema 3.6. *Dada una función* $F = (f_1, \dots, f_n)$ *de una red* $(\odot - \text{neg})$*, toda función*

$$f_i = \bigodot_{j \in A_i} x_j^{p_{ij}} \odot \bigodot_{j \in B_i} (\text{neg}(x_j))^{q_{ij}} \odot c_i, \ A_i, B_i \subseteq \{1, \dots, n\}, \ A_i \cap B_i = \emptyset,$$

con $p_{ij}, q_{i,j}$ *enteros en* $[1, m]$*, se puede escribir como*

$$f_i = max\{0, \ell_i(x)\}, \tag{3.3.1}$$

donde

$$\begin{aligned} \ell_i(x) &= \ell_i'(x) + c_i^*, \quad c_i^* \in (\mathbb{Z}\frac{1}{m}) \cap \mathbb{Q}_{\leq 1}, \\ \ell_i'(x) &= \sum_{j \in A_i} p_{ij} x_j - \sum_{j \in B_i} q_{ij} x_j, \quad y \quad c_i^* = c_i - \sum_{j \in A_i} p_{ij}. \end{aligned} \tag{3.3.2}$$

Demostración. Notemos que, por lo mencionado en (ii) de la Proposición 2.5,

$$\begin{aligned} x_1 \odot x_2 \cdots \odot x_n &= max\{0, x_1 + \cdots + x_n - (n-1)\} \\ x^n &= max\{0, nx - (n-1)\}. \end{aligned}$$

Por lo tanto

$$\bigodot_{j \in A_i} x_j^{p_{ij}} = max\{0, \sum_{j \in A_i} p_{ij} x_j - \sum_{j \in A_i} p_{ij} + 1\}. \tag{3.3.3}$$

De manera similar,

$$(neg(x))^n = max\{0, -nx + 1\}.$$

Por lo tanto

$$\bigodot_{j \in B_i} (neg(x_j))^{q_{ij}} = max\{0, -\sum_{j \in B_i} q_{ij} x_j + 1\} \tag{3.3.4}$$

usando (3.3.3) y (3.3.4) y multiplicando por c_i obtenemos:

$$\begin{aligned} \bigodot_{j \in A_i} x_j^{p_{ij}} \bigodot_{j \in B_i} (neg(x_j))^{q_{ij}} \odot c_i &= max\{0, \sum_{j \in A_i} p_{ij} x_j - \sum_{j \in B_i} q_{ij} x_j - \sum_{j \in A_i} p_{ij} + 1 + c_i - 1\} \\ &= max\{0, \sum_{j \in A_i} p_{ij} x_j - \sum_{j \in B_i} q_{ij} x_j - \sum_{j \in A_i} p_{ij} + c_i, \} \end{aligned}$$

con lo que

$$\ell_i(x) = \sum_{j \in A_i} p_{ij} x_j - \sum_{j \in B_i} q_{ij} x_j + c_i - \sum_{j \in A_i} p_{ij}.$$

Como $c_i \in [0, 1] \cap \mathbb{Z}.\frac{1}{m}$ y $\sum_{j \in A_i} p_{ij} \in \mathbb{Z}_{\geq 0}$ deducimos $c_i^* \in (\mathbb{Z}\frac{1}{m}) \cap \mathbb{Q}_{\leq 1}$, que es lo que queríamos demostrar.

□

Lema 3.7. *Toda función lineal de la forma* $\ell = \sum_{j\in A} p_j x_j - \sum_{j\in B} q_j x_j + (c - \sum_{j\in A} p_j)$, *donde* $A, B \subseteq \mathbb{N}$ *son conjuntos finitos disjuntos,* $p_j, q_j \leq m$, $p_j, q_j \in \mathbb{N}$ *y* $c \in X_m$, *define una función* $\odot - \text{neg}$.

Demostración. La función definida por $f = \bigodot_{j\in A} x_j^{p_j} \odot \bigodot_{j\in B} (\text{neg}(x_j))^{q_j} \odot c$ verifica todas las hipótesis. Cualquier otra función que verifique todas las hipótesis va a coincidir con f en todo punto. □

Observación 3.8. Ya vimos en la Observación 2.13 que la expresión de una función $\odot$-neg puede no ser única. En el Lema 5.7 y la Proposición 5.14 demostraremos que para toda función $\odot$-neg

$$f = \bigodot_{j\in A} x_j^{p_j} \odot \bigodot_{j\in B} \text{neg}(x_j)^{q_j} \bigodot c,$$

la constante c es única y f puede expresarse de manera única con todos los exponentes p_i, q_j entre 1 y c.

3.3.1. Algoritmo para calcular los estados estables

Proponemos el siguiente algoritmo para computar los estados estables de una red $(\odot - neg)$. El input del algoritmo va a ser una red $(\odot - neg)$ $F : X^n \longrightarrow X^n$. El output es una lista con los estados estables de F.

La idea del algoritmo será mirar todos los posibles casos en donde $l_i \geq 0$ o $l_i \leq 0$. Como hay n funciones l_i, hay 2^n posibles casos. Cada uno de estos casos es un sistema de ecuaciones distinto, al cual le vamos a buscar sus soluciones en X^n:

$$\begin{cases} l_i(x) = x_i & \text{si } i \in I \\ 0 = x_i & \text{si } i \notin I \end{cases} \tag{3.3.5}$$

donde $I = \{i \in \{1, ..., n\} : l_i(x) \geq 0\}$

Algorithm 2 Computamos los puntos fijos de una red $\odot$-neg con al menos una operación $\odot$ en cada f_i.

Input: Una red $\odot$-neg $F = (f_1, \ldots, f_n) : X_m^n \to X_m^n$, con $f_i = max\{0, \ell_i(x)\}$ como en (3.3.1).

Output: Los puntos fijos de F.

Paso 1: Para cada $I \subseteq \{1, \ldots, n\}$ consideremos la región

$$\mathcal{R}_I = \left\{ \begin{array}{l} \ell_i(x) \geq 0 \; i \in I \\ \ell_j(x) \leq 0 \; j \notin I \end{array} \right\}.$$

Paso 2: Resolvamos el sistema lineal $\left\{ \begin{array}{rl} \ell_i(x) = x_i & i \in I \\ 0 = x_j & j \notin I \end{array} \right.$

En el caso en el que haya una única solución racional, chequeemos si $x \in X_m^n$ y verifiquemos si $\ell_j(x) \leq 0$ para todo $j \notin I$. Si esto se satisface, entonces x es un punto fijo de F en $\mathcal{R}_I$.

Paso 3: En el caso en el que el sistema lineal tenga infinitas soluciones racionales, cambiemos las variables por $y_i = mx_i$, traduzcamos las formas lineales $\ell_i - x_i$ a formas lineales enteras $\bar{\ell}_i(y) - y_i$ y encontremos los puntos de reticulado en el polítopo racional

$$P_I = \{\bar{\ell}_i(y) - y_i = 0, i \in I, y_j = 0, j \notin I, 0 \leq y_i \leq m, i \in I, \bar{\ell}_j(y) \leq 0, j \notin I\}.$$

Traduzcamos de vuelta las soluciones a soluciones $x \in X_m^n$.

Observación 3.9. El Paso 2 y el 3 **no son necesarios cuando la red es Booleana** ($m = 1$) sino que se restringe a una argumentación sobre los índices que toman los valores 0 o 1.

Deducimos del Teorema 3.6 la validez del Algoritmo 2 debajo para computar los puntos fijos de cualquier función de red $(\odot - \text{neg})$ $F : X_m^n \to X_m^n$.

Primero, sea J el conjunto de índices i para los cuales $f_i = \ell_i$ consiste de una constante, una sola variable o su negación. Por supuesto, para cada una de estas funciones lineales podemos escribirlas como máx$\{0, \ell_i(x)\}$, pero esto introduciría ramas innecesarias en los cómputos.

Cuando J es no vacío, podemos aplicar reducciones (i) y (ii) en 3.2 y dejar de lado las variables con índices en J. Notar que si empezamos con una función $\odot$-neg, obtenemos una función reducida $f_{red} : X_m^{n-|J|} \to X^{n-|J|}$ que también es de esta forma. Entonces, la cantidad de regiones a considerar en el Paso 1 del siguiente Algoritmo 2 es $2^{n-|J|}$. El sistema lineal a resolver en el Paso 2 del Algoritmo también tiene a lo sumo $n - |J|$ variables. En el Paso 3, también agregamos los valores de las variables con índices en J como salida de los puntos fijos. Para alivianar la notación asumimos que $J = \emptyset$ pero el input del Algoritmo 2 debería ser la función reducida f_{red} y no f.

Ahora, cómo hace uno para algorítmicamente encontrar todos los puntos de reticulado en polítopos racionales? Una buena implementación del algoritmo de Barvinok mencionado en la introducción fue provista por el software gratuito Latte [12],

explicado en el paper [8] (ahora disponible en SageMath). Usamos la implementación de [10], explicada en el reporte [30] que cuenta los puntos de reticulado y los provee explícitamente.

Notar que para cualquier $m \geq 1$ hay a lo sumo $2^{n-|J|}$ regiones $\mathcal{R}_I$ (independientemente de m), mientras la búsqueda exhaustiva de puntos fijos requiere $(m+1)^n$ estados iniciales.

Más aún, los cómputos son paralelizables para distintos I y mayores simplificaciones y mejoras en el árbol de cómputos se pueden realizar para instancias particulares.

El mejor caso posible es cuando el sistema linear cuadrado del Paso 2 tiene solución única, necesariamente racional. En este caso, resolver cada sistema tiene complejidad de orden $O(|I|^3)$ y entonces chequeamos si su solución cae en X_m^n.

Damos en el teorema 3.11 debajo un simple condición general bajo la que esto sucede. Notemos igual que cuando el rango no es máximo, la complejidad de Paso 2 puede crecer, hasta el orden de $(m+1)^n$.

Definición 3.10. *Consideremos un función de red* $\odot$-neg $F = (f_1, \ldots, f_n)$ *con al menos una operación* $\odot$ *en cada* f_i. *Dado* $I \subseteq \{1, \ldots, n\}$ *de cardinalidad* s *denotemos* M_I *la matriz* $s \times s$ *con filas y columnas indicadas por* I *y con valores:*

$$(M_I)_{ij} = \begin{cases} \ell_{ii} - 1 & \text{si } i = j, \\ \ell_{ij} & \text{si } i \neq j \end{cases} \tag{3.3.6}$$

Teorema 3.11. *Dada una red* $(\odot - \text{neg})$ *definida por* ℓ_i $i \in \{1, \ldots, n\}$. *Sea* $J \subset \{1, \ldots, n\}$ *denotemos el conjunto de índices* i *en el cual* f_i *no tiene operaciones* $\odot$.

Si para $I \subseteq \{1, \ldots, n\}$ *con* $I \subset J^c$, *la matriz* M_I *definida en* (3.3.6) *satisface* $\det(M_I) \neq 0$, *entonces hay a lo sumo un punto fijo en la región* $\mathcal{R}_I$.

La demostración del Teorema 3.11 es inmediata, ya que M_I es la matriz del sistema lineal $\begin{cases} \ell_i(x) = x_i & i \in I \\ 0 = x_j & j \notin I \end{cases}$.

El siguiente corolario también es inmediato.

Corolario 3.12. *Sea* $L' \in \mathbb{Z}^{n \times n}$ *la matriz de coeficientes de las formas lineales* $\ell'_1, \ldots, \ell'_n$ *como en el Teorema 3.6 y* Id_n *sea la matriz identidad* $n \times n$.

Si $\det(L' - Id_n) \neq 0$ *entonces existe a lo sumo un punto fijo con todas sus coordenadas no nulas.*

Terminamos esta sección mostrando cómo computar los puntos fijos utilizando nuestra aplicación [11].

Ejemplo 3.13. Consideremos nuevamente el ejemplo biológico en 3.2.1, en el caso $m = 9$, $c = 5/9, c' = c'' = 8/9$. El sistema reducido (3.2.1) tiene 4 variables (ordenadas u_1, u_5, u_6, u_7). Se puede escribir como:

$$\begin{aligned} h_{u_1} &= \text{máx}\{0, 2u_5 - 1\} \\ h_{u_5} &= \text{máx}\{0, -u_6 - u_7 + 1/9\} \\ h_{u_6} &= \text{máx}\{0, -u_5 + 8/9\} \\ h_{u_7} &= \text{máx}\{0, -u_1 + 8/9\} \end{aligned}$$

Después de instalar nuestra aplicación *Puntos fijos de redes MV* del repositorio de GitHub [11], introducimos los coeficientes de las formas lineales en cada función, para conseguir el output como en la Figura 3.1. Por supuesto, hay mucho que se puede mejorar en esta implementación.

Nuestra implementación del algoritmo 2 2, de la cual hablaremos en detalle más adelante, se ve de la siguiente manera para este ejemplo:

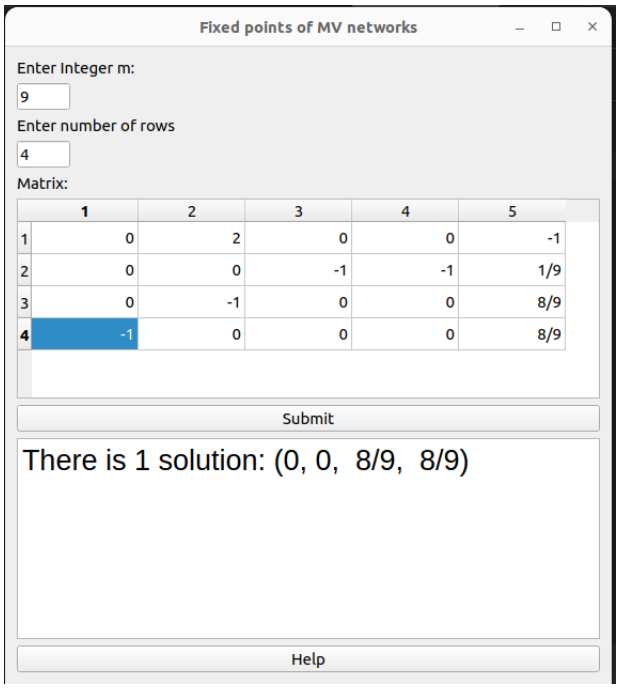


Figura 3.1: Screenshot de la aplicación [11]

3.4. Algoritmo de Barvinok

Comenzamos recordando algunas definiciones básicas.

Definición 3.14. *Llamamos standard integer lattice $\mathbb{Z}^d \subset \mathbb{R}^d$ al conjunto de puntos de coordenadas enteras.*

Definición 3.15. *Sea V un $\mathbb{R}$-espacio vectorial de dimensión finita. Llamamos poliedro $P \subset V$ al conjunto de puntos que satisfacen un conjunto finito de inecuaciones lineales, es decir*

$$P = \{x \in V : l_i(c) \leq \alpha_i, i \in I\} \tag{3.4.1}$$

para un conjunto finito I, donde $l_i : V \longrightarrow \mathbb{R}$ son funciones lineales con coeficientes reales y $\alpha_i \in \mathbb{R}$. Cuando los coeficientes de las formas lineales l_i y las constantes α_i son números racionales, el poliedro se llama racional.

Notemos que un poliedro racional siempre puede definirse con formas lineales con coeficientes enteros.

Definición 3.16. *Un polítopo es un poliedro acotado. Sea $P \subset V$ un poliedro y sea $v \in P$ un punto. El punto v es un vértice de P si cada vez que $v = \frac{v_1+v_2}{2}$ con $v_1, v_2 \in P$ entonces $v = v_1 = v_2$.*

3.4.1. Representaciones compactas de los puntos enteros

En 1994 Barvinok [30][4] dio un algoritmo que cuenta los puntos de coordenadas enteras de un polítopo racional convexo en tiempo polinomial cuando la dimensión del polítopo es fija. La primera implementación de dicho algoritmo fue hecha por De Loera y coautores [8] y está actualmente también disponible en el software libre SageMath [13].

Dado un polítopo racional $P = \{u \in \mathbb{R}^d : Au \leq b\}$, la idea es representar los puntos enteros de P como los exponentes del polinomio de Laurent:

$$f(P; x) = \sum_{\alpha \in P \cap \mathbb{Z}^d} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_d^{\alpha_d}.$$

Dada esta representación, la cantidad de puntos enteros en P se obtiene como la evaluación $f(P; (1, \dots, 1)) = \#P \cap \mathbb{Z}^d$, aunque por la forma de obtener el polinomio $f(P, x)$ no es inmediata.

Esta idea tiene sus orígenes en el trabajo de Michel Brion [5], a partir del cual se pueden representar las funciones generadoras de los puntos de coordenadas enteras de un polítopo racional como la *suma formal* de funciones racionales asociadas a los vértices del polítopo, que tienen un significado geométrico.

Para cada vértice v de un poliedro racional P, se considera la función racional $f(K(P, v); x)$ que tiene como exponentes los puntos enteros del menor cono poliedral racional $K(P, v)$ que contiene a P con vértice en v:

$$K(P, v) = v + \{u \in \mathbb{R}^d : v + \delta u \in P, \text{ para todo } \delta > 0 \text{ suficientemente chico}\}.$$

Cuando $K(P, v)$ es simplicial (es decir que de v emanan tantos ejes como la dimensión de P), la expresión de esta función es conocida [19, Cap. 4] y es particularmente sencilla cuando v es regular, es decir cuando $K(P, v)$ es simplicial y además los vectores $u_1, \ldots, u_n$ que generan los conos sobre los ejes desde v sobre $\mathbb{Z}$ forman una base de $\mathbb{Z}^n$. En este caso

$$f(P, v) = \frac{x^v}{(1 - x^{u_1}) \ldots (1 - x^{u_n})}.$$

Observación 3.17. La clave para generar estas series formales es recordar que

$$\frac{1}{1 - x} = \sum_{m \geq 0} x^m, \quad \text{para } |x| < 1.$$

Notar que son todas series formales con distintos dominios de convergencia. Sin embargo al sumar todas, se obtiene la suma finita $f(P)$.

Cuando $K_{P,v}$ es simplicial, esta función fue también descripta en [19] y tiene el mismo denominador y aparecen otros monomios en el numerador (tantos como del determinante $\det(u_1, \ldots, u_v)$). Algo a destacar aquí es que todo cono en dimensión 2 es simplicial. A partir de dimensión 3 esto ya no es cierto. En dimensión 3 existen conos con más de 3 generadores. En estos casos, la descomposición necesita subdividir el cono, pero deben luego restarse adecuadamente los puntos enteros de caras comunes a más de un cono.

Sea $\{u_1, u_2, ..., u_k\}$ un conjunto de vectores enteros linealmente independientes de $\mathbb{R}^d$, donde $k \leq d$. Sea K el cono generado por $\{u_1, u_2, ..., u_k\}$, en otras palabras, $K = \{\lambda_1 u_1 + \lambda_2 u_2 + ... + \lambda_k u_k$, donde $\lambda_i \geq 0$ e $i = 1, 2, ..., k\}$. Consideremos el paralelepípedo $S = \{\lambda_1 u_1 + \lambda_2 u_2 + ... + \lambda_k u_k, 0 \leq \lambda_i \leq 1, i = 1, 2, ..., k\}$. Entonces, la función generadora de los puntos de coordenadas enteras de K es la siguiente función racional:

$$\sum_{\beta \in K \cap \mathbb{Z}^d} \zeta^\beta = \left(\sum_{\tau \in S \cap \mathbb{Z}^d} \zeta^\tau \right) \prod_{i=1}^{k} \frac{1}{1 - \zeta^{u_i}} \tag{3.4.2}$$

Para poder llegar a una fórmula general para conos poliedrales racionales con vértice, es preciso subdividir el dato en conos simpliciales. Barvinok propuso una descomposición simplicial y una suma de estas funciones racionales con signos apropiados.

En el ejemplo 3.18 que sigue, damos a modo ilustrativo estas funciones y cómo a partir de ellas se obtiene el polinomio $f(P; x)$.

Ejemplo 3.18. Consideremos el cuadrilátero de la Figura 3.2.

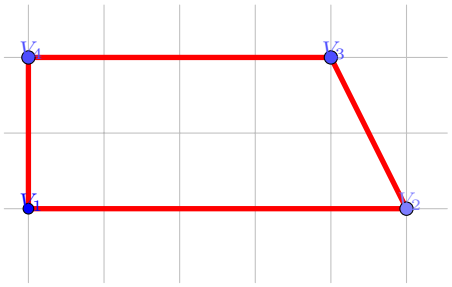


Figura 3.2: **Cuadrilátero del ejemplo: 3.18**

Sus vértices son $V_1 = (0,0)$, $V_2 = (5,0)$, $V_3 = (4,2)$ y $V_4 = (0,2)$. En este caso, obtenemos cuatro funciones racionales (una por cada vértice de P) y sumando sus desarrollos formales en serie se obtiene el polinomio buscado:

$$f(P;x,y) = f(K_{v_1};x,y) + f(K_{v_2};x,y) + f(K_{v_3};x,y) + f(K_{v_4};x,y)$$

$$f(K_{V_1};x,y) = \frac{1}{(1-x)(1-y)} = \sum_{m.n\geq 0} x^m y^n,\ |x|<1, |y|<1.$$

$$f(K_{V_2};x,y) = \frac{(x^5+x^4y)}{(1-x^{-1})(1-y^2x^{-1})} = \sum_{m\geq 1, n\geq 0} x^{5-m}(x^4y)^n,\ |x|>1, |y^2x^{-1}|<1.$$

$$f(K_{V_3};x,y) = \frac{(x^4y+x^4y^2)}{(1-x^{-1})(1-xy^{-2})} = \sum_{m\geq 1, n\geq 1} x^{4-m}(yx)^{n-1},\ |x|>1, |xy^{-2}|<1$$

$$f(K_{V_4};x,y) = \frac{y^2}{(1-y^{-1})(1-x)} = \sum_{m\geq 0, n\geq 2} x^m y^{-n},\ |x|<1, |y|>1.$$

Como mencionamos, la forma de conseguir estas funciones racionales consiste en tomar cada uno de los vértices y observar el cono generado a partir de ese vértice. Por ejemplo, en $V_1 = (0,0)$, tenemos las direcciones generadas sobre $\mathbb{Z}$ por $u_1 = (1,0)$ y $u_2 = (0,1)$ generando el cono K_{V_1}, que forman una base de $\mathbb{Z}^2$ (y similarmente en el vértice V_4). Como el determinante de la matriz generada por estas direcciones es 1, entonces la función generadora es $f(K_{V_1};x,y) = \frac{x^{v_1}y^{v_2}}{(1-x^{u_{1,1}}y^{u_{1,2}})(1-x^{u_{2,1}}y^{u_{2,2}})}$.

Si el determinante de la matriz (u_1,u_2) no hubiera sido 1 o -1, tendríamos que haber agregado monomios correspondiente a los puntos enteros del paralelogramo abierto desde el vértice. En el caso de V_2, los generadores del cono desde ese vértice son los vectores $(-1,0)$ y $(-1,2)$. Consideramos el paralelogramo $v + \{\lambda_1(-1,0) + \lambda_2(-1,2),\ 0 \leq \lambda_1, \lambda_2 \leq 1\}$, que tiene el punto $(4,1)$ entero interior, por lo que debemos agregarlo como se ve en el numerador de $f(K_{V_2};x,y)$.

En este ejemplo, el resultado de sumar los desarrollos en serie formales de estas funciones racionales es igual al polinomio $f(P;x,y)$:

$$x^5+x^4y+x^4+x^4y^2+yx^3+x^3+x^3y^2+yx^2+x^2+x^2y^2+xy+x+xy^2+y^2+y+1.$$

Notar que cada uno de los monomios de este polinomio representa en sus exponentes cada uno de los puntos de coordenadas enteras dentro de nuestro cuadrilátero. Y, como es esperado, $f(P;(1,1)) = 16$, ya que nos quedan 16 monomios correspondientes con los 16 puntos de coordenadas enteras.

Se pueden leer más detalles sobre esta descomposición en [30]. A continuación reseñamos la implementaciones disponibles.

3.5. Implementaciones disponibles

3.5.1. LattE

El nombre del programa, LattE, es una abreviación de **Latt**ice point **E**numeration, que significa enumeración de puntos de coordenadas enteras.

Latte fue desarrollado en 2001 con el objetivo de contar puntos de coordenadas enteras en un poliedro convexo definido por ecuaciones e inecuaciones lineales con coeficientes enteros. El poliedro puede ser de cualquier dimensión.

La implementación de LattE de Barvinok se centra en dos ideas: el uso de funciones racionales como una estructura de datos eficiente y en la descomposición signada de conos en conos unimodulares. Notar que como P es un polítopo (es decir, un poliedro acotado), los monomios de $f(P)$ están en biyección con los puntos de coordenadas enteras y $f(P)$ es un polinomio de Laurent. Para contar los puntos de coordenadas enteras, LattE usa las funciones racionales de Barvinok.

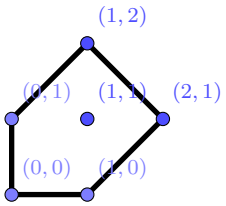
La función que usaremos de Latte es count, que hace precisamente lo antes mencionado.

Observación 3.19. Sea P un polítopo descripto por un sistema de inecuaciones $Ax \leq b$ donde $A \in \mathbb{Z}^{mxd}$, $A = (a_{ij})$ y $b \in \mathbb{Z}^m$.

Con $P = \{x : Ax \leq b\}$, el archivo de input para LattE es:

```
m  d+1
b  -A
```

Ejemplo 3.20. Sea $P = \{(x,y) : -x + y \leq 1, -x + y \geq -1, x + y \leq 3, x \geq 0, y \geq 0\}$



Reescribimos las inecuaciones para que todas queden con el mismo signo:

$$-x \leq 0$$

$$-y \leq 0$$
$$-x + y \leq 1$$
$$x - y \leq 1$$
$$x + y \leq 3.$$

Esto se puede ver entonces como la solución del sistema $Ax \leq b$, donde A y b son

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 3 \end{pmatrix}.$$

El archivo de input de LattE correspondiente a este sistema es

$$\begin{array}{rrr} 5 & 3 & \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ 3 & -1 & -1. \end{array}$$

Ejemplo 3.21. Sea $P = \{(x, y) : x \leq 1,\ y \leq 1,\ x + y = 1,\ x \geq 0,\ y \geq 0\}$. Esto se puede escribir como la solución del sistema $Ax \leq b$, donde A y b son

$$A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Y el archivo de input de LattE sería:

```
5 3
1 -1  0
1  0 -1
1 -1 -1
0  1  0
0  0  1
linearity 1 3
```

Observación 3.22. En general, la sintaxis para declarar cuales son las condiciones que son lineales es:

```
linearity <cantidad de ecuaciones> <indices de las
   filas>
```

Los indices de las filas arrancan en 1.

3.5.2. Barvinok

Barvinok, al igual que LattE, es otra librería que usa la descomposición de Barvinok. Se puede descargar y leer su manual de [33]. Con Barvinok, además de contar los puntos enteros en un polítopo, vamos a listarlos. La función que usaremos de Barvinok es polytope_scan. El archivo que se usa como input para este programa usa notación Polylib. En la librería existe un script que transforma la notación de LattE a notación PolyLib, latte2polylib.pl. Nosotros vamos a utilizar directamente notación PolyLib.

Notación PolyLib

Vamos a ilustrar nuestro uso de PolyLib con un ejemplo.

Ejemplo 3.23. La siguiente matriz como input

```
5  4
1  0  1  -1
1  -1 0  6
1  0  -1 7
1  1  0  -2
0  1  1  2
```

representaría las siguientes 5 ecuaciones e inecuaciones, con 4 valores por fila, el primero representando si estamos ante una ecuación o inecuación, 2 variables y una constante:

Significado de la primera columna	Significado de la línea completa
$1 =$ inecuación,	$0x + 1y - 1 \geq 0 \longrightarrow y \geq 1$
$1 =$ inecuación,	$-1x + 0y + 6 \geq 0 \longrightarrow x \leq 6$
$1 =$ inecuación,	$0x - 1y + 7 \geq 0 \longrightarrow y \leq 7$
$1 =$ inecuación,	$1x + 0y - 2 \geq 0 \longrightarrow x \geq 2$
$0 =$ ecuación,	$1x + 1y + 2 = 0 \longrightarrow x + y = -2$

3.6. Nuestra implementación

3.6.1. Uso del entorno gráfico de nuestro programa

Para describir el uso del entorno gráfico de nuestro programa, vamos a usar el siguiente ejemplo:

Sea f nuestra red $\odot -$ neg con $m = 3$ escrita en términos de operadores $\odot$ y neg:

$$
\begin{aligned}
f_{u_1} &= u_5^2 \\
f_{u_2} &= neg(u_1) \odot neg(u_4) \odot neg(u_5) \\
f_{u_3} &= neg(u_2)^2 \\
f_{u_4} &= u_1^2
\end{aligned}
$$

$$f_{u_5} = u_1 \odot u_5 \odot neg(\frac{1}{3}).$$

Se puede ver que la red del ejemplo del ciclo celular de los mamíferos de 3.2.1 puede ser llevada a esta función mediante un conjunto de reducciones distinto al ejemplificado en dicha sección.

En este caso, las ecuaciones de punto fijo se pueden describir como

$$\begin{aligned}
u_1 &= max\{0, 2u_5 - 1\} \\
u_2 &= max\{0, -u_1 - u_4 - u_5 + 1\}) \\
u_3 &= max\{0, -2u_2 + 1\} \\
u_4 &= max\{0, 2u_1 - 1\} \\
u_5 &= max\{0, u_1 + u_5 - \frac{4}{3}\}.
\end{aligned}$$

Sea $m+1$ la cantidad de estados posibles para cada variable, usualmente representado como $0, \frac{1}{m}, \frac{2}{m}, \ldots, 1$. En este ejemplo tenemos que poner como input $m = 3$ y la siguiente matriz, describiendo las ecuaciones de punto fijo, como input:

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 2 & -1 \\
-1 & 0 & 0 & -1 & -1 & 1 \\
0 & -2 & 0 & 0 & 0 & 1 \\
2 & 0 & 0 & 0 & 0 & -1 \\
1 & 0 & 0 & 0 & 1 & -4/3
\end{pmatrix}.$$

Para usar esta matriz como input, tendremos que poner la cantidad de filas de la matriz (en este caso, 5), y reemplazar los números correspondientes.

El output del programa sería el siguiente:

There is one solution:

- *(0, 1, 0, 0, 0)*

Esto significa que $(u_i) = (0, 1, 0, 0, 0)$ es el único punto fijo de la red.

El entorno gráfico está implementado en Python, usando la librería PyQt5. Pueden verse los detalles de implementación en [11].

3.6.2. Uso de las funciones del programa

Nuestro programa tiene una función principal:

- `compute_steady_states_barvinok(M, n, m)`, es la función que dada la función $(\odot - neg) - red$ devuelve la lista de estados estables.

El input de la función consiste de

- La función $(\odot - neg) - red$ en forma de matriz M.

- n, la cantidad de nodos de nuestra red biológica, que en este caso es la cantidad de f_i que tiene F.
- m, que es la cantidad de posibles estados que puede tener cada nodo.

Ejemplo 3.24. Veamos otro ejemplo para describir el uso de las funciones del programa sin el frontend.

Supongamos que tenemos la siguiente red $(\odot - neg)$

$$f_1 = max\{0, c2 - c1)\}$$
$$f_2 = max\{0, -2u_1 + 1)\}$$

si $m = 3$, $c1 = 1/3$ y $c2 = 2/3$ la matriz que pondremos en el programa es:

$$M = \begin{pmatrix} 0 & 0 & \frac{1}{3} \\ -2 & 0 & 1 \end{pmatrix}$$

donde la i-ésima fila de M corresponde a los coeficientes de la función lineal l_i

Con estos datos en mente, una posible forma de ejecutar la función principal del programa sería:

```
import barvinok

M = [[0, 0, 1/3], [-2,0,1]]
n = 2
m = 3
barvinok.compute_steady_states_barvinok(M, n, m)
> ['1, 1']
```

El resultado que devuelve está multiplicado por m. Es decir que esta red tiene una sola solución que es:

$$u_1 = \frac{1}{3}$$
$$u_2 = \frac{1}{3}$$

3.6.3. Descripción de la implementación

La función principal, compute_steady_states_barvinok

La función `compute_steady_states_barvinok(M, n, m)` recorre las 2^n regiones, y para cada una de ellas calcula el sistema al que le va a buscar los estados estables

```
86 def compute_steady_states_barvinok(M, n, m, reporter=
       None):
87     # Paso1: i es un nro entre 0 y 2^n, es decir todos
       los posibles subconjuntos de n elementos(es para
       armar las regiones R_I)
```

```
88      res = []
89      for i in range(1 << n):
90          if i % 100 == 0 and reporter is not None:
91              reporter.report(i)
92          solucion_parcial =
    compute_steady_states_for_subset(M, n, m, i)
93          res += solucion_parcial
94
95      return res
```

La función auxiliar `compute_steady_states_for_subset`

Esta función se encarga de hacer el trabajo para una de las 2^n regiones: llama a la función que traduce a notación PolyLib y corre el programa detrás.

```
79  def compute_steady_states_for_subset(M, n, m, subset):
80      ejemplo = guardar_instancia(M, n, m, subset)
81      res = correr_instancia(ejemplo)
82      return res
```

La función auxiliar `guardar_instancia`

Este función genera el archivo que es el input del programa que calcula puntos fijos para una de 2^n regiones posibles que describe el algoritmo 2.

Vamos a analizar esto en más detalle al ver un ejemplo, en 3.25

```
13  def guardar_instancia(M, n, m, subset):
14      file_name = '/tmp/instancia_%s_%s.txt' % (n, m)
15      len_subset = gmpy2.popcount(subset)
16      with open(file_name, 'w+') as output_file:
17          output_file.write(str(2 * n + len_subset) + ' '
    + str(n + 2) + '\n')
18          A = []
19          b = []
20          c = []
21          # construyo x=0 para lo que no esta en subset
22          for j in range(n):
23              # if j not in subset:
24              if subset & (1 << j) == 0:
25                  # condicion x_j=0
26                  l_j0 = [0] * n
27                  l_j0[j] = -1
28                  A.append(l_j0)
29                  b.append([0])
30                  c.append([0])
```

```
31          # construyo x>=1 y x<=m para el subset
32          for j in range(n):
33              # if j in subset:
34              if subset & (1 << j) != 0:
35                  # condicion x_j>0
36                  l_j0 = [0] * n
37                  l_j0[j] = 1
38                  A.append(l_j0)
39                  l_jm = [0] * n
40                  l_jm[j] = -1
41                  A.append(l_jm)
42                  b.append([-1])
43                  b.append([m])
44                  c.append([1])
45                  c.append([1])
46          # construyo Mx+c<=0 para lo que no esta en el
    subset
47          # Mx<=-c en este caso A =-M
48          for j in range(n):
49              # if j not in subset:
50              if subset & (1 << j) == 0:
51                  l_j = [-M[j][s] for s in range(n)]
52                  A.append(l_j)
53                  # multiplico por m para encontrar las
    soluciones enteras del sistema.
54                  # (asumimos que c* viene multiplicada
    por m para evitar problemas de precision)
55                  b.append([int(-M[j][-1])])
56                  c.append([1])
57
58          # armamos el sistema a resolver Ax +b=x <-> (A-
    I)x = -b
59          # (M-I)x=b para los casos del subset
60          # en este caso A= -(M-I) = I- M
61          for j in range(n):
62              # if j in subset:
63              if subset & (1 << j) != 0:
64                  l_j = [-M[j][s] if j != s else -M[j][s]
     + 1 for s in range(n)]
65                  A.append(l_j)
66                  # multiplico por m para encontrar las
    soluciones enteras del sistema.
67                  # (asumimos que c* viene multiplicada
    por m para evitar problemas de precision)
```

```
68                      b.append([int(-M[j][-1])])
69                      c.append([0])
70
71              for i in range(len(A)):
72                  output_file.write(str(c[i][0]) + ' ')
73                  for j in range(n - 1):
74                      output_file.write(str(A[i][j]) + ' ')
75                  output_file.write(str(A[i][n - 1]) + ' ' +
    str(b[i][0]) + '\n')
76          return file_name
```

Ejemplo 3.25. Mostramos a continuación uno de los archivos generados para el ejemplo 3.6.1. Este archivo es el input del programa que calcula puntos fijos para una de 2^n regiones posibles que describe el algoritmo 2. La región que describe este archivo es: $\mathcal{R}_{\{1,3,5\}} = \left\{ \begin{array}{l} \ell_i(u) \geq 0 \; i \in \{1,3,5\} \\ \ell_j(u) \leq 0 \; j \notin \{1,3,5\} \end{array} \right\}$.

```
13  7
0   0   -1  0   0   0   0
0   0   0   0   -1  0   0
1   1   0   0   0   0   -1
1   -1  0   0   0   0   3
1   0   0   1   0   0   -1
1   0   0   -1  0   0   3
1   0   0   0   0   1   -1
1   0   0   0   0   -1  3
1   1   0   0   1   1   -3
1   -2  0   0   0   0   3
0   1   0   0   0   -2  3
0   0   2   1   0   0   -3
0   -1  0   0   0   0   4
```

Recordando notación PolyLib, esto significa que

- La primera fila no indica cuantas filas y cuantas variables hay. En este ejemplo tenemos 13 ecuaciones con 7 variables en total.
- En todas las siguientes filas, la primera columna indica si es una ecuación o una inecuación.
  - si es 0 es porque es una ecuación
  - si es 1 es porque es una inecuación
- Las primeras 2 ecuaciones nos indican que $u_2 = u_4 = 0$, es decir que no se encuentran en nuestra región
- Las siguientes 6 filas describen, de a pares, que $1 \leq x_i \leq m$, para $i \in \{1,3,5\}$, es decir que están en nuestra región. Por ejemplo, la 3ra fila dice $x_1 - 1 \geq 0$ y la 4ta $-x_1 + 3 \geq 0$

- Las siguientes 2 líneas representan a quienes no están en la región. Por ejemplo, vemos $u_1+u_4+u_5-3 \geq 0$. que es equivalente a que $l_2(u) = -u_1-u_4-u_5+3 \leq 0$
- Las últimas 3 líneas representan los elementos que si están en la región por lo que no solo $l_i \geq 0$ sino que queremos que sea un punto fijo, es decir $l_i(u_i) = u_i$. Si escribimos a la función F en forma matricial $F(u) = M.u + b$ Entonces F tiene punto fijo si y solo si

$$M.u + b = u0 = (I - M)u - b \tag{3.6.1}$$

Como en nuestro ejemplo

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 2 & -1 \\ -1 & 0 & 0 & -1 & -1 & 1 \\ 0 & -2 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 1 & -4/3 \end{pmatrix}$$

recordando que b esta multiplicado por m. nos queda:

$$(I - M)u - b = \begin{pmatrix} 1 & 0 & 0 & 0 & -2 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 & 0 \\ -2 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{pmatrix} u + \begin{pmatrix} 3 \\ -3 \\ -3 \\ 3 \\ 4 \end{pmatrix}$$

Por ejemplo, la última línea viene a representar a nuestra ultima variable de nuestra región que es u_5, $((I - M)u - b)_5 = 0$, es decir, $-x_1 + 4 = 0$.

La función auxiliar `correr_instancia`

Esta función se encarga de hacer el llamado a Barvinok, utilizando como input el archivo en notación PolyLib antes generado.

```
5 def correr_instancia(archivo):
6     with open(archivo, "r") as input:
7         proc = subprocess.Popen(["./barvinok/
    polytope_scan"], stdin=input, stdout=subprocess.PIPE
    , close_fds=False)
8         output = proc.stdout.read().decode("utf-8")
9         return output.split('\n')[:-1]
```

4 Ejemplo: Denitrificación en Pseudomonas Aeruginosa

En este capítulo vamos a investigar la aplicación de nuestro modelo para una red de Denitrificación de Pseudomonas Aeruginosa.

4.1. Aplicación del modelo

En [1] los autores intentan descubrir los factores ambientales que contribuyen a la acumulación de gases invernadero N_2O, especialmente en Lake Erie (Laurentian Great Lakes, USA).

Su modelo predice el comportamiento de largo plazo del camino de denitrificación, teniendo 3 parámetros Booleanos externos (O_2, PO_4, NO_3), 4 variables booleanas ($PhoRB, PhoPQ, PmrA$,Anr), y 11 variables ternarias ($NarXL$, Dnr, $NirQ$, nar, nir, nor, nos, NO_2, NO, N_2O, N_2).

En nuestro contexto, suponemos que toman valores en X_2. Las correspondientes reglas de actualización en [1] son:

$$
\begin{array}{lll}
f_{PhoRB} = \text{neg}(PO_4), & f_{NarXL} = \text{mín}\{Anr, NO_3\}, & f_{NO} = \text{mín}\{NO_2, nir\}, \\
f_{PhoPQ} = PhoRB, & f_{nir} = \text{máx}\{NirQ, \text{mín}\{Dnr, NO\}\}, & f_{N_2O} = \text{mín}\{NO, nor\}, \\
f_{PmrA} = \text{neg}(PhoPQ), & f_{nor} = \text{máx}\{NirQ, \text{mín}\{Dnr, NO\}\}, & f_{N_2} = \text{mín}\{N_2O, nos\}, \\
f_{Anr} = \text{neg}(O_2), & f_{nos} = \text{mín}\{Dnr, NO\}. &
\end{array}
$$

Como no se pueden expresar precisamente 4 variables (Dnr, $NirQ$, nar y NO_2) usando *solo* los operadores MAX, MIN y NOT (indicados con un $*$ en [1, Table 3]), si bien en [1] hay una descripción utilizando dichos operadores, esta no se comporta como lo observado. Por ende, los autores del paper estuvieron forzados a mostrar todos los valores posibles en tablas suplementarias.

De las tablas de transición de cada una de las 15 variables, los autores construyen un sistema dinámico polinomial y computan los puntos fijos del sistema, con un software basado en cómputos de bases de Gröbner, bajo las condiciones ambientales de interés que emergen luego de fijar los valores de los parámetros externos.

Para mostrar la fortaleza de nuestra herramienta desarrollada en [14], modelamos su modelo con nuestra configuración. Para eso tradujimos las funciones mostradas arriba a operaciones con $\odot$ y $\oplus$ usando la proposición 2.5.

Para las funciones restantes,f_{Dnr}, f_{NirQ}, f_{nar} y f_{NO_2}, consideramos las tablas suplementarias en [1] sin el proceso de suavización, ya que los puntos fijos no cambian (ver lema 2.16).

Por ejemplo, los valores para $NirQ$ mostrados en la figura 4.1.

Podemos ver en la cuarta transición $(NarXL = 1, Dnr = 0) \mapsto NirQ = \frac{1}{2}$ no concuerda con la función máx$\{NarXL, Dnr\}$ in [1].

Sin embargo, nosotros podemos representar cualquier función multivaluada con nuestra estructura. Siempre hay una alternativa estándar de hacer esto, construyendo la función interpoladora en el teorema 2.12.

En este caso hay una mejor alternativa: encontrar una expresión más intuitiva y simple donde Dnr es un activador y $NarXL$ es un activador leve. (ver Motifs 1 y 4 en 2.1.3). Ver figura 4.1.

NarXL	Dnr	NirQ
0	0	0
0	1/2	1/2
0	1	1
1	0	1/2
1	1/2	1
1	1	1

La regla de actualización en [1]: $*$ máx$\{NarXL, Dnr\}$.
Este asterisco está dado a que el valor 1/2 en $(0, 1)$ en la cuarta fila no es obtenido con esta expresión.
Expresión interpoladora:

$$\begin{aligned} f_{NirQ} = {} & \frac{1}{2} \odot neg(NarXL)^2 \odot neg(neg(Dnr) \odot \frac{1}{2})^2 \odot neg(Dnr \odot \frac{1}{2})^2 \\ & \oplus neg(NarXL)^2 \odot Dnr^2 \\ & \oplus \frac{1}{2} \odot NarXL^2 \odot neg(Dnr)^2 \\ & \oplus NarXL^2 \odot neg(neg(Dnr) \odot \frac{1}{2})^2 \odot neg(Dnr \odot \frac{1}{2})^2 \\ & \oplus NarXL^2 \odot Dnr^2. \end{aligned}$$

Expresión más simple: $f_{NirQ} = (NarXL \ominus \frac{1}{2}) \oplus Dnr$.

Figura 4.1: $NirQ$ regla de actualización.
La tabla de la izquierda es la versión multivaluada no suavizada de S3 Tabla en [1].
A la derecha: la aproximación de la regla de actualización mostrada en [1].
La regla de actualización obtenida por la interpolación de la tabla de acuerdo al teorema 2.12; y la expresión obtenida por considerar Dnr como activador y $NarXL$ como un activador débil.
Las últimas dos expresiones coinciden con los valores de la tabla.

Dado que tienen representaciones intuitivas, podemos representar

$$\begin{aligned} f_{NirQ} &= (NarXL \ominus \frac{1}{2}) \oplus Dnr \\ f_{Dnr} &= (Anr \ominus \frac{1}{2}) \oplus (\text{neg}(PmrA) \odot Anr \odot NarXL) \\ f_{NO_2} &= NO_3 \odot nar \end{aligned}$$

Por otro lado, para f_{nar} nos queda:

$$f_{nar} = \left[\left(NarXL \oplus \frac{1}{2}\right)^2 \odot \left(Dnr \oplus \frac{1}{2}\right)^2 \odot \left(NO \oplus \frac{1}{2}\right)^2\right] \oplus \\ \oplus \left[\frac{1}{2} \odot \left(NarXL \oplus \left(Dnr^2 \odot NO^2\right)\right)\right].$$

En nuestra configuración, fijamos el valor de m para todas las funciones en la red. En este modelo, la mayoría de las variables toman tres valores, correspondiendo a la elección de $m = 2$.

Las variables Booleanas pueden ser reducidas a tres. En lugar de asignarles valores arbitrarios en $X_m = \{0, 1/2, 1\}$, por ejemplo, los valores 0 o 1, es más razonable en nuestra configuración considerar los 8 modelos diferentes resultantes en las restantes variables 3-valuadas, para cada uno de los posibles valores de las variables booleanas.

Colocamos los valores para cada parámetro externo booleano (O_2, PO_4, NO_3) y estudiamos el correspondiente punto fijo de cada uno de los ocho posibles sistemas por separado.

Las dos condiciones relevantes para la denitrificación con bajo O_2 son

- La condición perfecta para la denitrificación:

$$O_2 = PO_4 = 0, NO_3 = 1 \tag{4.1.1}$$

- La condición de denitrificación interrumpida por la presencia de PO_4:

$$O_2 = 0, PO_4 = NO_3 = 1 \tag{4.1.2}$$

Hay otras dos condiciones con bajo O_2 que los autores en [1] descartan porque son poco probables en agua dulce: $O_2 = PO_4 = NO_3 = 0$, y $O_2 = NO_3 = 0, PO_4 = 1$.

De todas maneras computamos los puntos fijos en nuestra configuración dado que son fáciles de encontrar, y los mostramos en la tabla 4.1. También abordamos las condiciones restantes, con alto O_2 (las "condiciones aeróbicas"). Los puntos fijos computados en todos los casos coinciden con los encontrados en [1].

External parameters			Steady State													
O_2	PO_4	NO_3	PhoRB	PhoPQ	PmrA	Anr	NarXL	Dnr	NirQ	nar	nir	nor	nos	NO_2	NO	N_2O	N_2
0	0	0	1	1	0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
0	1	0	0	0	1	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Cuadro 4.1: Puntos fijos bajo las condiciones ambientales determinadas por parámetros externos. Las primeras dos condiciones son descartadas en [1] por no ser biológicamente relevante. Los últimos cuatro casos (las "condiciones aeróbicas") coinciden con [1, Fig. 2].

Construimos el sistema $\odot -$ neg del teorema 2.28 para los 2 casos de interés, agregando 20 nuevas variables. Aplicamos las reducciones de 3.2 y obtenemos, en cada caso, un sistema reducido para los puntos fijos.

La condición de denitrificación interrumpida por la presencia de PO_4

Este caso 4.1.2 genera un sistema de tres ecuaciones en tres variables que tiene una solución única y determina el punto fijo:

PhoRB	PhoPQ	PmrA	Anr	NarXL	Dnr	NirQ	nar	nir	nor	nos	NO_2	NO	N_2O	N_2
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1

La condición perfecta para la denitrificación

Este caso (4.1.1) da lugar a un sistema de cinco ecuaciones en cinco variables que tiene solución única y determina el punto fijo:

PhoRB	PhoPQ	PmrA	Anr	NarXL	Dnr	NirQ	nar	nir	nor	nos	NO_2	NO	N_2O	N_2
0	0	1	1	1	$\frac{1}{2}$	1	1	1	1	$\frac{1}{2}$	1	1	1	$\frac{1}{2}$

Todos los puntos fijos obtenidos coinciden con los puntos fijos obtenidos en [1]. También implementamos el modelo usando la interpolación de funciones en teorema 2.12 para f_{Dnr}, f_{NirQ}, f_{nar} y f_{NO_2}, y computamos los puntos fijos de cada uno de los ocho sistemas determinados por fijar los valores de los parámetros externos.

Los estados estables en la tabla 4.1 son obtenidos directamente.

Al igual que los 2 casos de interés, utilizamos nuestra herramienta mencionada en esta tesis, y pública en [11], sin considerar ninguna reducción en la red, para los dos casos restantes.

5 Experimentación: El espacio de las funciones $\odot-$neg

Un tema interesante a recorrer es la exploración del espacio de funciones $\odot-$neg. Dado que las funciones $\odot-$ neg son un subconjunto del conjunto de funciones de $F : X_m^n \to X_m^n$ y serán menos que en la proposición 3.1, surgen muchas preguntas: cuántas funciones $\odot-$ neg efectivamente distintas hay? Cuántas funciones $\odot-$ neg con puntos fijos hay? Nuestro trabajo enumera los puntos fijos. Hay alguna forma de solo contarlos que sea más eficiente, o de predecir la proporción de funciones $\odot-$neg con al menos un punto fijo?

En este capítulo vamos a mostrar un breve análisis, en el que describimos 2 métodos de exploración del espacio de funciones junto con su implementación, y algunas tablas mostrando cantidades de funciones con ciertos parámetros fijos.

Sobre el final de este capítulo mostraremos en el Teorema 5.15 la cantidad exacta de funciones $\odot-$ neg para cada elección de m y n.

5.1. Generación de funciones

5.1.1. Aleatoriamente

La función `random_function_generator(n, m, cant_variables_por_fila)` busca construir al azar una función de la forma descripta en el Teorema 3.6.

En el input de dicha función

- n es la cantidad de nodos de nuestra red biológica, en este caso la cantidad de f_i que tiene F.
- m es la cantidad de posibles estados que puede tener cada nodo.
- $cant\_variables\_por\_fila$ limita la cantidad de variables no nulas relacionadas a cada l_i.

Esta función devuelve una matriz de n filas y $n+1$ columnas, donde la i-ésima fila representa a l_i como en (3.3.2), quedándose sólo con los coeficientes. Es decir, $[p_{i1}, p_{i2}, ..., p_{in}, c_i^*]$

Notar que, si bien estamos eligiendo parámetros aleatorios, los elegiremos entre $-m$ y m, porque cualquier parámetro de valor absoluto mayor a m será equivalente a utilizar m (o $-m$), como vimos en la Observación 2.13, ítem (i).

Entonces, se eligen los $p_{ij} \in [-m, m] \cap \mathbb{Z}$, usando `random.randint(low, high=None, size=None, dtype=int)` de la librería numpy, que devuelve enteros al azar con distribución uniforme. Además, pedimos que en cada fila haya a lo sumo `cant_variables_por_fila` no nulas.

```
16 def random_function_generator(n, m,
       cant_variables_por_fila):
17     fun = []
18     for i in range(n):
19         c_i = np.random.randint(m + 1) / m
20         xjs = set(np.random.randint(low=0, high=n, size
       =cant_variables_por_fila))
21         l_i_for_xjs = np.random.randint(low=-m, high=m,
        size=len(xjs)).tolist()
22         for xj in l_i_for_xjs:
23             if xj > 0:
24                 c_i -= xj
25         l_i = []
26         j=0
27         for i in range(n):
28             if i in xjs:
29                 l_i.append(l_i_for_xjs[j])
30                 j+=1
31             else:
32                 l_i.append(0)
33 
34         l_i.append(c_i)
35         fun.append(l_i)
36     return fun
```

5.1.2. Exhaustivamente

La función `exhaustive_l_i_generator(n,m)` busca construir todas las posibles l_i para un n y m dados.

```
12 def exhaustive_l_i_generator(n, m):
13     if n == 0:
14         return [[a] for a in range(m + 1)]
15     else:
16         return [x + [a] for x in
       exhaustive_l_i_generator(n - 1, m) for a in range(-m
       , m + 1)]
17 
18 
19 def fix_independent_terms(functions, n, m):
```

```
20      for f in functions:
21          add = 0
22          for j in range(n):
23              if f[j] > 0:
24                  add += f[j]
25          f[n] -= add * m
26      return functions
27
28
29  def points_generator(n, m):
30      if n == 0:
31          return []
32      if n == 1:
33          return [[a] for a in range(m + 1)]
34      else:
35          return [x + [a] for x in points_generator(n -
    1, m) for a in range(m + 1)]
```

5.1.3. Cantidad de funciones total para distintos n y m

Corriendo el programa generador de funciones exhaustivo para $n = m = 2$ y para $m = 3, n = 2$ y, posteriormente, nuestro programa que calcula puntos fijos de funciones $\odot -$ neg, obtuvimos como resultado listas de funciones con descripción de sus puntos fijos, y resumimos los resultados en las siguientes tablas:

m=2 y n=2	
Cantidad de puntos fijos	**Cantidad de funciones**
0	318
1	589
2	186
3	92
4	25
5	12
6	2
7	0
8	0
9	1
Total de funciones	**1225**
Funciones con al menos un punto fijo	**907**

Cuadro 5.1: Estadísticas sobre las funciones $\odot -$ neg con $m = n = 2$. La lista completa puede explorarse en [31]

m=3 y n=2	
Cantidad de puntos fijos	**Cantidad de funciones**
0	1885
1	3729
2	868
3	214
4	250
5	84
6	0
7	21
8	4
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	1
Total de funciones	**7056**
Funciones con al menos un punto fijo	**5171**

Cuadro 5.2: Estadísticas sobre las funciones $\odot - \text{neg}$ con $m = 3, n = 2$. La lista completa puede explorarse en [32]

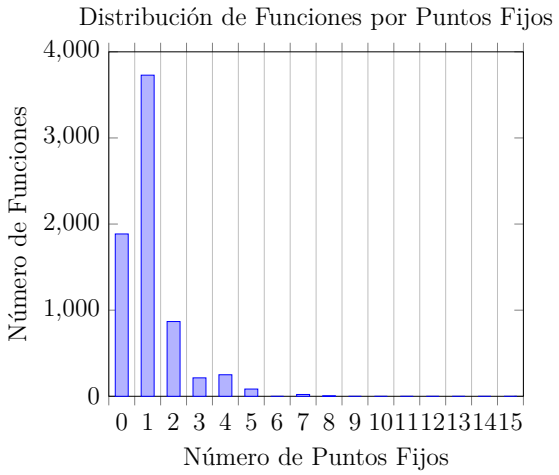


Figura 5.1: Histograma sobre las funciones $\odot - \text{neg}$ con $m = 3$, $n = 2$.

5.2. Análisis sobre la cantidad de funciones $\odot -$ neg en general

5.2.1. Cantidad de funciones $\odot -$ neg

Proposición 5.1. *La cantidad de funciones* $\odot -$ neg *para* m *y* n *es al menos* $(m+1)^{(m+1)^{n'}}$, *donde* $n' \in \mathbb{N}$ *es tal que* $n \geq (2n'+1)(m+1)^{n'} + n' - 1$

Demostración. Una cota inferior que podemos inferir nos la da nuestro interpolador: si observamos en el Teorema 2.12, recordemos que nuestras funciones $\odot -$ neg tienen la siguiente forma:

$$f(x_1, \ldots, x_n) = \bigoplus_{(i_1, i_2, \ldots, i_n) \in X_m^n} f(i_1, i_2, \ldots, i_n) \odot \ell_{i_1}(x_1) \odot \cdots \odot \ell_{i_n}(x_n),$$

- Hay $(m+1)^n$ elementos en X_m^n
- Por lo tanto, esa es la cantidad de sumandos en la interpolación. Usando la observación 2.4 para deshacernos del operador $\oplus$, esto agrega $(m+1)^n - 1$ variables.
- Dentro de cada sumando hay n l_{i_j}. Cada una tiene su g_j y h_j que tienen 1 sumando cada una. En total, al usar 2.4, agregamos $2n$ variables por cada sumando del interpolador.
- Nuevamente, como hay $(m+1)^n$ sumandos en la interpolación, los l_{i_j} agregan $2n \cdot (m+1)^n$ variables en total.
- Es decir que, en el interpolador traducido a una función $\odot -$ neg, hay $n + (m+1)^n - 1 + 2n \cdot (m+1)^n = (2n+1)(m+1)^n + n - 1$ variables.

Entonces, si partimos de funciones $f : X_m^n \to X_m$ distintas, llegamos a funciones $\odot -$ neg $f' : X_m^{(2n+1)(m+1)^n + n - 1}$ distintas.

Por lo tanto, la cantidad de funciones $\odot -$ neg $f' : X_m^{(2n+1)(m+1)^n+n-1} \to X_m$ distintas son al menos tantas como la cantidad de funciones $f : X_m^n \to X_m$ distintas, que sabemos que son $(m+1)^{(m+1)^n}$.

□

Proposición 5.2. *Dados* n *y* m, *la cantidad de funciones* $\odot -$ neg *es a lo sumo* $(m+1)^{(m+1)^n}$

Demostración. Esto vale ya que que esa cota es la cantidad total de funciones $f : X_m^n \to X_m$. El argumento para esta cuenta puede verse en la proposición 3.1 □

Recordemos que

Observación 5.3. Toda función $\odot -$ neg f tiene una representación de la forma máx$(0, \ell(x))$, donde

$$\ell(x) = [a_1, a_2, ..., a_n] * [x_1, ..., x_n]^t + (c - \sum_{i \in A} a_i) \text{ donde } a_i \geq 0 \text{ si } i \in A \text{ y} \quad (5.2.1)$$
$$a_i < 0 \text{ si } i \in B$$

con $a_i \in [-m, m]$.

Demostración. La forma matricial es la representación que venimos trabajando. Por (i) en la Observación 2.13, podemos pedir que ℓ cumpla $-m \leq a_i \leq m$ para todo i, con $1 \leq i \leq n$. □

Proposición 5.4. *Dados n y m, la cantidad de funciones $\odot -$ neg es a lo sumo* $(2m+1)^n m + 1$

Demostración. Si contamos todas las funciones posibles, usando la notación de (5.2.1), vamos a notar que

- Cada a_i tiene $2m+1$ opciones posibles, ya que $a_i \in [-m, m] \cap \mathbb{Z}$
  - $(2m+1)^n$ opciones para la tupla de a_i.
- c tiene $m+1$ opciones, ya que $c \in X_m$
  - Podemos separar $c = 0$, ya que $c = 0 \to l = 0$.

Por lo tanto, la cantidad total de funciones con estas características es $(2m+1)^n m + 1$. □

Observación 5.5.

- Para $n = m = 2$, la proposición 5.2 nos indica que la cantidad de funciones $\odot -$ neg es menor o igual a $3^{3^2} = 19683$.
- En cambio, la proposición 5.4 nos indica que la cantidad de funciones $\odot -$ neg es menor o igual a $(2 \cdot 2 + 1)^2 \cdot 2 + 1 = 51$.
- En 5.1.3 vimos que la cantidad de funciones $\odot -$ neg en dicho caso es 35.

Buscaremos una cota más ajustada.
Recordemos la observación (ii). Con esto se puede generalizar a:

Proposición 5.6.

$$\bigodot_{i \in A} x_i^{p_i} \bigodot_{j \in B} neg(x_j)^{q_j} \bigodot \frac{1}{m} = \bigodot x_i \bigodot neg(x_j) \bigodot \frac{1}{m}$$

Esta observación se puede extender de varias formas:

Lema 5.7. *Si $p \geq c$, entonces*

$$x^p \odot \frac{c}{m} = x^c \odot \frac{c}{m}$$

Demostración. Si $x = 1$, es trivial que la función vale $\frac{c}{m}$. Si $x = 0$ la función vale 0. Supongamos $x \in X_m \backslash \{0, 1\}$.

Entonces la función va a valer 0:

$$\begin{aligned} x^p \odot \frac{c}{m} &\leq 0 \\ p(x-1) + \frac{c}{m} &\leq 0 \\ p &\geq -\frac{c}{m}\frac{1}{(x-1)} \\ p &\geq \frac{c}{m}\frac{1}{(1-x)}. \end{aligned}$$

Como $x \in X_m \backslash \{0, 1\}$ entonces $x = \frac{c'}{m}$ con $1 \leq c' \leq m-1$. Entonces

$$\begin{aligned} p &\geq \frac{c}{m}\frac{1}{(1-x)} \\ &= \frac{c}{m}\frac{1}{(1-\frac{c'}{m})} \\ &= \frac{c}{m}\frac{m}{(m-c')} \\ &= \frac{c}{m-c'}. \end{aligned}$$

Es decir que, en particular, sucede si $p \geq c \geq \frac{c}{m-c'}$. □

Observación 5.8. La cota en el exponente en el Lema 5.7 es tan ajustada como puede ser, pues puede verse que, para $p = c - 1$, $x = \frac{m-1}{m}$ daría un contraejemplo.

Observación 5.9. Si $p \geq c$, entonces para todo y vale que

$$neg(y)^p \odot \frac{c}{m} = neg(y)^c \odot \frac{c}{m},$$

ya que basta tomar $x = neg(y)$ en el Lema 5.7.

Esto quiere decir que

Observación 5.10.

$$\bigodot_{i \in A} x_i^{p_i} \bigodot_{j \in B} neg(x_j)^{q_j} \bigodot \frac{c}{m} = \bigodot x_i^{min(p_i, c)} \bigodot neg(x_j)^{min(p_i, c)} \bigodot \frac{c}{m}$$

Proposición 5.11. *Dados n y m, la cantidad de funciones $\odot -$ neg es a lo sumo $\sum_{c=0}^{m} (2c+1)^n$.*

Demostración. Usando la notación matricial (5.2.1) y la observación 5.10 nos queda que para para c, con $0 \leq c \leq m$,

$$\#\{l_c(x) = \bigodot_{i\in A} x_i^{p_i} \bigodot_{j\in B} neg(x_j)^{q_j} \bigodot \frac{c}{m}\} = (2c+1)^n$$

Por lo tanto

$$\#\{l\} = \sum_{c=0}^{m} \#\{l_c(x)\} = \sum_{c=0}^{m} (2c+1)^n$$

por lo que

$$\#\{\text{funciones} \odot -\text{neg}\} \leq \#\{l\} \leq \sum_{c=0}^{m} (2c+1)^n \tag{5.2.2}$$

□

Observación 5.12.

- Para $n = m = 2$, esto nos indica que la cantidad de funciones $\odot-$neg es menor o igual a $\sum_{c=0}^{2}(2c+1)^2 = 1^2+3^2+5^2 = 35$ ahora coincide con la cantidad de funciones $\odot -$ neg calculadas exhaustivamente en 5.1.3.
- Para $n = 2$ y $m = 3$, esto nos indica que la cantidad de funciones $\odot -$ neg es menor o igual a $\sum_{c=0}^{2}(2c+1)^2 = 1^2+3^2+5^2+49 = 84$ ahora coincide con la cantidad de funciones $\odot -$ neg calculadas exhaustivamente en 5.1.3.

Proposición 5.13. *Dados n, m, $p_1 \leq c_1, p_2 \leq c_2$,*
$f_1 : X_m^n \to X_m, f_1(x) = x^{p_1} \odot \frac{c_1}{m}$, $f_2 : X_m^n \to X_m, f_2(x) = x^{p_2} \odot \frac{c_2}{m}$.
Si $f_1 = f_2$, entonces $c_1 = c_2$ y $p_1 = p_2$.

Demostración.
Primero veamos que $c_1 = c_2$. Con $f \in \{f_1, f_2\}$, $f(1) = 1^p \odot \frac{c}{m} = \frac{c}{m}$. Luego, $\frac{c_1}{m} = f_1(1) = f_2(1) = \frac{c_2}{m} \Leftrightarrow c_1 = c_2$
Para ver que $p_1 = p_2$, sabiendo que $c = c_1 = c_2$, evaluemos en $\frac{m-1}{m}$ a $f \in \{f_1, f_2\}$:

$$\begin{aligned} f\left(\frac{m-1}{m}\right) &= \text{máx}\left(0, \frac{c}{m} - p\left(1 - \frac{m-1}{m}\right)\right) \\ &= \text{máx}\left(0, \frac{c-p}{m}\right) \\ &= \frac{c-p}{m}, \text{ pues } c \geq p \end{aligned}$$

Luego, $\frac{c-p_1}{m} = f_1\left(\frac{m-1}{m}\right) = f_2\left(\frac{m-1}{m}\right) = \frac{c-p_2}{m} \Leftrightarrow p_1 = p_2$ □

Proposición 5.14. *Consideremos dos funciones f_1, f_2 de la forma*

$$f_k = \bigodot_{i\in A} x_i^{p_i} \bigodot_{j\in B} neg(x_j)^{q_j} \bigodot \frac{c}{m},$$

donde $p_i \leq c, q_j \leq c \; \forall i, j$. Entonces f_1 y f_2 son iguales si y solo si todos sus exponentes y sus constantes son iguales.

Demostración. Evaluando en $x_i = 1, x_j = 0\ \forall i, j$, $f_k(x) = \frac{c}{m}$, por lo que $f_1 = f_2$ implica que las constantes son iguales.

Si $A \neq \emptyset$, evaluando en $x_i = 1, x_j = 0\ \forall i, j, i \neq l$, tenemos que $f_k(x) = x_l^{p_l} \odot \frac{c}{m}$. Por 5.13, esto significa que $f_1 = f_2$ implica que los exponentes referidos a x_l son iguales. Evaluando en $x_i = 1, x_j = 0\ \forall i, j, j \neq l$, llegamos a una conclusión similar. □

Teorema 5.15. *Dados n y m, la cantidad de funciones $f : X_m^n \to X_m$ que son $\odot -$ neg es $\sum_{c=0}^{m}(2c+1)^n$*

Demostración. Se sigue de lo demostrado en las Proposiciones 5.11, 5.13 y 5.14. □

6 Conclusiones y trabajo futuro

Modelos con tiempo y valores discretos, como las redes booleanas multivaluadas estudiadas acá, son ampliamente aplicables en biología y más allá, como en ingeniería, ciencias de la computación y física. Matemáticamente, son un conjunto simple de funciones sobre cadenas de longitud finita dada sobre un alfabeto finito. Como tal, no hay una estructura matemática que pueda ser usada para clasificarlas o analizarlas. Su dinámica es capturada por grafos dirigidos, exponencial en el número de variables en el modelo o por reglas lógicas encapsuladas en diagramas dirigidos, dependientes de la elección de un orden de las variables. Tradicionalmente, una simulación exhaustiva o un muestreo en el espacio de estados, para modelos grandes, ha sido el único enfoque para caracterizar la dinámica del modelo.

Una estrategia para crear herramientas matemáticas disponibles para este propósito es basarse en la observación de que toda función $f : k^n \longrightarrow k$ puede ser expresada como una función polinómica sobre un cuerpo finito k. Esto permite usar herramientas del álgebra computacional. Por ejemplo, la descomposición primaria de ideales monomiales puede ser usada para encontrar una ingeniería inversa de redes reguladoras de genes a partir de la evolución temporal de observaciones experimentales [25, 29].

Basado en nuestro trabajo conjunto [14] proponemos un enfoque diferente para aplicar herramientas matemáticas al problema del análisis de modelos discretos con cualquier número finito de valores. Usando las operaciones de la lógica multivaluada, conectamos esta clase de modelos a problemas combinatorios y de geometría enumerativa, en particular el cálculo de puntos racionales en polítopos. Esto, combinado con un método de reducción de modelos generalmente resulta en modelos sustancialmente mas chicos con estados de equilibrio en correspondencia uno a uno con los estados de equilibrio del modelo original. Los pasos de reducción son ad hoc, y no hay teoría detrás que nos permita derivar en un modelo minimal reducido o incluso un resultado acerca de si el orden en el que se aplican las reducciones importa en términos del modelo reducido final. Creemos que en este contexto se pueden descubrir aún más resultados matemáticos interesantes. El próximo paso importante de nuestro trabajo es utilizar estas herramientas presentadas para entender por completo las dinámicas de los modelos.

En general, creemos que las redes multivaluadas y modelos relacionados son ricos y fascinantes objetos matemáticos que mezclan combinatoria, álgebra y sistemas dinámicos. Estos pueden ser estudiados desde el punto de vista aplicado o por pura razones matemáticas o ambos, como en [15].

Un camino prometedor para trabajo futuro es la implementación del Algoritmo 1, introduciendo la capacidad de usar variables auxiliares y transformar funciones ar-

bitrarias en una función $\odot - neg$.

También nos quedan preguntas interesantes sobre el espacio de funciones $\odot -$neg, como las nombradas en el capítulo 5:

- ¿ Cuántas funciones $\odot -$ neg con al menos un punto fijo hay, dados n y m?
- ¿ Se puede predecir, quizás en algún límite, la proporción de funciones con al menos un punto fijo?
- ¿ Cómo se pueden explicar a priori los resultados de las tablas 5.1.3 y 5.1.3?
- ¿ Hay alguna forma de solo contar los puntos fijos de una función que sea más eficiente?

7 Bibliografía

[1] Arat S., Bullerjahn G.S., Laubenbacher R. (2015) *A network biology approach to denitrification in Pseudomonas aeruginosa.* PLoS One. 10(2):e0118235.

[2] Bardet M., Faugère J.C., Salvy B., Spaenlehauer P.J. (2013) *On the complexity of solving quadratic boolean systems.* J. Complex., 29(1), 53–75.

[3] Barvinok A. (1993). *A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed.* In: 34th Annual Symposium on Foundations of Computer Science, 566–572. IEEE.

[4] Barvinok A. (2008) *Integer Points in Polyhedra.* European Mathematical Society Publishing House, Seminar for Applied Mathematics

[5] Brion M. (1988) *Points entiers dans les polyèdres convexes.* Ann. Sci. École Norm. Sup. (4)21, no.4, 653–663.

[6] Chifman J., Arat S., Deng Z., Lemler E., Pino J. C., Harris L. A., Kochen M., Lopez C. F., Akman S.A., Torti F.M., Torti S.V., Laubenbacher R. (2017) *Activated Oncogenic Pathway Modifies Iron Network in Breast Epithelial Cells: A Dynamic Modeling Perspective.* PLoS Comput. Biol. 13(2): e1005352.

[7] Cignoli R. L. O., D'Ottaviano I. M. L., Mundici D. (2000) *Algebraic foundations of many-valued reasoning.* Trends in Logic—Studia Logica Library, Vol. 7, Kluwer Academic Publishers, Dordrecht.

[8] De Loera J. A., Hemmecke R., Tauzer J., Yoshida R. (2004) *Effective Lattice Point Counting in Rational Convex Polytopes.* J. Symb. Comput. 38(4), 1273–1302.

[9] Epstein G. (1960) *The lattice theory of Post algebras.* Trans. Am. Math. Soc. 95(2), 300–317.

[10] Free software *Barvinok*, by Verdoolaege S. and collaborators. Available at: https://barvinok.sourceforge.io/.

[11] Free software *Fixed points of MV networks*, by Galarza Rial A., available at https://github.com/ayegari89/multivalued_fixed_points.

[12] Free software *Latte*, by de Loera J., Köppe M. and collaborators, available at: https://www.math.ucdavis.edu/~latte/.

[13] Free software SageMath, available at: https://www.sagemath.org/.

[14] J. García Galofre, M. Pérez Millán, A. Galarza Rial, R. Laubenbacher, A. Dickenstein (2024) *Beyond Boolean networks, a multi-valued approach* arXiv preprint arXiv:2404.16760

[15] Kadelka C., Wheeler M., Veliz-Cuba A., Murrugarra D., Laubenbacher R. (2023) *Modularity of biological systems: a link between structure and function.* J. R. Soc. Interface 20: 20230505.

[16] Kauffman, S. (1969) *Homeostasis and Differentiation in Random Genetic Control Networks.* Nature 224, 177–178.

[17] Li, Z., Cheng, D. (2010) *Algebraic approach to dynamics of multivalued networks.* International Journal of Bifurcation and Chaos, 20(03), 561–582.

[18] Naldi, A., Thieffry, D., Chaouiya, C. (2007) *Decision diagrams for the representation and analysis of logical models of genetic networks.* Computational methods in systems biology, 233—247. Lecture Notes in Comput. Sci., 4695 Lect. Notes in Bioinform., Springer, Berlin.

[19] Stanley, R., (1979) Enumerative Combinatorics, vol. I., Cambridge University Press.

[20] Motoyosi Sugita, (1963) *Functional analysis of chemical systems in vivo using a logical circuit equivalent. II. The idea of a molecular automaton*, J.Theor. Biol. 4, 179–192.

[21] Thomas R., (1973) *Boolean formalisation of genetic control circuits.* J. Theor. Biol. 42, 565–583.

[22] Thomas R. (1991) *Regulatory networks seen as asynchronous automata: a logical description*, J. Theor. Biol. 153, 1–23.

[23] Thomas R., d'Ari R. (1990) *Biological feedback.* CRC press.

[24] Tonello E. (2019) *On the conversion of multivalued to Boolean dynamics.* Discrete Appl. Math. 259, 193–204.

[25] Veliz-Cuba A. (2012) *An Algebraic Approach to Reverse Engineering Finite Dynamical Systems Arising from Biology.* SIAM J. Appl. Dyn. Syst. 11(1), 31–48.

[26] Veliz-Cuba A., Jarrah A., Laubenbacher R. (2010) *Polynomial algebra of discrete models in systems biology.* Bioinformatics, 26, 1637–1643.

[27] Veliz-Cuba A, Aguilar B., Laubenbacher R. (2015) *Dimension reduction of AND-NOT network model.* Electronic Notes in Theoretical Computer Science 316 , 83–95.

[28] Veliz-Cuba A., Buschur K., Hamershock R., Kniss A., Wolff E., Laubenbacher, R. (2013) *AND-NOT logic framework for steady state analysis of Boolean network models.* Appl. Math. Inf. Sci. 7(4): 1263–1274.

[29] Veliz-Cuba A, Newsome-Slade V., Dimitrova E. (2024) *A Unified Approach to Reverse Engineering and Data Selection for Unique Network Identification.* SIAM J. Appl. Dyn. Syst. 23(1), 592–615.

[30] Verdoolaege S., Woods K., Bruynooghe M., Cools R. (2005) *Computation and Manipulation of Enumerators of Integer Projections of Parametric Polytopes.* Report CW 392, March. Katholieke Universiteit Leuven Department of Computer Science.

[31] A. Galarza Rial (2024) *Lista de funciones* $\odot -$ neg *para* $n = m = 2$ https://docs.google.com/spreadsheets/d/1Vtgem_WZNhC15Arwzi-BfIOW51NOfLrcgLwwzxcJJ-I/edit?usp=sharing.

[32] A. Galarza Rial (2024) *Lista de funciones* $\odot -$ neg *para* $n = 2, m = 3$ https://docs.google.com/spreadsheets/d/12GijRe8x6-3wn7I_3Gb2514IKKsNm1NzcD_2iSrpWP4/edit?usp=sharing.

[33] S. Verdoolaege (2024) barvinok: User guide https://barvinok.sourceforge.io/barvinok.pdf. Descarga del programa: https://barvinok.sourceforge.io/.

[34] Mammalian Cell Cycle (1607) https://cellcollective.org//#module/1607:1/mammalian-cell-cycle/1.

[35] Cell Collective www.cellcolective.org.