Computación Bioinspirada: Spiking Neural P-systems

Mario Chacón Falcón

Computación Bioinspirada: Spiking Neural P-systems

Mario Chacón Falcón

Memoria presentada como parte de los requisitos para la evaluación de la asignatura de Computación Bioinspirada, del master universitario en lógica, computación e inteligencia artificial.

Índice general

1. Introducción **3**

1.1. Inspiración biológica 3

1.2. Aplicando la inspiración biológica 7

1.3. Prerrequisitos 10

2. Formalización de modelos **15**

2.1. Máquinas de registro 15

2.2. Sistemas P de neuronas spiking 16

2.2.1. **Formalización** 16

2.2.2. **Trenes de spikes** 19

2.2.3. **Ejemplo** 21

3. Resultados de universalidad **27**

3.1. Universalidad para sistemas SN P con spikes no acotados 27

3.2. Universalidad para sistemas SN P con spikes acotados 46

4. Apéndice **63**

Resumen

Las redes neuronales son actuales protagonistas en los métodos de computación. Las neuronas se caracterizan por su cantidad inmensa de conexiones, que, junto con su capacidad de actuar en paralelo, nos permiten procesar toda la información necesaria. Inspirados en estos impulsos eléctricos, se llega al modelo de computación presentado en la memoria, los sistemas P a modo de neuronas spiking (sistemas SN P).

Los sistemas SN P que se presentan son de dos tipos. En uno, no se restringe la cantidad de "potencial" que puede contener una neurona, resultando en la capacidad computacional de las máquinas de Turing (equivalentes a NRE). Sin embargo, si limitamos el contenido de las neuronas, se llega a que la potencia de los sistemas baja drásticamente, pudiendo computar sólo conjuntos semilineales de números naturales ($NREG$).

Se ha desarrollado una memoria que es mayoritariamente autocontenida, recogiendo todos los resultados de universalidad probados en los artículos [2],[1]. Además de esto, se trata de completar algún resultado. Son de especial interés dos demostraciones nuevas, que se han dado para tratar de completar los resultados dados en el artículo, que son el Lema 3.1 y el Teorema 3.10.

La memoria comienza con un primer capítulo ilustrando la relación de nuestro modelo con las neuronas biológicas, dando también algunos conceptos necesarios sobre teoría de autómatas y lenguajes. Tras esto, en el capítulo dos, se da la formalización de los dos modelos de computación utilizados. Por último, en el capítulo tres, se demuestran todos los resultados de universalidad ya mencionados en un párrafo anterior (equivalencias con NRE y $NREG$).

1 Introducción

Actualmente, en 2024, las neuronas son las protagonistas de los métodos de computación (redes neuronales). Estos modelos, a pesar de estar inspirados en las neuronas biológicas, presentan ciertos cambios. En esta memoria se presenta la formalización de un modelo de computación más fiel a las propias neuronas, las redes neuronales spiking. Antes de presentar el modelo, se expone la inspiración biológica que hay detrás, repasando el funcionamiento básico de una neurona.

1.1 Inspiración biológica

Una neurona spiking típica puede ser dividida en tres partes distintas, llamadas dendritas, soma y axón. Sin entrar en mucho detalle, las dendritas juegan el papel de "recibir la señal", recibe spikes de otras neuronas y los pasa al soma. En el soma es "donde se procesa la información". Aquí es donde se realizan ciertas operaciones de procesamiento. Si la "entrada" (potencial) recibida en el soma sobrepasa cierto umbral, entonces una "salida" (se dispara un spike) es generada. El encargado de repartir la señal a otras neuronas es el axón, tratado como "dispositivo de salida".

A la conexión entre dos neuronas se le denomina sinapsis. Normalmente nos referiremos a la neurona que manda la salida como la neurona presináptica y a la que la recibe como la postsináptica. Las sinapsis entre dos neuronas se pueden considerar instantáneas o con un cierto retraso. En la figura 1.1 se puede ver un dibujo de una neurona.

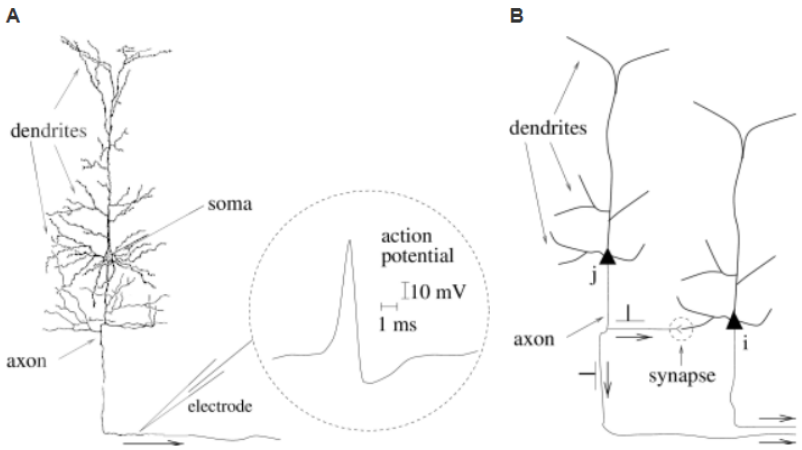


Figura 1.1: Dibujo de una neurona hecho por Ramón y Cajal

La clave para la inspiración de nuestro modelo de computación vendrá dada por estas spikes que consisten en impulsos eléctricos. Estos impulsos eléctricos no cambian conforme el potencial de acción se propaga por el axón. Por esto, los distintos spikes de una neurona serán parecidos, y, por tanto, la información esencial vendrá dada por el número de spikes, así como el tiempo en el que se producen, no por el potencial de acción (la energía que pasa por el axón) de la neurona.

A una cadena de spikes se le denomina **tren de spikes**. Normalmente, los spikes de un tren de spikes están separados. La distancia mínima entre dos spikes define el periodo refractario de la neurona. A continuación se muestran gráficas de una simulación de un tren de spikes de ciertas redes neuronales.

Ejemplo 1.1. Se presenta como primer ejemplo una única neurona que recibe un potencial de acción v que varía en el tiempo según la siguiente ecuación diferencial:

$$\frac{dv}{dt} = \frac{I - v}{\tau}$$

donde $\tau = 10$, $I = 1$. Esta neurona tendrá un umbral de $0.8v$ y cada vez que envíe un spike su potencial de membrana se reseteará a 0. En la figura 1.2 se puede ver una gráfica de la variación del potencial con el tiempo.

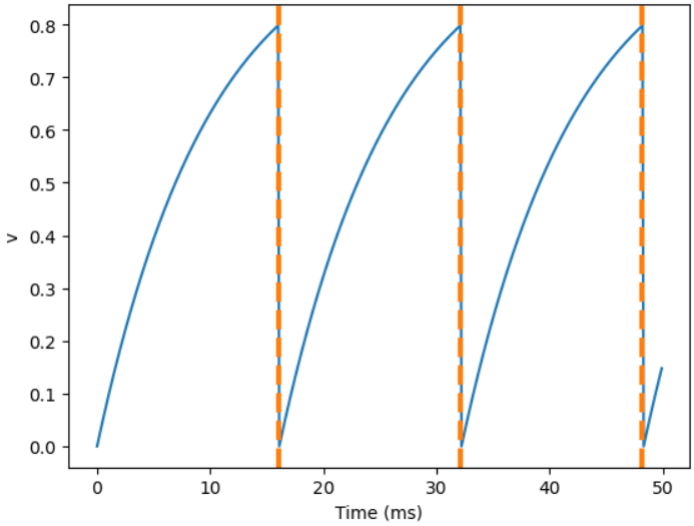


Figura 1.2: Variación del potencial de membrana de la neurona del ejemplo 1.1 cuando varía el tiempo sin periodo refractario.

Si además le añadimos a los spikes un periodo refractario de $5ms$ nos queda la siguiente gráfica:

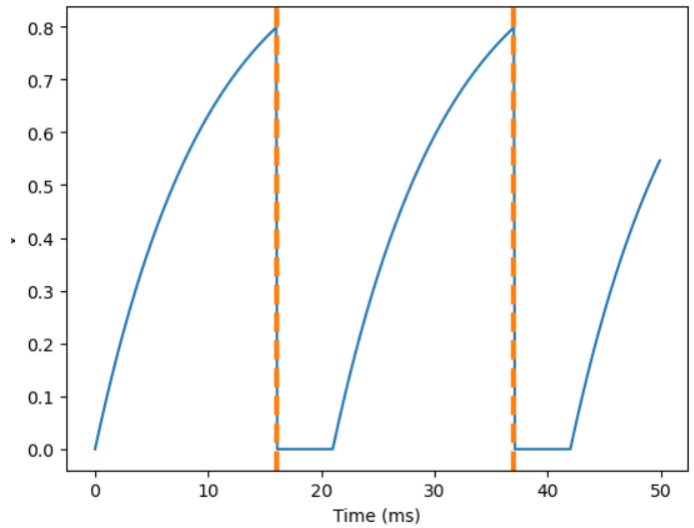


Figura 1.3: Variación del potencial de membrana de la neurona del ejemplo 1.1 cuando varía el tiempo con periodo refractario.

Ejemplo 1.2. Veamos ahora otro ejemplo con más de una neurona. Consideremos el siguiente ejemplo:

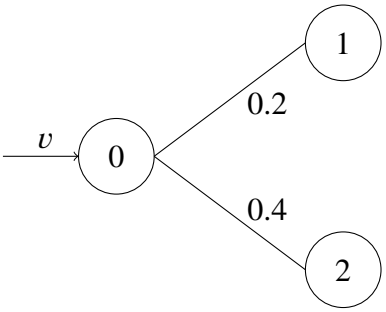


Figura 1.4: Red neuronal con 3 neuronas

A Donde se tienen 3 neuronas, de las cuales la neurona número 0 se comporta como la neurona del ejemplo 1, recibiendo el mismo potencial de acción v dado por la ecuación diferencial. Esta neurona 0 está conectada por unas sinapsis a la neurona 1 y la neurona 2. Cada vez que la neurona 0 mande un spike, las neuronas 1 y 2 recibirán un potencial de acuerdo a sus pesos (0.2 y 0.4) respectivamente. El spike se transmitirá sin retraso. Las neuronas 1 y 2 tienen también un umbral de 0.8. Al igual que la neurona 0, cuando lo alcancen emitirán un spike. Esta descripción se puede ver en la siguiente gráfica.

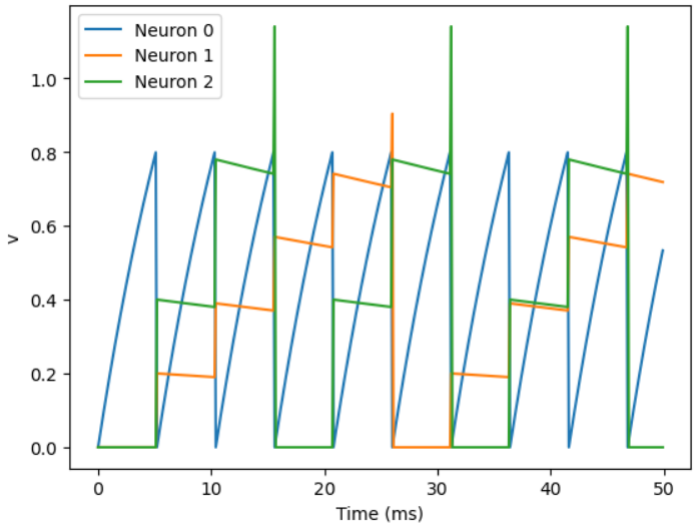


Figura 1.5: Variación del potencial de membrana de las distintas neuronas de la red neuronal de la figura 1.4 sin retraso en la sinápsis.

Si ahora introducimos un retraso en las sinápsis de la neurona 0 a la 1 y a la 2, de $2ms$ y $4ms$ respectivamente, se produce la siguiente gráfica:

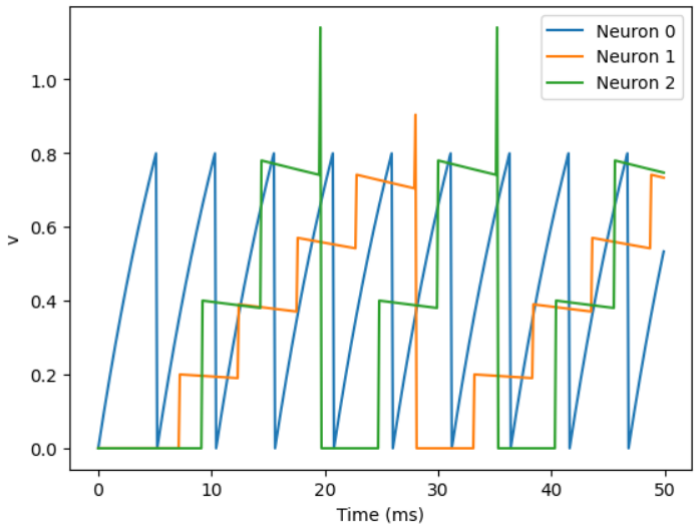


Figura 1.6: Variación del potencial de membrana de las distintas neuronas de la red neuronal de la figura 1.4 con retraso en la sinápsis.

1.2 Aplicando la inspiración biológica

En el apartado anterior vimos un esquema de cómo se comportan las neuronas spiking. En esta sección daremos un esquema de nuestro modelo de computación, destacando sus parecidos a las neuronas biológicas y comparándolo con otros modelos de computación bioinspirados ya estudiados.

El modelo de sistema P spiking (sistema SN P) se representa como un grafo dirigido con nodos representando neuronas y aristas representando sinapsis (como en la figura 1.4). Los spikes mandados por cada una de las neuronas se representarán como un símbolo específico (a).

Como se ha visto en la inspiración biológica, cada neurona puede recibir y mandar potenciales de acción en tiempos distintos. Para representar cuándo y cómo cada una de las neuronas de nuestro sistema SN P manda spikes, cada neurona σ tendrá sus propias reglas para mandar spikes (reglas de disparo) o bien para consumir potencial de acción, consumir energía. A estas últimas se las denomina reglas de olvido. Además, cada regla r tendrá asociado un conjunto S_r, de forma que la regla r se activará solo si la energía de su neurona asociada σ está en el conjunto S_r. Mimetizando el funcionamiento de retraso de las conexiones que se observa en la biología, dada una

regla r de disparo de σ ya activa, puede o bien dispararse inmediatamente (y repartirse por todas las sinapsis que están conectadas a σ) o bien dispararse con retraso. Esto se representará asociando a cada regla r un número de retraso d_r. Si se tiene que $d_r = 0$ entonces el spike se disparará justo en el momento de activación de la regla, t. Ahora bien, si $d_r > 0$, entonces la neurona σ disparará d_r momentos después de que se activara. Además de esto, inspirado en el periodo refractario de una neurona biológica, la neurona σ se bloqueará durante el intervalo $t, t+1, \ldots, t+(d_r-1)$, desbloqueándose en el momento $t + d_r$ (cuando se dispara el spike). Cuando σ está bloqueada no podrá recibir spikes de otras neuronas, es decir, si otra neurona manda un spike a σ, este spike es "desperdiciado" y no entra en σ. Es importante destacar que no hay retraso entre sinapsis. Si σ dispara un spike en el momento t a la neurona σ' esta recibirá este spike en el mismo instante t (si no está bloqueada).

Además, inspirado también en la biología de la neurona, cada regla r tendrá un número fijo c_r que representará la energía que consume la neurona al disparar. Se debe cumplir que $c_r \leq z$ para todo $z \in S_r$. Así, dada una neurona σ con una única regla r sin retraso, un contenido n de spikes(energía) y $n \in S_r$, entonces se dispara la regla r y consume c_r spikes ("unidades de energía") dejando así a σ con $n - c_r$ spikes.

El tiempo para todas las neuronas es determinado por un "reloj global", que hace que el tiempo pasado sea el mismo para todas las neuronas. Las reglas descritas en los párrafos anteriores informalmente tienen la siguiente forma:

1. **Reglas de disparo** $(E/a^c \rightarrow a; d)$: Aquí E representa una expresión regular sobre $\{a\}$. Si una neurona σ contiene esta regla, entonces E nos da el conjunto S_r de activación, c denota el consumo de la regla y d especifica el retraso de la regla.
2. **Reglas de olvido**$(a^s \rightarrow \lambda)$: Esta regla se activa en el instante t si el contenido en el momento t de la neurona es s. Cuando este tipo de regla es aplicada, el efecto es consumir s spikes de la propia neurona, referido como se mencionó antes "como olvido".

Una vez se ha visto cómo funciona cada neurona queda plantearse cómo sabemos cuantos spikes (potencial) hay en cada neurona en cada instante de tiempo. En la configuración inicial una neurona σ guardará la cantidad de ptencial que posee al principio (cierta cantidad de a). En un tiempo determinado, la cantidad de potencial en una neurona σ estará determinada por la cantidad de potencial inicial que poseía σ y y el historial de funcionamiento de la neurona (los spikes que ha recibido de

otras neuronas, los spikes que envía y los spikes que ha olvidado). De manera más específica:

Dada una neurona σ y un tiempo t, se denota por $con_\sigma(0)$ a los spikes iniciales de la neurona σ, por $con_\sigma(t)$ al número de spikes que hay en la neurona σ en el instante t y por $get_\sigma(t)$ a los spikes que llegan a σ en el instante t. Se tiene que:

Si σ está bloqueada en el instante t, entonces:

$$con_\sigma(t+1) = con_\sigma(t)$$

Si σ no está bloqueada y ninguna regla es activada con $con_\sigma(t)$, entonces:

$$con_\sigma(t+1) = con_\sigma(t) + get_\sigma(t)$$

Si σ no está bloqueada y al menos una regla de σ esta activa, entonces, de manera no determinística, exactamente una regla r de σ se escoje. Si r es una regla de disparo, $E/a^c \rightarrow a; d$ entonces:

$$con_\sigma(t+1) = con_\sigma(t) - c + get_\sigma(t) \quad \text{si } d = 0$$

$$con_\sigma(t+1) = con_\sigma(t) - c \quad \text{si } d > 0$$

Además en el segundo caso se tiene que σ manda spikes a través de todas las sinapsis en el instante $t + d$.

Si r es una regla de olvido, $a^s \rightarrow \lambda$, entonces:

$$con_\sigma(t+1) = con_\sigma(t) - s + get_\sigma(t)$$

Una vez se ha presentado un esquema de cómo funcionan los sistemas P spiking, uno puede señalar parecidos entre estos y los sistemas P a modo de tejidos.

Las neuronas de los sistemas P spiking funcionan de forma similar a células de sistemas P a modo de tejidos, conteniendo reglas de evolución (de disparo y de olvido) y multiconjuntos de moléculas donde solo hay una molécula, a, en todo el sistema.

Sin embargo, hay una gran diferencia que distingue los sistemas P spiking de los sistemas P a modo de tejidos. Mientras que los sistemas P a modo de tejido actúan de manera paralela y maximal, cada célula de los sistemas P spiking solo puede aplicar 1 regla a la vez (los sistemas SN P actúan de manera secuencial y paralela). Tiene

sentido por tanto plantearse los sistemas SN P como un nuevo modelo de computación y estudiar su universalidad.

Ahora que sabemos que los sistemas SN P presentan diferencias respecto a otros sistemas, cabe preguntarse cuál es la forma de computar con estos. Para acabar la sección se presenta una idea (inspirada en el tren de spikes) de interpretar las computaciones de sistemas SN P, vista en detalle en el capítulo 2 junto con la definición formal de estos sistemas.

Como ya se destacó en la inspiración biológica, los spikes son todos muy parecidos y, por sí solos no dan información. Dado un sistema SN P, se considera una de las neuronas de nuestra red como la neurona de salida, para la cual se recogen los spikes que emite, así como los tiempos en los que lo hace (el tren de spikes de la neurona de salida). De esta forma, una manera de interpretar la salida de un sistema SN P es considerando el tiempo que pasa entre el primer spike y el segundo, tomando este tiempo como la salida de mi sistema SN P. Hay otras muchas maneras de interpretarlas y se verán con detalle más adelante.

En esta memoria se recogen resultados de universalidad probados en los artículos [2] y [1], asociados a todas estas maneras de interpretar la salida de mis sistemas SN P. Para ver los resultados, serán necesarios ciertos conceptos técnicos que se expondrán en el siguiente apartado, cuyas definiciones han sido en parte sacadas del material de la asignatura de Ciencias de la Computación, de la facultad de matemáticas de la Universidad de Sevilla.

1.3 Prerrequisitos

Durante el desarrollo de la memoria serán necesarios algunos conceptos básicos sobre teoría de autómatas y lenguajes. Para dejar clara la notación usada en el artículo, se repasan algunos conceptos básicos.

Dado un alfabeto de trabajo Σ, Σ^* denota el conjunto de todas las cadenas finitas de símbolos de Σ. La cadena vacía se representará por λ, y el conjunto de todas las cadenas de símbolos (sin la cadena vacía) se denotará por Σ^+. Si $\Sigma = \{a\}$, entonces se escribe a^* en vez de $\{a\}^*$ y a^+ en vez de $\{a\}^+$. La longitud de una cadena $x \in \Sigma^*$ se denota por $|x|$.

La memoria consiste en demostrar resultados de universalidad de los sistemas SN

P. Se usará la siguiente notación:

- Se denota por $\boldsymbol{RE}$ a la familia de lenguajes recursivamente enumerables (o cadenas finitas).
- Se denota por $\boldsymbol{NRE}$ a la familia de conjuntos de números naturales Turing computables (familia de conjuntos de longitudes de palabras de lenguajes recursivamente enumerables).
- Se denota por $\boldsymbol{NREG}$ a la familia de conjuntos semilineales de números naturales (familia de conjuntos de longitudes de palabras de lenguajes regulares).

Dentro de las familias anteriores se verán equivalencias de sistemas SN P con $\boldsymbol{NRE}$ y $\boldsymbol{NREG}$ (en función de si acotamos o no el número de spikes).

Un conjunto de números naturales se dice que es semilineal si es unión de un conjunto finito de números naturales con un número finito de progresiones aritméticas.

A la hora de formalizar las reglas de los sistemas SN P, serán necesarios los siguientes conceptos sobre expresiones regulares y lenguajes generados por una expresión regular.

Definición 1.1. *El conjunto de expresiones regulares sobre un alfabeto Σ se define como el menor conjunto cumpliendo:*

- *$\emptyset$ es una expresión regular*
- *λ es una expresión regular*
- *Para cada $a \in \Sigma$, a es una expresión regular.*
- *Si e y d son expresiones regulares entonces $(e + d)$ y $e \cdot d$ también lo son.*
- *Si e es una expresión regular entonces e^* también lo es.*

Definición 1.2. *Dada una expresión regular e sobre Σ el lenguaje generado por e, denotado por $L(e)$, se define recursivamente como sigue:*

1. $L(\emptyset) = \emptyset$
2. *Si* $e = \lambda$ *entonces* $L(e) = \lambda$.
3. *Para cada* $a \in \Sigma$, *si* $e = a$, *entonces* $L(e) = a$.
4. *Si* $e = (e_1 + e_2)$, *entonces* $L(e) = L(e_1) + L(e_2)$
5. *Si* $e = (e_1 \cdot e_2)$, *entonces* $L(e) = L(e_1) \cdot L(e_2)$
6. *Si* $e = d^*$, *entonces* $L(e) = L(d)^*$

Teorema 1.1. *Un lenguaje L sobre Σ es regular si y solo si existe una expresión regular e sobre Σ tal que $L = L(e)$.*

Con esto ya tenemos herramientas suficientes para definir los sistemas SN P. Por último se repasa algo de contenido sobre gramáticas, que nos será de utilidad en una demostración más adelante.

Definición 1.3. *Se define una **producción** (o **regla de escritura**) sobre un alfabeto Σ como un par de palabras sobre Σ, $(g, \overline{g})$.*

La regla $(g, \overline{g})$ se representará por $g \to \overline{g}$.

Si P es una producción $g \to \overline{g}$, y $u, v \in \Sigma^*$ escribiremos $u \Rightarrow_P v$ para expresar que existen $r, s \in \Sigma^*$ tales que $u = rgs$ y $v = r\overline{g}s$.

Definición 1.4. *Se define un **proceso semi-Thue** como un conjunto finito de producciones.*

Si ϕ es un proceso semi-Thue y $u, v \in \Sigma^*$ escribimos:

- *$u \Rightarrow_\Phi v$ si existe $P \in \Phi$ tal que $u \Rightarrow_P v$.*
- *$u \Rightarrow_\Phi^* v$ si existe una sucesión $w_1, \dots, w_n \in \Sigma^*$ tal que $w_1 = u, w_n = v$ y*

$$\text{Para cada } j = 1, \dots, n-1, w_j \Rightarrow_\Phi w_{j+1}$$

(decimos que $w_1, \dots, w_n$ es una derivación de v a partir de u.)

Definición 1.5. *Una **gramática** es un proceso semi-Thue en el que los símbolos del alfabeto se han dividido en dos clases (terminales y no terminales), y existe un símbolo no terminal distinguido que llamamos símbolo inicial.*

Una gramática $\Gamma = (V, T, \Phi, S)$ está determinada por:

- *Un alfabeto $\Sigma = V \cup T$ con $V \cap T = \emptyset$ (V será el conjunto de símbolos no términales y T será el conjunto de símbolos terminales)*
- *$S \in V$ (símbolo inicial).*
- *Un conjunto de reglas de reescritura, Φ.*

Definición 1.6. *Se define el **lenguaje generado** por la gramática $\Gamma = (V, T, \Phi, S)$ como:*

$$L(\Gamma) = \{u \in T^* : S \Rightarrow_\Phi^* u\}$$

Una grámatica $\Gamma = (V, T, \Phi, S)$ es independiente del contexto si para cada producción $u \to v$ de Φ, se cumple $u \in V$.

Definición 1.7. *Una gramática $\Gamma = (V, T, \Phi, S)$ independiente del contexto es **regular** si para cada regla de producción $u \to v$ de Φ, $v \in T^*(V + \lambda)$.*

Acabamos con un teorema de equivalencia entre lenguajes regulares y gramáticas regulares.

Teorema 1.2. *Un lenguaje regular L es regular si y solo si existe una gramática regular Γ que lo genera (es decir, $L = (L(\Gamma))$.*

Para acabar con esta sección, es importante establecer una **convención** habitual que se hace a la hora de comparar dos sistemas generadores (o en modo de aceptación) de números. A la hora de ver que generan/aceptan los mismos números no se tiene en cuenta el cero. Esto se corresponde con una práctica habitual cuando se trabaja con gramáticas y teoría de autómatas, donde la palabra vacía λ es ignorada cuando se compara la capacidad de dos máquinas de generar/aceptar un determinado lenguaje.

2 Formalización de modelos

En esta sección se introducen la definición formal de las máquinas generadoras y de los sistemas SN P.

Como ya se adelantó en la introducción, el objetivo de la memoria es recopilar resultados de universalidad sobre sistemas SN P. Para demostrar estos resultados, será clave una caracterización de NRE por máquinas de registro, y, como consecuencia, se introduce la definición formal de estas (de manera no determinista).

2.1 Máquinas de registro

Definición 2.1. *Se define una* ***máquina de registro*** M *como una tupla* $M = (m, H, l_0, l_h, I)$ *donde:*

- m *es el número de registros.*
- H *es el conjunto de etiquetas.*
- l_0 *es la etiqueta de entrada.*
- l_h *es la etiqueta de parada.*
- I *es el conjunto de instrucciones*

Es necesario destacar que cada etiqueta de H *identifica una y solo una instrucción de* I*. Las instrucciones son de la siguiente forma:*

- l_1 : $(\mathrm{ADD}(r), l_2, l_3)$*: Suma uno al registro* r *y ve a la instrucción* l_2 *o a la instrucción* l_3 *de manera no determinista.*
- l_1 : $(\mathrm{SUB}(r), l_2, l_3)$*: Si el registro* r *es distinto de cero resta uno al registro y ve a la instrucción con etiqueta* l_2*, en otro caso ve a la instrucción con etiqueta* l_3*.*

- l_h : HALT *: La instrucción de parada.*

Una máquina de registro M computa un número n de la siguiente manera:

Se empieza con todos los registros a cero, aplicamos la instrucción con etiqueta l_0 y seguimos las instrucciones siguiendo las normas de la definición. Si se llega a la instrucción de parada, entonces el número n guardado en ese momento en el primer registro se dice que ha sido computado por M. Al conjunto de todos los números computados por M se le denota $N(M)$. Sin pérdida de generalidad se puede asumir que en la configuración de parada todos los registro menos el primero son cero y que el registro de salida no decrece nunca durante la computación.

El siguiente resultado será clave para demostrar los teoremas de la memoria. Para el lector interesado se puede consultar en [3].

Teorema 2.1. *Las máquinas de registros computan todos los conjuntos de números que son Turing computables (y por lo tanto caracterizan NRE).*

2.2 Sistemas P de neuronas spiking

Se presenta en esta sección el modelo protagonista en la memoria, los sistemas P a modo de neuronas spiking, basados en las neuronas spiking expuestas en la introducción. Comenzamos con la definición formal del modelo.

2.2.1 Formalización

Definición 2.2. *Se definen los sistemas P a modo de neuronas spiking (sistema SN P) de grado $m \geq 1$ como una tupla:*

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_0)$$

Donde:

1. $O = \{a\}$ *es el alfabeto unitario (a se le llama spike)*
2. $\sigma_1, \ldots, \sigma_m$ *son neuronas definidas como:*

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m$$

cumpliendo:

- $n_i \geq 0$ *es el número inicial de spikes en* σ_i
- R_i *es un conjunto finito del siguiente tipo de reglas:*
  (a) *(regla de disparo)* $E/a^c \to a; d$*, donde* E *es una expresión regular sobre* a*,* $c \geq 1$ *y* $d \geq 0$*.*
  (b) *(regla de olvido)* $a^s \to \lambda$ *para algún* $s \geq 1$*, con la restricción de que para cada regla del tipo (a) de* R_i*, se tiene que* $a^s \notin L(E)$ *(esta restricción hace que no sean aplicables reglas de olvido y de disparo a la vez en la misma neurona en el mismo paso de computación)*

3. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ *con* $(i, i) \notin syn$ *para* $1 \leq i \leq m$ *(sinapsis entre neuronas).*
4. $i_0 \in \{1, 2, \ldots, m\}$ *indica la neurona de salida (*σ_{i_0} *es la neurona de salida).*

Cabe destacar que, como es habitual, se asume que hay un tiempo global, que se aplica a toda la red, haciendo que el sistema P funcione de manera sincronizada (En un mismo instante t todas las neuronas dan un paso de computación a la vez). Veamos ahora informalmente cómo se aplican las distintas reglas. Comenzamos con las reglas del tipo (a), reglas de disparo.

Supongamos que la neurona σ_i contiene k spikes, $a^k \in L(E)$ y $k \geq c$. Entonces, la regla $E/a^c \to a; d$ es aplicable. Aplicar esta regla significa consumir c spikes de σ_i (dejando solo $k-c$) y disparar, produciendo un spike después de d instantes de tiempo. Si $d = 0$ entonces el spike se transmite inmediatamente en el instante t. Ahora bien, si $d \geq 1$ entonces en los instantes de tiempo $(t, \ldots, t+d-1)$ la neurona σ_i permanecerá cerrada (o bloqueada). Esto es, que durante este periodo no puede recibir ningún spike (si otra neurona le manda un spike durante ese periodo este spike es "desperdiciado"). En el paso $t+d$ la neurona σ_i manda un spike y vuelve a estar abierta, pudiendo volver a recibir spikes en ese momento (y por tanto pueden ser usados en el paso $t + d + 1$).

Veamos ahora las reglas de tipo (b), reglas de olvido. Supongamos que la neurona σ_i contiene exactamente s spikes, y la regla $a^s \to \lambda$ está en R_i. Entonces, esta se aplica, consumiendo directamente s spikes de σ_i. (Nótese que, por la condición que se puso en la creación de las reglas de olvido, cuando esta es aplicable ninguna regla de disparo lo es).

Observando la definición uno puede darse cuenta de que es posible que dos reglas de disparos, $E_1/a^{c_1} \to a; d_1$ y $E_2/a^{c_2} \to a; d_2$, sean aplicables a la vez. Basta que pase que $L(E_1) \cap L(E_2) \neq \emptyset$. Es necesario por tanto precisar cómo se aplican las reglas.

En cada instante de tiempo, si una neurona σ_i está activa, es decir, que al menos una

regla es aplicable, entonces siempre debe aplicarse una y solo una regla. En el caso de que exista más de una, la regla que se aplica será escogida de manera no determinista. Nótese que del hecho de que solo se aplique una regla en cada neurona y de que haya un reloj universal para toda la red, se tiene que los sistemas SN P computan de manera secuencial en cada neurona pero de forma paralela entre estas.

Es importante resaltar que las reglas son aplicables en base al contenido total de spikes en la neurona. Por ejemplo, supongamos que tenemos una neurona σ_i con 5 spikes y un conjunto de reglas R_i dado por:

$$R_i = \{aa(aa)^+/a \to a; 0, a^3/a \to a; 0, a^2 \to \lambda\}$$

Definida de esta manera, ninguna de las reglas de R_i es aplicable, ya que $a^5 \notin L((aa)^*)$ y a^5 es distinto de a^3 y de a^2.

Una vez se ha descrito el sistema SN P y cómo funcionan las reglas, queda especificar cómo se describirán las computaciones los sistemas SN P. Dado un sistema SN P, su configuración inicial vendrá dada por el número de spikes $n_1, \dots, n_m$ en cada neurona. A partir de esta idea se define una configuración en un instante t determinado.

Nótese que fijado el sistema SN P, ya conocemos las reglas y las sinapsis del sistema. De esta manera, para saber el estado en el que se encuentra, solo es necesario saber la cantidad de spikes en cada neurona y si están cerradas o abiertas.

Definición 2.3. *Dado un sistema sistema SN P* $\Pi = (\{a\}, \sigma_1, \dots, \sigma_m, syn, i_0)$ *se define el "estado" o* ***configuración*** *del sistema en un instante de tiempo* t *como el estado de cada una de sus neuronas. El "estado" de la neurona* σ_i *vendrá dado por un par* (n_i, t_i)*, donde* n_i *representa el número de spikes en la neurona* i *y* t_i *representa el número de pasos hasta que la neurona* i *esté abierta. Si* $t_i = 0$*, como pasa en el instante inicial, quiere decir que la neurona está abierta, por lo que puede activar reglas y recibir spikes de otras neuronas.*

De esta forma, una computación C *del sistema* Π *se denotaría como:*

$$C = ((n_1, t_1), \dots, (n_m, t_m))$$

Usando las reglas vistas anteriormente se definen las **transiciones** entre configuraciones del sistema SN P. Dadas dos configuraciones C_1, C_2, una transición entre estas se denota como $C_1 \Rightarrow C_2$.

Definición 2.4. *Se define una* ***computación*** *de un sistema SN P como cualquier sucesión de transiciones empezando por la configuración inicial.*

Una computación se dice de parada si llega a una configuración donde todas las neuronas están abiertas y ninguna regla es aplicable.

Definición 2.5. *Dado un sistema SN P* Π *una configuración* C *se dice alcanzable en* Π *si existe una computación de* Π γ *tal que existe un instante* t *de la computación en el que se alcanza la configuración* C*, es decir que* γ *es de la forma:*

$$\gamma = C_0 \Rightarrow_1 \cdots \Rightarrow_t C \Rightarrow \ldots$$

Definición 2.6. *Se define un bucle de transiciones,* γ_b *de un sistema SN P como una sucesión de transiciones que empieza y acaba en la misma configuración:*

$$\gamma_b = C \Rightarrow \cdots \Rightarrow C$$

Dada una computación (de parada o no), se llama tren de spikes de la computación a la secuencia de ceros y unos describiendo el comportamiento de la neurona de salida (1 si emite un spike y 0 si no), o equivalentemente a la sucesion de instantes en los que la neurona de salida manda spikes. En el siguiente apartado se exponen distintas formas de interpretar estos trenes de spikes como salidas de nuestros sistemas SN P.

2.2.2 Trenes de spikes

En esta sección, como se ha adelantado anteriormente, se verán formas de interpretar las computaciones de los sistemas SN P. Se trabajará con sistemas SN P sin restricción en sus spikes, es decir, que sus neuronas pueden contener una cantidad no acotada de spikes.

Definición 2.7. *Sea* $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_0)$ *un sistema SN P y denotemos por* $\gamma = C_0 \Rightarrow C_1 \Rightarrow C_2 \Rightarrow \ldots$ *a una computación en* Π*, donde* C_0 *es la configuración inicial y* $C_{i-1} \Rightarrow C_i$ *es el paso i-ésimo de la computación. Se define el* ***tren de spikes*** *de la computación* γ *como la sucesión formada por los pasos de computación en los que la neurona de salida emite un spike y se denotará por:*

$$st(\gamma) = \langle t_1, t_2, \ldots \rangle \quad con\ 1 \leq t_1 < t_2 < \ldots$$

Al conjunto de todos los trenes de spikes (sobre todas las computaciones) de Π *se denota por* $ST(\Pi)$*.*

Observación 2.1. Nótese que la sucesión dada por el tren de spikes puede ser finita (si la computación para o si manda un número finito de spikes) o infinita (en cuyo caso la computación no para).

Observación 2.2. Si Π es determinista, entonces solo tendrá una computación posible, y, por tanto, $ST(\Pi)$ será un conjunto unitario.

Se denota por $COM(\Pi)$ al conjunto de todas las computaciones de Π y por $HCOM(\Pi)$ al conjunto de todas las computaciones de parada de Π.

Ya definido el tren de spikes de una computación, lo usaremos para definir nuestra forma de interpretar las salidas de las distintas computaciones de Π, y, así, ver los conjuntos de números que pueden ser computados por los sistemas SN P.

Definición 2.8. *Sea Π un sistema SN P. Se puede definir el conjunto de números computados por $ST(\Pi)$ de las siguientes maneras:*

- *Si se tienen en cuenta las primeras $k \geq 2$ spikes:*

  $$N_k(\Pi) = \{n | n = t_i - t_{i-1}, 2 \leq i \leq k, \gamma \in COM(\Pi),\ \gamma \text{ } st(\gamma) \text{ con al menos } k \text{ spikes}\}$$

- *Si se tienen en cuenta las primeras $k \geq$ spikes y además se exige que tenga exactamente k spikes:*

  $$N_{\underline{k}}(\Pi) = \{n | n = t_i - t_{i-1}, 2 \leq i \leq k, \gamma \in COM(\Pi),\ \gamma \text{ } st(\gamma) \text{ con exactamente } k \text{ spikes}\}$$

- *Si se tiene en cuenta todas los spikes de las computaciones con trenes de spikes infinitos:*

  $$N_\omega(\Pi) = \{n | n = t_i - t_{i-1}, i \geq 2, \gamma \in COM(\Pi),\ \gamma \text{ } st(\gamma) \text{ infinito }\}$$

- *Si se tienen en cuenta todos los intervalos de todas las computaciones:*

  $$N_{all}(\Pi) = \bigcup_{k \geq 2} N_k(\Pi) \cup N_\omega(\Pi)$$

También se tienen en cuenta ciertos subconjuntos de los anteriores:

- *Aquellos que solo tienen en cuenta las computaciones de parada, denotados por $N_k^h(\Pi)$, $N_\omega^h(\Pi)$, que claramente son subconjuntos de $N_k(\Pi)$ y de $N_\omega(\Pi)$ respectivamente.*

- *Aquellos en los que se consideran intervalos alternos:*

$$N^a(\Pi) = \{n | n = t_{2i} - t_{2i-1}, i \geq 1, \gamma \in COM(\Pi)\}$$

  Esta interpretación se puede usar para cualquiera de los conjuntos anteriores, generando $N_k^a(\Pi)$, $N_\omega^a(\Pi)$, $N_{all}^a(\Pi)$ *subconjuntos de* $N_k(\Pi)$, $N_\omega(\Pi)$, $N_{all}(\Pi)$.
- *Por último también se pueden mezclar las restricciones de parada con intervalos alternos dando pie a los conjuntos* $N_\alpha^{ha}(\Pi)$ *con* $\alpha \in \{\omega, all\} \cup \{k | k \geq 2\}$ *y a* $N_{\underline{k}}^{ha}(\Pi)$ *para* $k \geq 2$.

Observación 2.3. Nótese que todas tienen en común que se consideran los intervalos entre spikes consecutivos de la neurona de salida como números computados por una computación.

Observación 2.4. La variante donde se toman intervalos alternos es interesante porque aquellos intervalos que no se registran pueden ser usados para cálculos auxiliares.

Definición 2.9. *Se denota por* $Spik_\alpha^\beta P_m(rule_k, cons_p, forg_q)$ *al conjunto de conjuntos:*

$$Spike_\alpha^\beta P_m(rule_k, cons_p, forg_q) = \{N_\alpha^\beta(\Pi_i)\}_i$$

Donde Π_i *recorre el conjunto de sistemas SN P que tienen a lo más m neuronas con k reglas en cada una, con todas las reglas de disparo* $E/a^r \to a; d$ *cumpliendo* $r \leq p$, *reglas de olvido* $a^s \to \lambda$ *cumpliendo* $s \leq q$ *y entendiendo la computación del SN P como* $N_\alpha^\beta(\Pi_i)$, *con* $\alpha \in \{all, \omega\} \cup \{k | k \geq 2\} \cup \{\underline{k} | k \geq 2\}$ *y* $\beta \in \{h, a, ha\}$ *u omitido. Cuando uno de los parámetros* m, k, s, q *está no acotado, se sustituye por* $*$.

En lo que sigue, se pretenden dar resultados de universalidad para cada uno de los tipos de sistemas SN P descritos en la definición. Antes de empezar con estos resultados, se expone un ejemplo para terminar de familiarizarse con los sistemas SN P.

2.2.3 Ejemplo

A la hora de escribir las reglas haremos la siguiente simplificación. Si existe una regla de disparo $E/a^c \to a; d$ con $L(E) = \{a^c\}$ se reescribirá como $a^c \to a; d$. Se considera el sistema SN P Π dado por la figura 2.1:

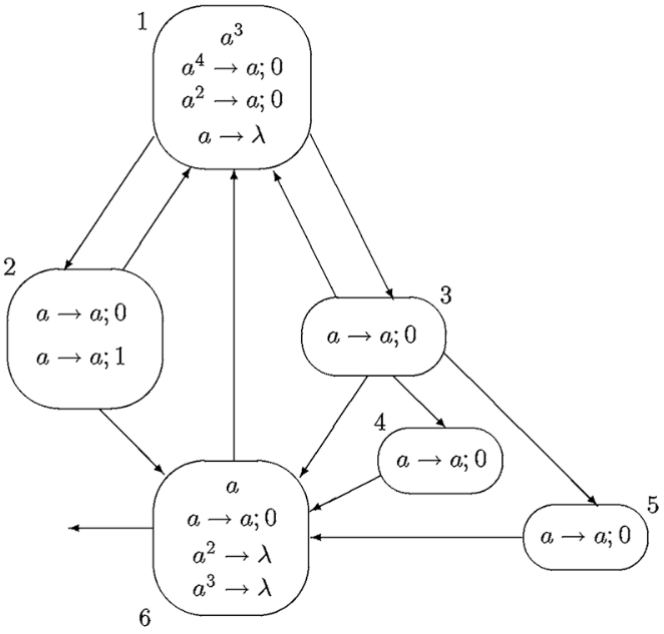


Figura 2.1: Un sistema SN P

Ya que este es el primer ejemplo, antes de explicarlo, demos una descripción formal del sistema, según la definición de sistema SNP.

Para dar un sistema SN P formalmente según su definición, es necesario especificar:

- El número de neuronas $\sigma_1, \ldots, \sigma_m$
- el par (n_i, R_i) denotando el número de spikes de cada neurona σ_i y su conjunto de reglas.
- syn denotando las sinapsis de la neurona.
- La neurona de salida.

De esta forma, el sistema Π de la figura viene definido formalmente como

$$\Pi = (\{a\}, \sigma_1 \ldots, \sigma_6, syn, 6)$$

Donde:

$$n_1 = 3, n_2 = n_3 = n_4 = n_5 = 0, n_6 = 1$$
$$R_1 = \{a^4 \to a; 0, a^2 \to a; 0, a \to \lambda\}$$

$$R_2 = \{a \to a; 0, a \to a; 1\}$$

$$R_6 = \{a \to a; 0, a^2 \to \lambda, a^3 \to \lambda\}$$

$$R_i = \{a \to a; 0\}, i \in \{3, 4, 5\}$$

$$syn = \{(1,2), (1,3), (2,1), (2,6), (3,1), (3,4), (3,5), (3,6), (4,6), (5,6), (6,1)\}$$

Estudiemos cómo funciona este sistema. Nótese que las únicas neuronas con spikes en la configuración inicial son la neurona 1 y la neurona 6. La neurona 1 no tiene ninguna regla aplicable para a^3 así que en el primer paso de computación la neurona de salida dispara la regla $a \to a; 0$ inmediatamente y manda spikes al exterior y a la neurona 1. En el siguiente paso de computación, como la neurona 1 ha recibido un spike de la neurona 6, la neurona 1 puede disparar la regla $a^4 \to a; 0$. Esta regla manda inmediatamente spikes a las neuronas 2 y 3. En el siguiente paso, como la neurona 2 tiene 2 reglas de las que escoger, se escogerá de manera no determinista. Supongamos primero que la neurona 2 elige la regla $a \to a; 0$ (esto es, ambas reglas son aplicables pero se dispara la regla $a \to a; 0$). Entonces, en este paso de computación (tercero), tanto la neurona 2 como la neurona 3 mandan spikes inmediatamente, resultando en la neurona 1 con 2 spikes, la neurona 6 con 2 spikes y las neuronas 4 y 5 con 1 spike. En el siguiente paso de computación (cuarto), la neurona 6 olvida los 2 spikes que tiene, la neurona 1 manda spikes a las neuronas 2 y 3 y las neuronas 4 y 5 mandan spikes a la neurona 6. Al final de este paso tenemos 1 spike en la neurona 2 y 3 que recibieron de la neurona 1 y 2 spikes en la neurona 6 que recibió de las neuronas 4 y 5. Ahora volvemos a un estado que es prácticamente equivalente al estado del paso tercero (en el paso tercero la neurona 6 no tenía spikes y ahora tiene 2 que va a olvidar).

Se puede observar que si 2 usa la regla $a \to a; 0$ "volvemos" al paso 3 de computación tras dos pasos de computación, y la neurona de salida no habrá emitido ningún spike al exterior. Veamos qué pasa si la neurona 2 escoge la regla $a \to a; 1$ (como antes, esto quiere decir que ambas reglas son aplicables pero de manera no determinista se dispara la regla $a \to a; 1$). Retomamos la computación anterior (quinto paso de computación) suponiendo que ahora la neurona 2 escoge la regla $a \to a; 1$. Entonces, se tendrá que la neurona 2 se activa pero no se dispara todavía (hay que esperar al paso sexto) y la neurona 3 manda un spike inmediatamente a las neuronas 1 4 5 y 6. En el sexto paso de computación la neurona 1 olvida el spike que tenía, la neurona 6 manda un spike al exterior y a la neurona 1 y las neuronas 4 y 5 mandan spikes a la neurona 6. Además de esto, la neurona 2 que estaba mandando un spike con retraso manda este spike que llega a la neurona 1 y a la neurona 6. Por tanto, al final del paso de computación se tienen 3 neuronas en la neurona 6 (de la 2 la 4 y la 5) y 2 neuronas

en la neurona 1 (de la 2 y de la 6). Observamos entonces que en el séptimo paso de computación volvemos a estar en un estado "equivalente al tercero y al quinto, solo que ahora, como en el quinto se escogió la regla $a \to a$ en la neurona 2, en el paso 6 la neurona 6 ha emitido un spike al exterior. Es fácil entonces intuir que las distancia entre spikes de la neurona 6 viene dado por $1 + 2i$ para algún $i \geq 1$ y que la computación nunca para. Este i vendrá determinado por las reglas que vaya tomando la neurona 2 de manera no determinista. Se puede consultar una computación hecha como ejemplo en la tabla siguiente:

Neurona	1	2	3	4	5	6	ent
Inicial	a^3	—	—	—	—	a	
Paso 1	— a^3a_6	— —	— —	— —	— —	$a \to a;0$ —	a
Paso 2	$a^4 \to a;0$ —	— a_1	— a_1	— —	— —	— —	
Paso 3	— a_2a_3	$a \to a;0$ —	$a \to a;0$ —	— a_3	— a_3	— a_2a_3	
Paso 4	$a^2 \to a;0$ —	— a_1	— a_1	$a \to a;0$ —	$a \to a;0$ —	$a^2 \to \lambda$ a_4a_5	
Paso 5	— a_3	$a \to a;1$ —	$a \to a;0$ —	— a_3	— a_3	$a^2 \to \lambda$ a_3	
Paso 6	$a \to \lambda$ a_2a_6	spiking —	— —	$a \to a;0$ —	$a \to a;0$ —	$a \to a;0$ $a_2a_4a_5$	a
Paso 7	$a^2 \to a;0$ —	— a_1	— a_1	$a \to a;0$ —	$a \to a;0$ —	$a^3 \to \lambda$ —	
Paso 8	— a_3	$a \to a;1$ —	$a \to a;0$ —	— a_3	— a_3	— a_3	
Paso 9	$a \to \lambda$ a_2a_6	spiking —	— —	$a \to a;0$ —	$a \to a;0$ —	$a \to a;0$ $a_2a_4a_5$	a
Paso 10	$a^2 \to a;0$ —	— a_1	— a_1	— —	— —	$a^3 \to \lambda$ —	

En la tabla aparecen 10 pasos de computación en los cuales se mandan spikes en los pasos 1 6 y 9, computando así los números 5 y 3. En cada fila de la tabla se da la regla que se aplica en cada paso de computación y los spikes que quedan después de completar el paso para cada neurona. Si ninguna regla se aplica o si no hay ningún spike en la neurona se representa con un guión. Nótese que, como se destacó en

el párrafo explicativo, la configuración del sistema tras aplicar los pasos 2,7 y 10 es la misma, que es justo cuando se manda un spike. Esto parece indicar que ninguna computación del sistema SN P Π es de parada y que $N_\alpha(\Pi) = \{1 + 2i | i \geq 1\}$ para $\alpha \in \{\omega, all\} \cup \{k | k \geq 2\}$.

Para terminar con el ejemplo, se da una descripción formal de cada una de las configuraciones descritas en la tabla, es decir, para los 10 primeros pasos de la computación. Se denotará por C_0 a la configuración inicial y por C_i a la configuración a la que se llega tras aplicar el paso i. Recuerdese que las configuraciones se describen como $C = ((n_1, t_1), \ldots, (n_6, t_6))$ donde n_i es el número de spikes en la neurona i y t_i el tiempo que les queda para abrirse, y las computaciones no son más que una sucesión de transiciones entre configuraciones, empezando por la configuración inicial.

$$C_0 = ((3,0),(0,0),(0,0),(0,0),(0,0),(1,0))$$
$$C_1 = ((4,0),(0,0),(0,0),(0,0),(0,0),(0,0))$$
$$C_2 = ((0,0),(1,0),(1,0),(0,0),(0,0),(0,0))$$
$$C_3 = ((2,0),(0,0),(0,0),(1,0),(1,0),(2,0))$$
$$C_4 = ((0,0),(1,0),(1,0),(0,0),(0,0),(2,0))$$
$$C_5 = ((1,0),(0,1),(0,0),(1,0),(1,0),(1,0))$$
$$C_6 = ((2,0),(0,0),(0,0),(0,0),(0,0),(3,0))$$
$$C_7 = ((0,0),(1,0),(1,0),(0,0),(0,0),(0,0)) = C_2$$
$$C_8 = ((1,0),(0,1),(0,0),(1,0),(1,0),(1,0))$$
$$C_9 = ((2,0),(0,0),(0,0),(0,0),(0,0),(3,0))$$
$$C_{10} = C_2 = C_7$$

Y la computación ilustrada se denotaría como

$$C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_{10} \Rightarrow \ldots$$

3 Resultados de universalidad

En este capítulo se exponen resultados de universalidad asociados a los conjuntos dados en las definiciones 2.8, 2.9. Daremos resultados para sistemas SN P que no tienen el número de spikes acotados y para sistemas SN P que sí lo tienen.

3.1 Universalidad para sistemas SN P con spikes no acotados

Comenzamos con un resultado muy importante, que es la base para muchos otros teoremas de la sección. Este teorema enuncia que

Teorema 3.1. *$Spik_{\underline{2}}^{\beta}P_*(rule_k, cons_p, forg_q) = NRE$ para todo $k \geq 2, p \geq 3, q \geq 3$ con $\beta = h$ u omitido.*

Demostración. $\boxed{\subseteq}$ Esta contención es inmediata teniendo en cuenta que

$$Spik_{\underline{2}}^{\beta}P_m(rule_k, cons_p, forg_q) \subseteq Spik_{\underline{2}}^{\beta}P_{m'}(rule_{k'}, cons_{p'}, forg_{q'}) \subseteq$$

$$\subseteq Spik_{\underline{2}}^{\beta}P_*(rule_*, cons_*, forg_*) \subseteq NRE$$

para todo $m' \geq m \geq 1, k' \geq k \geq 1, p' \geq p \geq 1, q' \geq q \geq 0$. Esta cadena de contenciones se deduce de la propia definición de los conjuntos a excepción de la última, para la que se acude a la tesis de Church-Turing.

$\boxed{\supseteq}$ Para esta contención, es suficiente demostrar que

$$NRE \subseteq Spik_{\underline{2}}^{\beta}P_*(rule_2, cons_3, forg_3)$$

Para ello, usaremos el teorema 2.1, que enunciaba que toda máquina de registro computa todos los conjuntos de números Turing computables (y por tanto caracterizan NRE). Así, si dada cualquier máquina de registro $M = (m, H, l_0, l_h, I)$ soy capaz de construir un sistema SN P que simule el comportamiento de la máquina de registro M y de forma que la neurona de salida del sistema SN P mande exactamente dos spikes, t_1, t_2, cumpliendo que $t_2 - t_1$ sea un número computado por M, se demostrará el resultado.

Esto es porque dado $Q \in NRE$, por su caracterización mediante máquinas de registro, se tendrá que existe una máquina de registro M tal que $N(M) = Q$. Si soy capaz de simular la máquina de registro con un sistema SN P, Π, de la manera anterior, se tendrá que $N_{\underline{2}}^{\beta}(\Pi) = N(M) = Q$.

Sea entonces $M = (m, H, l_0, l_h, I)$ una máquina de registro arbitraria. Para simular M se construirán tres "módulos", simulando las instrucciones ADD, SUB y $HALT$ de las máquinas de registro en un sistema SN P y se comprobará que funcionan bien conjuntamente.

Estos módulos se darán de forma informal, proporcionando esquemas para cada uno, ya que si se desarrolla todos los detalles técnicos será más difícil de seguir. Comenzamos explicando las etiquetas asociadas a cada una de las neuronas de los módulos. Cada neurona etiquetada con el nombre l_i corresponderá a la etiqueta l_i asociada a una instrucción de I de la máquina de registro. Para cada simular la instrucción etiquetada por l_i se ejecutará uno de los tres módulos del sistema SN P. En estos simuladores, a parte de las neuronas representando las etiquetas de la máquina de registro, habrá otras neuronas auxiliares, denotadas por l_i', l_i'', l_i''' y por br("before r") que nos ayudarán a simular el no determinismo y a que todo el sistema SN P esté coordinado. Además, cada registro de M tendrá asociada una neurona de Π denotada por r, de forma que si el valor del registro r en la máquina es n, la neurona r contendrá $2n$ spikes. Por último, también se incluirán neuronas etiquetadas por $f_1, f_2, f_3, f_4, f_5, f_6, out$ incluidas en el módulo $HALT$. Toda esta notación se comprenderá mejor conforme se vayan viendo los distintos módulos.

La idea básica para entender los módulos es la siguiente. Cuando una instrucción de la máquina de registro etiquetada por l_i va a ser ejecutada, esto se corresponde con un estado de mi sistema SN P en el que la neurona con la etiqueta l_i contiene 2 spikes y va a ser disparada. Después de disparar, comienza una serie de pasos (que varían en función de la instrucción) hasta ejecutar pasos equivalentes a la instrucción con etiqueta l_i, pasando a otra instrucción y repitiendo el proceso. Con esta forma de plantear, la configuración inicial de nuestro sistema SN P vendrá dado con todas

las neuronas vacías (pues se empieza con los registros a cero) menos la neurona con etiqueta l_0, que comienza con 2 spikes.

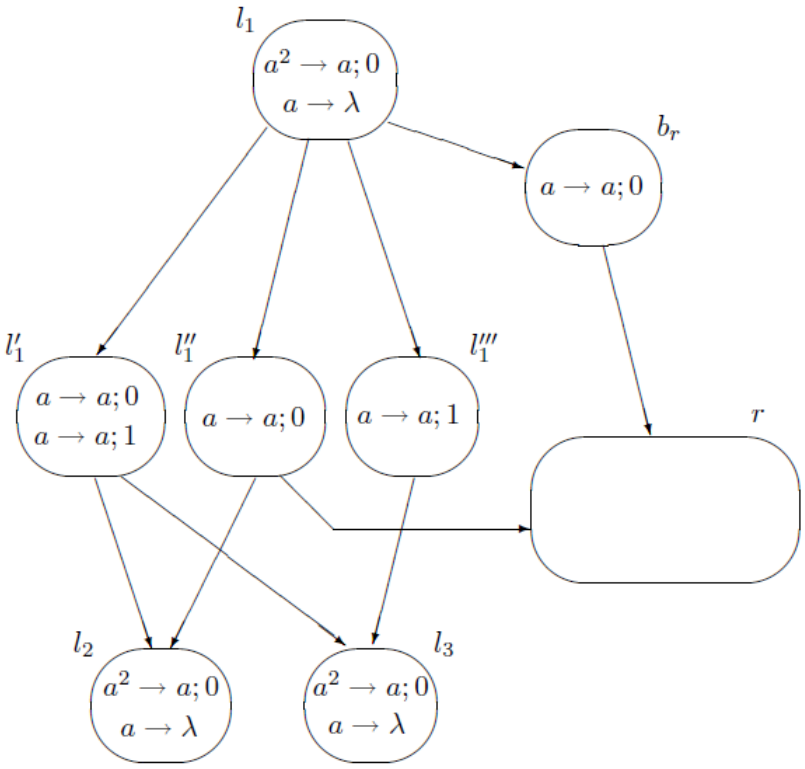


Figura 3.1: Módulo ADD (simulando $l_1 : (ADD(r), l_2, l_3)$)

Instruccion ADD. $l_1 : (ADD(r), l_2, l_3)$: Supongamos que nos encontramos en un paso de computación donde tenemos que simular la instrucción $l_1 : (ADD(r), l_2, l_3)$ con dos spikes en l_1, y ninguna neurona más con spikes, excepto las neuronas asociadas con los registros. Este módulo debe hacer dos cosas, primero, sumar 1 r al registro (es decir, añadir 2 spikes a la neurona r), y segundo escoger de manera no determinista si irse a la etiqueta l_2 o la l_3. En el primer paso del módulo la neurona l_1 se dispara, mandando un spike a l_1', l_1'', l_1''' y a br. En el siguiente paso, las neuronas br y l_1'' mandan spikes a r, completando así el primer paso de aumentar el registro r en 1(o equivalentemente los spikes en 2). Nótese que en la figura aparece que la neurona r no contiene ninguna regla, veremos que sí que las contiene, pero solo serán aplicables cuando r tenga un número impar de spikes. Falta entonces ver que el módulo decide no deterministicamente entre l_2 y l_3.

Para conseguir el no determinismo, se usa el no determinismo de la neurona l_1'.

Veamoslo por casos. Supongamos primero que la neurona l_1' escoge de manera no determinista aplicar la regla $a \to a; 0$, entonces las neuronas l_2 y l_3 recibirán spikes inmediatamente de parte de l_1'. Además, la neurona l_1'' dispara también inmediatamente a la neurona l_2, mientras que la neurona l_1''' dispara con retraso, implicando que en el siguiente paso de computación la neurona l_2 tiene 2 spikes (y por tanto dispara) y la neurona l_3 olvida el spike que tenía (y vuelve a recibir otro que olvida en el siguiente paso de computación). En este caso se ha escogido la neurona l_2 de manera no determinista. Veamos el otro caso. Supongamos ahora que la neurona l_1' escoge la regla $a \to a; 1$. Entonces, mientras que la neurona l_1'' dispara inmediatamente un spike a l_2 y a l_3, las neuronas l_1', l_1''' disparan con retraso de 1. De esta forma, en el siguiente paso de computación, se tiene solo 1 spike en l_2 (que se olvida), le llega otro nuevo (que se olvida en el paso siguiente) y llegan 2 spikes a l_3, haciendo que se dispare.

Así, con este módulo hemos conseguido aumentar en 2 el número de spikes en r y pasar de manera no determinista a l_2 o l_3.

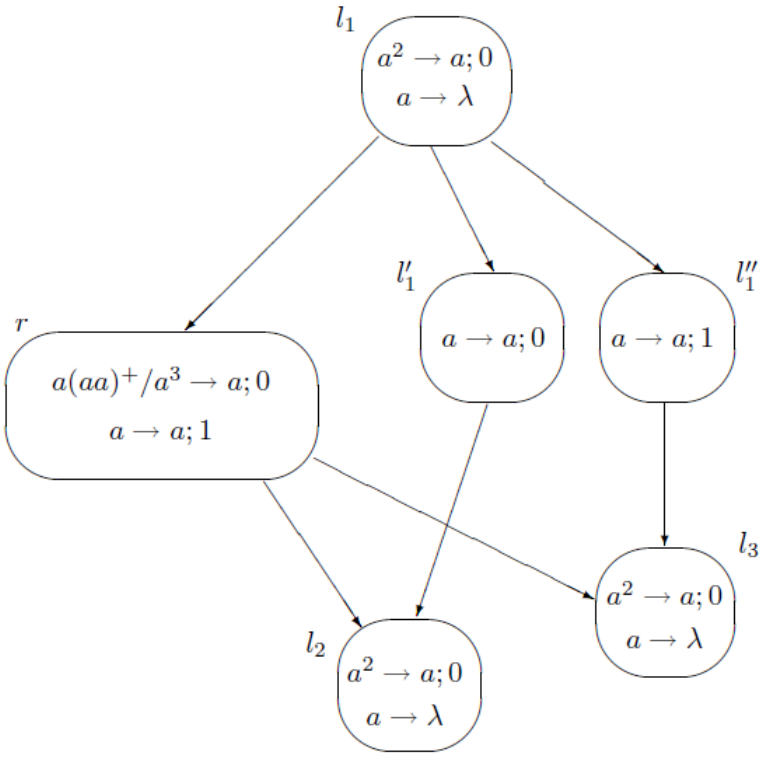


Figura 3.2: Módulo SUB (simulando $l_1 : (\text{SUB}(r), l_2, l_3)$)

Instruccion SUB. $l_1 : (\text{SUB}(r), l_2, l_3)$: Supongamos que, como se hizo en el mó-

dulo ADD, nos encontramos en una paso de computación donde se tiene que simular la instrucción l_1, que se corresponde con la instrucción SUB. Por esto, supongamos que en este instante de la computación la neurona etiquetada por l_1 contiene 2 spikes. El comportamiento de este módulo dependerá del valor del registro. Se debe comportar de la siguiente manera. Si el valor del registro es distinto de cero, resta 1 y ve a la instrucción con etiqueta l_2, si el valor del registro es 0, ve a la instrucción con etiqueta l_3. En el primer paso de computación, como se indica en la figura 3.2, la neurona l_1 manda un spikes al registro y a las neuronas l_1', l_1''. Nótese que la neurona etiquetada por r posee 2 reglas, ambas solo aplicables si r contiene un número impar de spikes (explicando por tanto el por qué el registro r no se disparaba en el módulo ADD). La aplicabilidad de estas reglas determinará si el registro estaba a cero o no y, por tanto, a qué etiqueta se debe ir. Supongamos en primer lugar que el registro estaba a cero. Entonces, El único spike contenido en r es el spike que le ha llegado de l_1. Así, la única regla aplicable es $a \to a$; 1. Paralelamente, las reglas de l_1' y l_1'' disparan en este paso. De manera similar a como funcionaba el módulo ADD, teniendo en cuenta el retraso de los spikes de las neuronas r y l_1'', se concluye que tras otro paso de computación la neurona l_3 recibe 2 spikes y la computación continúa por esa rama. Supongamos ahora que el registro r no estaba a cero, digamos que $r = n$. Entonces, el registro r contenía a^{2n} spikes. Tras recibir el spike de l_1 r pasa a contener a^{2n+1} spikes. Es aquí cuando la regla $a(aa)^+/a^3 \to a$; 0 es la única aplicable. Al dispararse, consume 3 spikes (dejando en r a^{2n-2} spikes, es decir, $r = n - 1$). De nuevo, de forma similar al módulo ADD, teniendo en cuenta que ahora r genera un spike inmediatamente junto con l_1', la rama de la computación continúa por l_2, y por tanto el módulo SUB funciona bien.

Nótese que los módulos SUB y ADD se pueden unir. Esto se sigue de varios argumentos. En primer lugar, por definición de las máquinas de registro, se tiene que cada instrucción tiene una etiqueta asociada, y por tanto las únicas neuronas compartidas por varios módulos serían neuronas que representan los registros,r, y la neurona b_r. Como la neurona b_r solo entra en juego cuando activamos el módulo ADD, no hay problema. Con respecto a las neuronas registro, nótese que solo activan reglas en el módulo SUB. Además, se tiene que una neurona r estará conectada a una instrucción l_i si existe una instrucción l_j de la forma $l_j : (SUB(r), l_k, l_m)$ con $k = i$ o $m = i$. De esta forma, cada vez que un registro mande un spike mandará un spike a todas estas neuronas (estén o no estén en la instrucción que se esté ejecutando). Esto no es un problema, pues en el siguiente paso de computación todas las neuronas l_i contendrán 1 único spike (y por tanto lo olvidarán) menos aquella instrucción por donde continúa la computación. Concluimos entonces que los módulos se pueden unir sin problema.

Por último, veamos cómo implementar la instrucción de parada. El módulo se

puede visualizar gráficamente en la figura 3.3. **Instrucción $HALT$:** Supongamos que

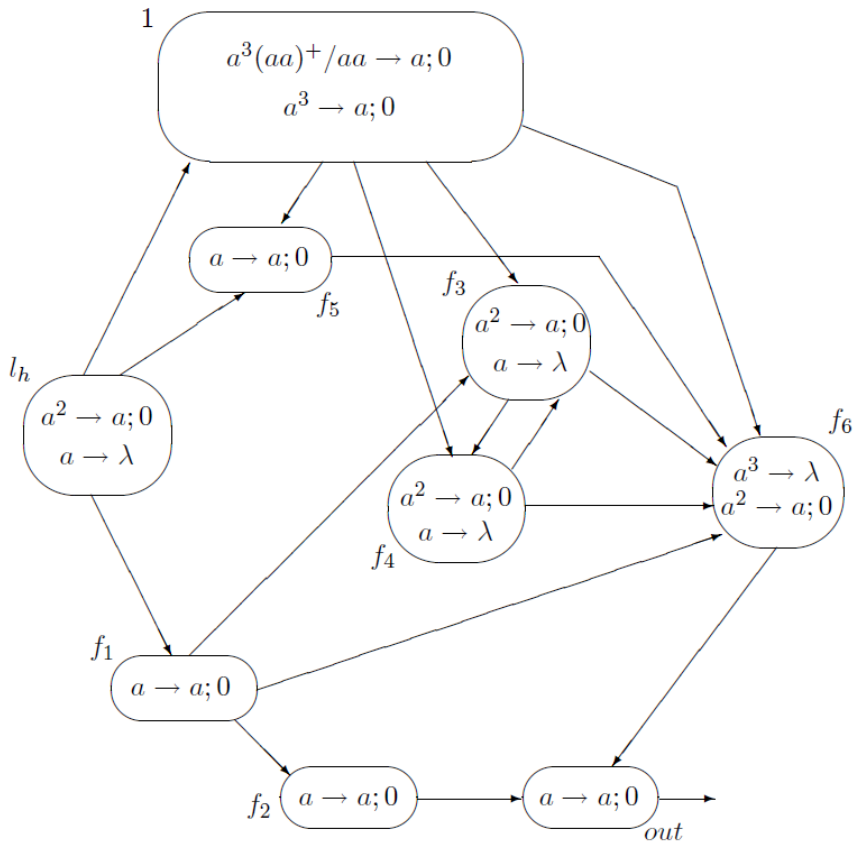


Figura 3.3: Módulo HALT

la computación de M, la máquina de registro, para, esto es, se llega a la instrucción con etiqueta l_h. En nuestro sistema Π, esto se corresponde con que se la neurona etiquetada con l_h contiene 2 spikes. Supongamos además que el contenido del registro 1 de la máquina M es n, por lo tanto en nuestro sistema Π se tendrán $2n$ spikes en la neurona 1. El objetivo del módulo $HALT$ será disparar exactamente 2 spikes con una distancia entre ellos de n. Así, se tendrá que $N_{\underline{2}}^{\beta}(\Pi) = N(M)$, demostrando el resultado.

Veamos cómo funciona el módulo. En primer lugar nótese que las reglas del registro 1 son distintas a las reglas dadas en la instrucción SUB. Esto es posible pues hemos supuesto que el valor en el registro 1 nunca decrecía, luego nunca hemos tenido que aplicarle la instrucción SUB.

Supongamos entonces que estamos en el paso t de la computación de mi sistema Π, con 2 spikes en la neurona l_h. La idea del funcionamiento del módulo es la siguiente:

1. Por un lado, la computación continuará por f_1, f_2, f_3 generando un spike en el instante $t+3$.
2. Por otro lado, con la ayuda de $f_3, f_4, 1$ y f_5 (también f_1 en el paso $t+1$), el módulo f_6 recibirá 3 spikes en cada paso de computación hasta que el registro 1 se quede sin spikes. Esto hará que vayan pasando instantes de tiempo (uno por cada 2 spikes que olvida el registro 1) hasta que hayan pasado $t+n$ instantes. Entonces, tras un par de pasos de computación f_6 por fin contendrá solo 2 spikes, mandando un spike a la neurona de salida y acabando la computación con otro spike en el instante $t+n+3$.

Veámoslo en detalle. En el instante t, l_h dispara 3 spikes, a f_1, a f_5 y a 1. Seguidamente, en el instante $t+1$, la neurona 1 manda spikes a f_3, f_4, f_5 y f_6 con la regla $a^3(aa)^+/aa \to a; 0$. la neurona f_1 manda spikes a f_3 (empezando el ciclo de f_3, f_4 que veremos en el instante $t+2$), a f_2 y a f_6. Así, f_6 ha recibido 3 spikes, de f_1, f_5 y 1. En el instante $t+2$ el spike de f_2 pasa a la neurona de salida. Por otro lado, la neurona f_6 que había recibido 3 spikes se olvida de ellos. Del paso anterior, f_5 recibió 1 spike de 1, que vuelve a mandar a f_6. Notemos ahora el papel de f_3 y f_4. En el instante $t+2$ f_3 manda spikes a f_6 y a f_4, mientras que f_4 se olvida del spike que tenía. Teniendo en cuenta que ambas reciben spikes de 1, en el paso $t+3$ será f_4 la neurona que tenga 2 spikes (y que por tanto dispare a f_6) mientras que f_3 simplemente se olvidará del spike recibido. La neurona 1, a parte de enviar spikes a f_3 y f_4 también manda spikes a f_5 y a f_6. Se tiene por tanto que en el siguiente paso de computación, $t+3$, f_6 contendrá 3 spikes que olvidará y $f_5, f_4, 1$ podrán disparar de nuevo, volviéndose a repetir este proceso. Además, en el paso $t+3$ la neurona de salida disparará registrando un spike en el instante $t+3$. Supongamos que han pasado ya n instantes y que por tanto estamos en el paso de computación $t+n$. En este instante, el contenido de la neurona 1 será de a^3 spikes y por tanto será la última vez que se disparará. Supongamos, sin pérdida de generalidad, que en f_3 hay 2 spikes en el paso $t+n$ (si no las hubiera simplemente se considera f_4). Entonces, análogamente a antes, se tendrá que en el paso $t+n+1$ la neurona 1 no contendrá spikes, las neuronas f_5 y f_3 contendrán un spike, y la neurona f_6 contendrá 3 spikes. Se tiene entonces que la neurona f_6 se olvida de los spikes y recibe 2 spikes de f_5 y f_4 (nótese que como 1 ya no contiene spikes solo recibe 2). Así, en el paso $t+n+2$ las únicas neuronas con spikes son f_3 (que tiene 1 y la olvida) y f_6 con 2 spikes, que manda un spike a la neurona de salida, que finalmente en el instante $t+n+3$ dispara.

Como se adelantó, se tiene por tanto que el intervalo entre spikes es $n = t+n+3-t-3$, que es justamente el valor del registro 1 de la máquina de registro M. Además, se tiene que ninguna neurona contiene spikes, luego la computación es de parada. Se demuestra así que $N_{\underline{2}}^{\beta}(\Pi) = N(M)$ completando la prueba.

Es interesante destacar que el resultado se tiene para $\beta = h$ porque siempre que el sistema SN P genera un número $n \in N(M)$, este emite 2 spikes y para. Por lo tanto, si tenemos en cuenta las computaciones de parada el resultado sigue siendo válido.

Teorema 3.2. *Se tiene que* $Spik_{\alpha}^{\beta}P_{*}(rule_k, cons_p, forg_q) = NRE$ *para todo* $k \geq 2, p \geq 3, q \geq 3, \alpha \in \{all, 2, \underline{2}\}, \beta \in \{h, a, ha\}$ *u omitido.*

Demostración. Del teorema 3.1 se tiene que se puede construir un sistema SN P Π que simula una máquina de registro. Las computaciones de Π o bien no paran (si la máquina de registro no para) o bien mandan exactamente 2 spikes y después paran. Se deduce por tanto que:

$$N(M) = N_2(\Pi) = N_{\underline{2}}(\Pi) = N_2^{\beta}(\Pi) = N_{\underline{2}}^{\beta}(\Pi) \text{ para todo } \beta \in \{h, a, ha\}$$

Por otro lado, como las computaciones infinitas del sistema SN P Π construido en 3.1 no mandan spikes, se tiene que $N_{\omega}^{\beta}(\Pi) = \emptyset$ para $\beta = a$ u omitido (los que paran siempre serán vacíos porque en $N_{\omega}^{\beta}(\Pi)$ solo tiene en cuenta computaciones infinitas). Así, por definición de $N_{all}(\Pi)$, el teorema 3.1 y por la tesis de Church-Turing,

$$NRE \subseteq N_2^{\beta}(\Pi) \subseteq \bigcup_{k\geq 2} N_k^{\beta}(\Pi) = \bigcup_{k\geq 2} N_k^{\beta}(\Pi) \cup N_{\omega}^{\beta}(\Pi) = N_{all}^{\beta}(\Pi) \subseteq NRE$$

Concluyendo la prueba.

Teorema 3.3. $Spik_{\omega}^{a}P_{*}(rule_k, cons_p, forg_q) = NRE$ *para todo* $k \geq 2, p \geq 3, q \geq 3$

Demostración. La prueba de este resultado es otra vez una consecuencia del teorema 3.1. La idea será añadir un módulo al sistema SN P del teorema 3.1 haciendo que las computaciones de parada se repitan una y otra vez. En la figura se puede ver un esquema del módulo añadido para la demostración.

Consideremos entonces el sistema Π dado en la demostración del teorema 3.1. Recuérdese que se iniciaba con 2 spikes en la neurona de salida, l_0, y acababa (si paraba) con la neurona de salida, i_0, mandando exactamente 2 spikes. Sea Π' el sistema SN P añadiendo 2 neuronas, c_1, c_2, como se indica en la figura 3.4. Nótese que estas neuronas al principio no contienen ningún spike y que reciben spikes de i_0 y mandan a l_0. El propósito de estas dos neuronas es "volver a iniciar" el sistema Π (si es

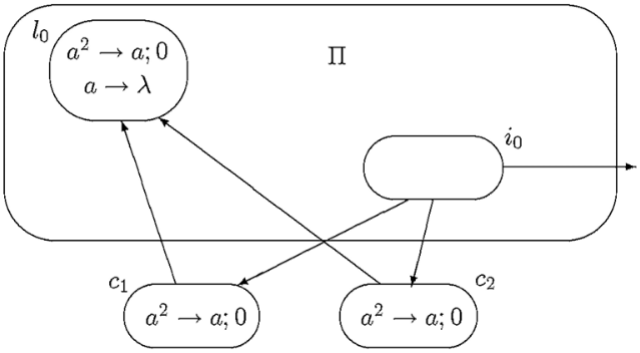


Figura 3.4: Idea de la construcción de la demostración del teorema 3.3

que para). Si el sistema Π para quiere decir que mandará 2 spikes en un intervalo de tiempo $n \in N(M)$. Estos 2 spikes serán recibidos por c_1, c_2, que en el paso siguiente mandan ambos un spike a la neurona l_0. Esto hace que el sistema Π vuelva a empezar una computación (simulando M) y que por tanto la computación de Π′ será infinita. Estudiemos en concreto el valor de $N^a_\omega(\Pi')$ por casos sobre $N(M)$.

Caso 0 ($N(M) = \emptyset$): Si se tiene que $N(M) = \emptyset$ quiere decir que todas las posibles computaciones de de M no paran, por tanto las computaciones de Π no paran, luego las computaciones de Π′ no paran. Así, $N^a_\omega(\Pi') = \emptyset = N_2(\Pi) = N(M)$.

Caso 1 ($N(M) \neq \emptyset$): Veamos que fijada M (y por tanto fijados Π y Π′), se tiene que $N^a_\omega(\Pi') = N_2(\Pi) = N(M)$. Nótese que por definición del sistema Π′ se tiene que $N^a_\omega(\Pi') \subset N_2(\Pi) = N(M)$. Esto es porque en caso de que Π′ emita spikes siempre lo hará de la misma forma que Π (por construcción) y porque solo consideramos intervalos alternos. Sea entonces $n \in N(M)$. Veamos que $n \in N^a_\omega(\Pi')$. Como $N(M) = N_2(\Pi)$, se tiene que existe una computación γ de Π tal que $st(\gamma) = \langle t, t+n \rangle$. A partir de γ construyamos una computación γ' de Π′ cumpliendo $st(\gamma') = \langle t_0, t_0 + n, t_1, t_1 + n, t_2, t_2 + n, \dots \rangle$. Nótese que el tren de spikes es infinito y, contando solo los spikes alternos, se deduce que $n \in N^a_\omega(\Pi)$, obteniendo así el resultado.

Considérese la computación de Π′ que comienza por la computación γ de Π (sabemos que existirá una por el no determinismo del sistema). Supongamos que llegamos al paso t donde se emite el primer spike. En este paso un spike es mandado a las neuronas c_1, c_2. Tras n instantes, la neurona i_0 manda otro spike, mandando otros dos spikes a las neuronas c_1, c_2, que disparan cada una un spike a la neurona l_0. Esto hace que otra computación de la máquina de registro M sea simulada por el sistema Π,

comenzando en el instante $t + n + 2$. Supongamos que esta simulación vuelve a ser la computación γ. Así, en el instante $2t + n + 2 := t_1$ se manda un spike al exterior y a las neuronas c_1, c_2. Análogamente a antes, se llega a 2 spikes en la neurona l_0 en el instante $2t + 2n + 4$, donde se empieza de nuevo otra computación. Si se sigue así de forma infinta (que es posible por el no determinismo), se llega a una computación infinita γ' de Π', justamente cumpliendo $st(\gamma') = \langle t_0, t_0 + n, t_1, t_1 + n, t_2, t_2 + n, \dots \rangle$, deduciendo que $n \in N^a_\omega(\Pi')$ y por tanto que $N(M) = N^a_\omega(\Pi')$.

Por tanto se tiene que para toda M máquina de registro existe un sistema SN P, Π', de forma que $N^a_\omega(\Pi') = N(M)$, deduciendo así el resultado.

Al módulo anterior se le pueden hacer ciertas modificaciones, cubriendo así el caso de N^{ha}_α para $\alpha \in \{k, \underline{k} | k \geq 2\}$. Este resultado se demuestra en el siguiente teorema. Se da simplemente la idea de la construcción, la demostración detallada se puede razonar de manera análoga a lo anterior, haciendo algún ligero cambio.

Teorema 3.4. *$Spik^{ha}_\alpha P_*(rule_h, cons_p, forg_q) = NRE$ para todo $h \geq 2, p \geq \max(3, k)$, $q \geq 3$ y $\alpha \in \{k, \underline{k} | k \geq 2\}$.*

Demostración. Análogamente a la construcción de la prueba anterior, se parte del sistema SN P construido en el teorema 3.1. La idea de la prueba es añadir un módulo que haga que vuelva a empezar el sistema como en el teorema 3.3 pero haciendo que pare tras k veces. La figura 3.5 representa los cambios hechos.

Este sistema SN P funciona de la siguiente manera. Supongamos que estamos en el marco del caso 1 del teorema 3.3, y "repetimos" la computación γ. Las primeras $k - 1$ veces, el sistema descrito en la figura, Π' funcionará como el del teorema 3.3, reiniciando nuestro simulador Π. Nótese que cada vez que se manda un spike a la neurona d_1, que mandará spikes cuando contenga a^k spikes. Esto pasará justamente cuando se hayan mandado k spikes al exterior. En este punto, el objetivo es hacer que el sistema no emita más spikes y pare. Esto se consigue haciendo que las neuronas d_2, d_3, d_4 manden spikes a c_1, c_2 (haciéndoles contener 3 spikes por lo que ya su regla no será nunca aplicable) y a i_0, que de manera análoga a c_1 y c_2 contendrá 2 spikes y ya su regla nunca será aplicable (hemos hecho "rebosar" la neurona). De esta forma, existirá una computación (por ejemplo la computación γ que llega a i_0 pero no dispara porque tiene spikes de más) donde mi sistema Π' para, emite justamente k spikes y cumple que $N^{ha}_k(\Pi') = N^{ha}_{\underline{k}}(\Pi') = N_2(\Pi)$, terminando la demostración.

Nótese un pequeño detalle que no se ha justificado. Desde que Pi emite el k-ésimo

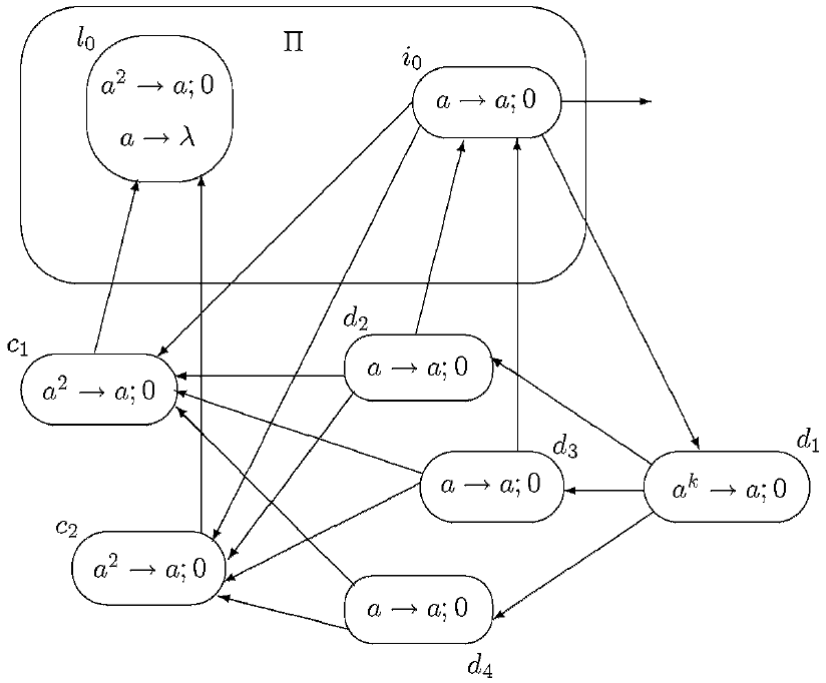


Figura 3.5: Idea de la construcción de la demostración del teorema 3.4

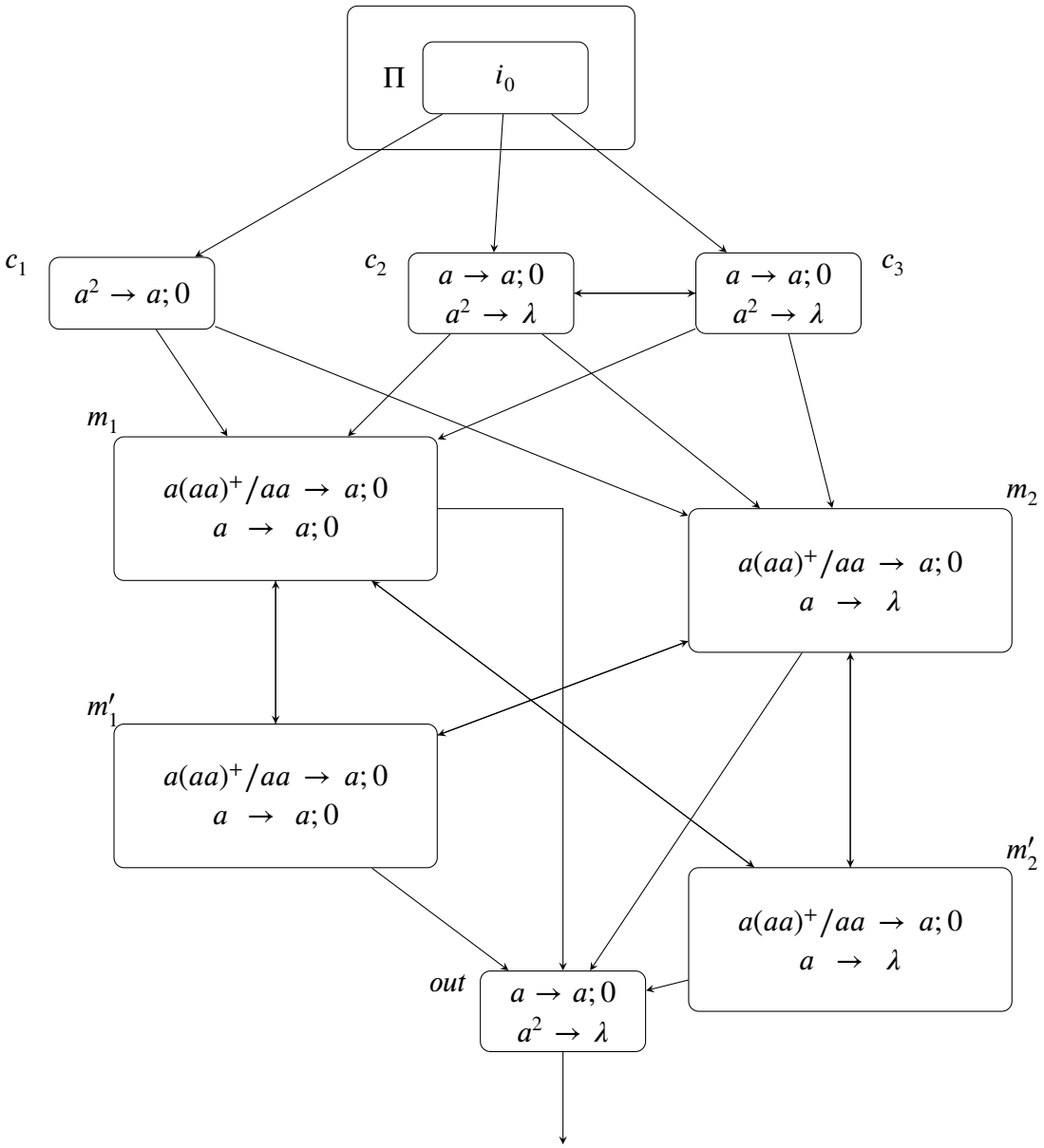
spike hasta que las neuronas c_1, c_2 e i_0 rebosan pasan 2 pasos. ¿Podría pasar que mi sistema Π emitiera otro spike antes de esto? Si se observa la construcción del sistema Π en la demostración del teorema 3.1 se puede ver que son necesarios al menos 3 pasos para que el sistema emita un spike (en el módulo $HALT$, hay que pasar por f_1, f_2 y *out*), por tanto nunca podrá emitir otro spike.

En la construcción de estos dos últimos teoremas ha sido clave el hecho de que estemos considerando intervalos alternos para poder "reiniciar" la computación. Para los casos donde no se consideran intervalos alternos es necesario el siguiente lema técnico.

Lema 3.1. Sea Π un sistema SN P tal que cada computación γ de Π cumple que o bien $st(\gamma) = \emptyset$ o bien $st(\gamma) = \langle t, t+n \rangle$. Entonces existe un sistema SN P Π′ cumpliendo:

1. Para cada computación γ de Π con $st(\gamma) = \langle t, t+n \rangle$ existe una computación γ' de Π′ tal que $st(\gamma') = \langle \hat{t}, \hat{t}+(n+1), \hat{t}+2(n+1), \dots \rangle$.
2. Cada computación γ' de Π′ o bien no manda nunca spikes, o bien cumple$st(\gamma') = \langle \hat{t}, \hat{t}+(n+1), \hat{t}+2(n+1), \dots \rangle$ con $\langle t, t+n \rangle$ siendo el tren de spikes de una computación de Π.

Demostración. Sea Π un sistema SN P tal como se dice en el enunciado del lema. Se construirá el sistema Π′ de acuerdo con la siguiente figura :



Formalmente, dado $\Pi = (\{a\}, \sigma_1, \ldots, \sigma_m, syn, i_0)$ se construye el sistema

$$\Pi' = (\{a\}, \sigma_1, \ldots, \sigma_m, \sigma_{c_1}, \sigma_{c_2}, \sigma_{c_3}, \sigma_{m_1}, \sigma_{m_2}, \sigma_{m'_1}, \sigma_{m'_2}, \sigma_{out}, syn', out)$$

definido manteniendo las neuronas de Π como tal y como son, añadiendo las neuronas

$\sigma_{c_1}, \sigma_{c_2}, \sigma_{c_3}, \sigma_{m_1}, \sigma_{m_2}, \sigma_{m'_1}, \sigma_{m'_2}, \sigma_{out}$ dadas por:

$$\sigma_{c_1} = (0, \{a^2 \to a; 0\})$$

$$\sigma_{c_i} = (0, R_{c_i}), R_{c_i} = \{a \to a; 0, a^2 \to \lambda\}, i \in \{2, 3\}$$

$$\sigma_{m_1} = \sigma_{m'_1} = (0, R_1), R_1 = \{a(aa)^+/aa \to a; 0, a \to a; 0\}$$

$$\sigma_{m_2} = \sigma_{m'_2} = (0, R_2), R_2 = \{a(aa)^+/aa \to a; 0, a \to \lambda\}$$

$$\sigma_{out} = (0, R_{out}), R_{out} = \{a \to a; 0, a^2 \to \lambda\}$$

Y con las sinapsis cumpliendo:

$$syn' = syn \cup A$$

Donde A es la unión de los siguientes conjuntos:

$$\{(i_0, c_i) : i = 1, 2, 3\}$$

$$\{(c_i, m_j) : i = 1, 2, 3; j = 1, 2\}$$

$$\{(m_i, m'_j) : i = 1, 2; j = 1, 2\}$$

$$\{(m_1, out), (m'_1, out), (m_2, out), (m'_2, out), (c_2, c_3), (c_3, c_2)\}$$

Veamos intuitivamente que así definido cumple lo enunciado en el lema. El sistema Π' está construido de tal manera que comienza simulando una computación de Π. De esta forma, si Π no para, entonces Π' no para. Supongamos entonces que Π simula una computación γ de forma que $st(\gamma) = \langle t, t + n \rangle$. Veamos que la computación γ' de Π' asociada a γ (es decir que la computación de Π' comienza con la computación γ de Π) cumple que $st(\gamma') = \langle \hat{t}, \hat{t} + (n + 1), \hat{t} + 2(n + 1), \dots \rangle$.

La idea de cómo funciona el sistema es la siguiente:

1. En primer lugar, en los primeros $t - 1$ pasos se comporta igual que Π.
2. A partir del paso t, se activa el módulo copia spikes, cuya función es contar el tiempo n que pasa entre los dos spikes del sistema Π, y transmitirlo a las neuronas m_1, m_2. Es importante notar que los spikes se transfieren de 2 en 2, esto es porque las reglas en m_1, m_2 solo se activan con un número impar de spikes. Tras los n pasos (y 1 extra) se llega a la configuración donde todas las neuronas están a cero a excepción de m_1, m_2 que tienen $2n + 1$ spikes

3. Es ahora cuando empieza el módulo mover m_1, m_2 a m'_1, m'_2. Este módulo trata de contar los n pasos de computación para en el último pasar una neurona a la neurona de salida, para que así mande un spike al pasar $n+1$ instantes. Mientras se hace esto, el contenido de m_1, m_2 se pasa a m'_1, m'_2 para que esto se pueda repetir indefinidamente y obtener un tren de spikes infinito.

Veámoslo ahora más detenidamente. Como se ha comentado, en los primeros $t-1$ pasos Π y Π' se comportan igual. En el paso t la neurona i_0 emite un spike, que llega hasta las neuronas c_1, c_2, c_3. La neurona c_1 todavía no puede activar la regla. Así, en el paso $t+1$ las neuronas c_3, c_2 se mandan un spike entre ellas y cada una manda spikes a m_1, m_2. De esta forma, en la configuraciçon C_{t+1} se tienen 1 spike en c_2, c_3 (que se volverán a disparar) y 2 spikes en m_1, m_2. De esta forma, en el siguiente paso seguirán solo siendo aplicables las reglas de c_2, c_3, y esto será así hasta el paso $t+n$. En el paso $t+n$ la neurona i_0 emite otro spike, haciendo que en la configuración C_{t+n} se tengan 2 spikes en c_1, c_2, c_3 y $2n$ spikes en m_1, m_2. De esta forma, en el paso $t+n+1$ c_2, c_3 olvidan los spikes que tienen y solo c_1 manda un spike a m_1 y m_2, haciendo que m_1, m_2 tengan una cantidad impar de spikes y puedan aplicar reglas.

De esta forma en el paso $t+n+2(C_{t+n+1} \Rightarrow C_{t+n+2})$ m_1, m_2 mandan spikes a m'_1, m'_2, out, usando la regla $a(aa)^+/aa \to a;0$ todas recibiendo 2 spikes y m_1, m_2 olvidando 2. Nótese que tal y como m_1 y m_2, las reglas de m'_1 y m'_2 solo son aplicables cuando su contenido de spikes sea impar. Como reciben de 2 en 2, ninguna regla es aplicable, ni lo será hasta que no quede solo 1 spike en m_1, m_2. Paralelamente, mientras que m_1, m_2 vayan mandando ambas spikes, la neurona de salida recibirá 2 spikes que olvidará. Esto será así hasta que se llegue a la configuración C_{t+2n+1} (han pasado n pasos). En esta configuración, las neuronas m_1, m_2 contendrán 1 spike cada una, la neurona *out* tendrá 2 spikes que olvidará y las neuronas m'_1, m'_2 tendrán $2n$ spikes. En el paso $t+2n+2$ m_1 manda un spike, mientras que m_2 lo olvida. De esta forma, m'_1, m'_2 tendrán $2n+1$ spikes en la configuración C_{t+2n+2}, y la neurona de salida contendrá 1 solo spike (y estará lista para disparar). De esta forma, en el paso $t+2n+3$ la neurona de salida dispara su primer spike y comienza el módulo mover m'_1, m'_2 a m_1, m_2. Tras n pasos (ojo, contando desde C_{t+2n+2}) llegamos a la configuración C_{t+3n+2}. En el instante siguiente(paso $t+3n+3$), un spike es mandado a la neurona de salida y en el paso $t+3n+4$ la neurona de salida vuelve a disparar.

De esta forma sigue repitiéndose el proceso infinitamente, computando siempre $t+3n+4-(t+2n+3)=n+1$, demostrando así el resultado buscado. ∎

Para la demostración de universalidad siguiente se necesitará también demostrar

que los conjuntos generados por sistemas SN P son cerrados bajo unión finita. Para ello, se usa un resultado auxiliar. Dado un sistema SN P Π se denota por **spin(Π)** al número de spikes máximo que posee una neurona de Π en la configuración inicial.

Lema 3.2. Para cada sistema SN P Π existe un sistema SN P equivalente Π' de forma que $spin(\Pi') = 1$. Además, cumple que solo hay una neurona con spikes. Este sistema Π' constará de $spin(\Pi) + 1$ neuronas más, todas conteniendo solo la regla $a \to a; 0$.

Demostración. Sea Π un sistema SN P arbitrario y construyamos Π' como sigue. Todas las neuronas de Π estarán en Π' con las mismas reglas pero sin spikes. Como se dice en el enunciado, añadimos $spin(\Pi) + 1$ neuronas más a Π' con etiquetas $0, 1, 2, \dots, spin(\Pi)$ (se asume que estas etiquetas no están asignadas) y todas con una única regla $a \to a; 0$. En Π' la única neurona con spikes será la neurona 0, que contendrá un spike. Además, esta estará conectada con todas las neuronas $1, \dots, spin(\Pi)$. Como originalmente todas las neuronas de Π tenían a lo más $spin(\Pi)$ spikes, cada neurona de Π que en la configuración inicial tuviera $k \leq spin(\Pi)$ spikes la conectamos con con sinapsis a k neuronas de $1, \dots, spin(\Pi)$. De esta forma, tras 2 pasos de computación, el sistema Π' se encontrará en una configuración equivalente a la configuración inicial de Π, demostrandose de esta manera que $N_\alpha^\beta(\Pi) = N_\alpha^\beta(\Pi')$ para cualquier α, β posible.

Teorema 3.5. *Si $Q_1, Q_2 \in Spik_\alpha^\beta P_m(rule_k, cons_p, forg_q)$ para algún $m \geq 1, k \geq 2, p \geq 2, q \geq 1$, $\alpha \in \{\omega, all\} \cup \{k, \underline{k} | k \geq 2\}, \beta \in \{h, a, ha\}$ u omitido, entonces $Q_1 \cup Q_2 \in Spik_\alpha^\beta P_{2m+6}(rule_k, cons_p, forg_q)$.*

Demostración. Sean Π_1, Π_2 dos sistemas SN P de forma que $Q_1 \in N_\alpha^\beta(\Pi_1), Q_2 \in N_\alpha^\beta(\Pi_2)$, en la forma normal dada por el lema 3.2. Denotemos por in_1, in_2 a las etiquetas de aquella neurona que contiene un spike en la configuración de salida. El sistema SN P dado en la figura 3.6 computa la unión $Q_1 \cup Q_2$.

Es fácil ver que es así gracias al no determinismo generado por la neurona 2 de la construcción (puede ir hacia un sistema u otro de manera no determinista y empezar a computarlo). El razonamiento es análogo a aquel hecho para elegir las etiquetas del módulo ADD de la demostración del teorema 3.1 de manera no determinista.

Estamos ya en condiciones de demostrar otro caso de universalidad.

Teorema 3.6. *$Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q) = NRE$ para todo $k \geq 2, p \geq 3, q \geq 3$ y $\alpha \in \{\omega\} \cup \{k | k \geq 3\}, \beta = a$ o β omitido.*

Demostración. Como siempre, gracias a la tesis de Church-Turing basta probar que $NRE \subset Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$. Sea $Q \in NRE$ y denotemos por $Q' = \{n -$

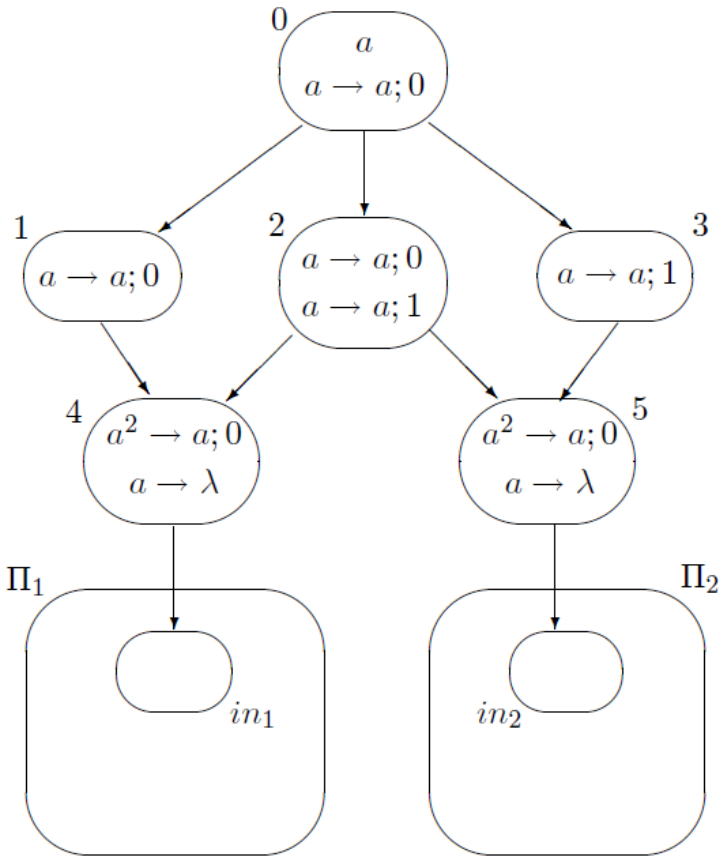


Figura 3.6: SN P que computa la unión

$1|n \in Q\} \in NRE$. Por el teorema 3.1 se tiene que existe un sistema SN P Π cumpliendo $N_{\underline{2}}^h(\Pi) = Q'$. Aplicando ahora el lema 3.1 al sistema Π se obtiene un sistema Π' cumpliendo $N_\omega(\Pi') = N_k(\Pi') = \{n+1|n \in N_{\underline{2}}^h(\Pi) = Q'\} = Q - \{1\}$ para todo $k \geq 2$.

Caso 1: $1 \notin Q$: Entonces se tiene que $Q = Q - \{1\} \in Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$ demostrando así el teorema.

Caso 2: $1 \in Q$: Para ver que Q está entonces en $Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$ basta construir un sistema SN P Π'' que compute el conjunto unitario $\{1\}$ y aplicar el teorema 3.5 (que nos dice que podemos hacer la unión finita de los dos sistemas) a Π' y a Π'', obteniendo así que el conjunto $Q = \{1\} \cup Q - \{1\}$ es computado por la unión de los sistemas Π'', Π' y por tanto $Q \in Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$.

Este sistema SN P Π'' que computa el conjunto unitario $\{1\}$viene dado por dos neuronas conectadas entre sí ($syn = \{(1,2),(2,1)\}$ cumpliendo $R_i = \{a \to a; 0\}$ para $i = 1, 2$ con una configuración inicial dada por:

$$(\sigma_1 = (1,0), \sigma_2 = (1,0))$$

Donde es conveniente recordar que el 1 representa el número de spikes en la neurona σ_i y el 0 representa el tiempo que queda para que la neurona esté abierta.

Es fácil comprobar que este sistema SN P computa el conjunto unitario $\{1\}$, completando la demostración del teorema.

A partir de las ideas de las pruebas anteriores se puede dar otro resultado de universalidad. Si combinamos la idea del teorema 3.4 (del mandar spikes para "rebosar" neuronas y que no vuelvan a mandar spikes) con la construcción del lema 3.1 se tiene el siguiente resultado.

Teorema 3.7. *$Spik^h_\alpha P_*(rule_h, cons_p, forg_q) = NRE$ para todo $h \geq 2, p \geq \max(3,k)$, $q \geq 3$ y $\alpha \in \{k, \underline{k} | k \geq 2\}$.*

Demostración. Basta añadir a la construcción del lema 3.1 un módulo como en el teorema 3.4 en el que mande 3 spikes a las neuronas m_4, m'_4 tras k spikes de la neurona de salida para no dejar al sistema volver a disparar.

Un último lema que nos permite pasar de computaciones infinitas a computaciones finitas será necesario para demostrar el último teorema de universalidad restante.

Lema 3.3. Dado un sistema SN P Π se puede construir un sistema SN P Π_k de forma que:

1. Para cada compputación γ de Π con $st(\gamma) = \langle t_1, t_2 \dots, t_j \rangle, j \leq k$, existe una computación γ' de Π_k tal que $st(\gamma') = \langle t_1 + 2, t_2 + 2, \dots, t_j + 2 \rangle$.
2. Para cada computación γ de Π con $st(\gamma) = \langle t_1, t_2, \dots, t_k, \dots \rangle$; existe una computación γ' de Π_k tal que $st(\gamma') = \langle t_1 + 2, t_2 + 2, \dots, t_k + 2 \rangle$
3. Cada computación γ' de Π_k cumple una de las siguientes:
   a) No manda nunca spikes
   b) Se corresponde con una computación γ de Π con $st(\gamma) = \langle t_1, t_2 \dots, t_j \rangle, j \leq k$ cumpliendo $st(\gamma') = \langle t_1 + 2, t_2 + 2, \dots, t_j + 2 \rangle$.
   c) Se corresponde con una computación γ de Π con $st(\gamma) = \langle t_1, t_2, \dots, t_k, \dots \rangle$; cumpliendo $st(\gamma') = \langle t_1 + 2, t_2 + 2, \dots, t_k + 2 \rangle$.

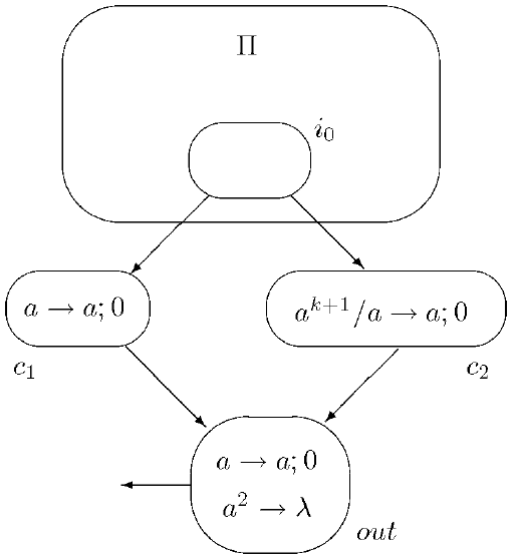


Figura 3.7: Figura asociada a la demostración del lema 3.3

Demostración. Dado un sistema SN P Π construimos el sistema SN P Π_k como se indica en la figura 3.7.

El sistema funciona de la siguiente manera. En primer lugar se simula una computación de Π. Así, si Π no para entonces mi nuevo sistema Π_k tampoco para. Supongamos ahora que el sistema Π emite un spike en el instante t_1. Entonces, en mi sistema Π_k, la neurona de salida estará conectada a otras dos neuronas, $c_1 c_2$. La neurona c_2 actuará como contador de spikes, lo veremos ahora con más detalle. Si nos centramos en la neurona c_1, esta dispara un spike nada más recibe el spike de la neurona i_0, provocando un spike en el sistema Π_k en el instante $t_1 + 2$. Esto es así, siempre y cuando la neurona c_2 no dispare.

Centrémonos ahora en cómo funciona la neurona c_2. Esta neurona disparará cuando contenga $k + 1$ spikes. Como solo recibe spikes de la neurona de salida i_0 de Π, solo disparará cuando el sistema Π haya emitido $k + 1$ spikes, provocando que en el siguiente paso la neurona de salida reciba 2 spikes (de c_1 y c_2 y por tanto olvide los spikes. Nótese además que c_2 olvida solo 1 spike al disparar, quedándose con a^k spikes en su interior. Así, al recibir otro spike, volverá a disparar, provocando que la neurona de salida del sistema Π_k no dispare nunca más.

Por lo tanto, el sistema Π_k funcionará como sigue:

Si el sistema Π no dispara entonces el sistema Π_k no dispara.

Si el sistema Π dispara j veces con $j \leq k$ entonces el sistema Π_k disparará j veces con un desfase de 2 (el tiempo de pasar por c_1).

Si el sistema Π dispara más de k veces entonces el sistema Π_k disparará k veces (igual con desfase de 2) y a partir del spike $k+1$, las neuronas c_1, c_2 dispararán siempre que i_0 dispare, haciendo que *out* olvide los dos spikes y por tanto que Π_k no dispare.

De estas afirmaciones se deduce la demostración del lema.

Con esto, se puede demostrar el último resultado de universalidad.

Teorema 3.8. *$Spik^{\beta}_{\underline{k}}P_*(rule_k, cons_p, forg_q) = NRE$ para todo $k \geq 2, p \geq 3, q \geq 3$ y $\beta = a$ o β omitido.*

Demostración. Sea $Q \in NRE$. Análogamente a la construcción hecha en el Teorema 4.1, construimos un sistema SN P Π conteniendo un tren de spike es infinito, cumpliendo las condiciones del lema 3.1 (con spikes repetidos en intervalos idénticos) y tal que $N^{\beta}_{\omega}(\Pi) = Q$, para $\beta = a$ u omitido. Aplicamos ahora a este sistema la construcción del lema 3.3. De esta forma, se obtiene un sistema Π' que manda exactamente k spikes al exterior (pues Π manda infinitos spikes). Así, $Q = N^{\beta}_{\omega}(\Pi) = N^{\beta}_{\underline{k}}(\Pi') \in Spik^{\beta}_{\underline{k}}P_*(rule_k, cons_p, forg_q)$.

A modo resumen de la sección, puesto que se han visto muchos resultados de universalidad, para finalizar y aclarar conceptos se da una tabla recogiendo todos los resultados demostrados en los distintos teoremas.

α \ β	omitido	h	a	ha
2	Teorema 3.2	Teorema 3.2	Teorema 3.2	Teorema 3.2
$\underline{2}$	Teorema 3.2	Teorema 3.2	Teorema 3.2	Teorema 3.2
k	Teorema 4.1	Teorema 3.7	Teorema 4.1	Teorema 3.5
$\underline{k}$	Teorema 3.8	Teorema 3.7	Teorema 3.8	Teorema 3.5
ω	Teorema 4.1	—	Teorema 3.3	—
all	Teorema 3.2	Teorema 3.2	Teorema 3.2	Teorema 3.2

3.2 Universalidad para sistemas SN P con spikes acotados

En esta sección se restringe el número máximo de spikes que puede contener una neurona. Esto tiene sentido, ya que en realidad las neuronas biológicas disparan a partir de un cierto umbral, siempre de igual manera, sin importar la cantidad de potencial de membrana que tuviera, reseteándose siempre al mismo punto.

Con esta consideración se verá que la capacidad computacional de los sistemas SN P baja, pudiendo generar exactamente los conjuntos semilineales de números naturales ($NREG$).

Se comienza con algunas definiciones necesarias para el desarrollo de la teoría.

Definición 3.1. *Dado un sistema SN P* Π *se dice que* γ *es una* ***computación s-acotada*** *si el contenido de las neuronas de* Π *a lo largo de la computación es a lo más* s *spikes.*

Al conjunto de computaciones s-acotadas de Π *se le denota* $COM_s(\Pi)$ *y al de las computaciones s-acotadas de parada se le denota* $HCOM_s(\Pi)$.

Si una computación llega a una configuración en la que alguna neurona contiene más de s *spikes, entonces esta computación para y no produce ningua salida.*

Definición 3.2. *Dado un sistema* Π *se define el conjunto* $N_\alpha^\beta(\Pi, s)$ *análogamente a la definición de* $N_\alpha^\beta(\Pi)$ *pero tomando las computaciones en* $COM_s(\Pi)$. *Por ejemplo, si:*

$$N_k(\Pi) = \{n | n = t_i - t_{i-1}, 2 \leq i \leq k, \gamma \in COM(\Pi),\ y\ st(\gamma)\ con\ al\ menos\ k\ spikes\}$$

Entonces

$$N_k(\Pi, s) = \{n | n = t_i - t_{i-1}, 2 \leq i \leq k, \gamma \in COM_s(\Pi),\ y\ st(\gamma)\ con\ al\ menos\ k\ spikes\}$$

A $N_\alpha^\beta(\Pi, s)$ *se le denomina el conjunto de computaciones s-acotadas.*

Definición 3.3. *Se denota por* $Spik_\alpha^\beta P_m(rule_k, cons_q, forg_p, bound_s)$ *a las familias* $Spik_\alpha^\beta P_m(rule_k, cons_q, forg_p)$ *con la restricción adicional de que para un sistema SN P* Π *se consideran los conjuntos de números* $N_\alpha^\beta(\Pi, s)$, *en vez de* $N_\alpha^\beta(\Pi)$.

Si se escribe $bound_*$ *quiere decir que consideramos aquellos sistemas SN P con un número acotado de spikes en cualquier neurona, pero esta cota puede variar al tomar varios sistemas* Π *(no es uniforme).*

El objetivo de la sección será por tanto demostrar que

$$Spik_{\alpha}^{\beta}P_m(rule_k, cons_q, forg_p, bound_s) = NREG$$

para ciertos valores de k, q, p, s. Comenzamos con la primera inclusión,

$$Spik_{\alpha}^{\beta}P_*(rule_*, cons_*, forg_*, bound_*) \subseteq NREG$$

Para demostrar esta inclusión se usará la siguiente técnica.

Como ya se dijo en el capítulo 1, se tiene que $NREG$ se corresponde con la familia de conjuntos de longitudes de palabras de lenguajes regulares. Ahora bien, recuérdese también que un lenguaje es regular si y solo si existe una gramática regular que lo genera. El objetivo será por tanto dado un sistema SN P Π construir una gramática regular G_{α}^{β} de forma que se cumpla que $\{|v| : v \in L(G_{\alpha}^{\beta})\} = N_{\alpha}^{\beta}(\Pi, s)$. De esta forma $N_{\alpha}^{\beta}(\Pi, s)$ será un conjunto de longitudes de palabras de un lenguaje regular y por tanto se tendrá que $N_{\alpha}^{\beta}(\Pi, s) \in NREG$. Como el sistema Π es arbitrario, se concluye el resultado.

Es necesario poder constrir las gramáticas asociadas a cada uno de los conjuntos $N_{\alpha}^{\beta}(\Pi, s)$ de manera efectiva. Para poder afirmar esto, serán importantes los resultados que vienen a continuación.

Teorema 3.9. *Dado un sistema SN P Π se tiene que el número de configuraciones alcanzables por todas las computaciones acotadas $\gamma \in COM_s(\Pi)$ es finito.*

Demostración. Sea $\Pi = (\{a\}, \sigma_1, \dots, \sigma_m, syn, i_0)$ un sistema SN P y $\gamma \in COM_s(\Pi)$. Se tiene que el número de neuronas de Π, σ_i, es finito. Recuérdese que las configuraciones alcanzables por cualquier $\gamma \in COM_s(\Pi)$ deben cumplir que el contenido de las neuronas en la configuración debe ser menor o igual que s. Denotemos por C al conjunto de configuraciones alcanzables en Π por cualquier computación $\gamma \in COM_s(\Pi)$. Veamos que el cardinal de C, $\#(C)$, es finito.

Considérese el sistema Π' construido a partir de Π como sigue:

El sistema $\Pi' = (\{a\}, \sigma_1', \dots, \sigma_m', syn, i_0)$ con $\sigma_i' = (n_i, R_i')$ para cada $\sigma_i = (n_i, R_i)$ neurona de Π, construyendo el conjunto de reglas R_i' a partir de R_i como:

Si existe $r \in R_i$ de la forma $E/a^c \to a; d$, añado a R_i' el conjunto de reglas $\{a^k/a^c \to a; d\}$ para todo k cumpliendo $a^k \in \{a^1, a^2 \dots, a^s\} \cap L(E)$.

Si $r \in R_i$ es de la forma $a^j \to \lambda$, se añade a R_i' si y solo si $j \leq s$.

Nótese que como el conjunto de reglas R_i de cada neurona de Π es finito, el conjunto R'_i se puede construir de manera efectiva. Así construido, se tiene que el conjunto de configuraciones alcanzables en Π', C', por cualquier computación $\gamma \in COM(\Pi')$ es finito.

Efectivamente, como en el conjunto de reglas de cada una de las neuronas del sistema Π' puede ser aplicable (por construcción) solo cuando el contenido de spikes de las neuronas es menor o igual a s y además la cantidad de neuronas de Π' es finita, para obtener todas las configuraciones alcanzables basta considerar todas las combinaciones de transiciones desde la configuración inicial, que serán finitas. Veamos por último que $C \subseteq C'$.

Sea $C \in C$. Por definición de configuración alcanzable, se tiene que existe una computación $\gamma = C_0 \Rightarrow_{T1} \dots C \Rightarrow_{T_n} \cdots \in COM_s(\Pi)$.

Veamos que $C \in C'$. Para ello, basta construir una computación de Π' donde se alcance C. Sea γ' la computación $C_0 \Rightarrow_{T'_1} \dots, C \Rightarrow_{T'_n}$, construida de la siguiente manera:

Para cada transición T_i de la computación γ, se aplicarán $l \leq m$ reglas, denotemoslas por $r_1, \dots r_l$. Como $\gamma \in COM_S(\Pi)$, cada regla será aplicada con a lo más s spikes. Por construcción de Π', para cada $(r_1, \dots, r_l)$ en Π existirá unas reglas $(r'_1, \dots, r'_l)$ en Π' que llevan a la misma configuración (por ejemplo si la regla aplicada es $r_i = E/a^c \rightarrow a; d$, el contenido de la neurona i es $a^k \in L(E)$, entonces en Π' existe una regla $r'_i = a^k/a^c \rightarrow a; d$ por construcción). De esta forma, se construye una computación de Π' donde C es alcanzable, y por tanto se tiene que $C \in C'$.

Por tanto, $C \subseteq C'$. Como C' es finito, se tiene que C es finito. ∎

La construcción hecha en este teorema nos permite dar un grafo asociado a todas las posibles computaciones s-acotadas de un sistema SN P Π dado, cumpliendo que el número de nodos y de aristas es finito.

Corolario 3.1. Dado un sistema SN P Π existe un grafo finito asociado a todas las posibles computaciones s-acotadas del sistema.

Demostración. Considérese el sistema Π' construido en el teorema anterior. Se ha demostrado que el número de configuraciones y de transiciones de este es finito. Si denotamos por C' al conjunto de todas las configuraciones alcanzables en Π' por computaciones $\gamma' \in COM(\Pi')$, considérese el grafo mixto (es dirigido pero también tiene bucles) $G = (V, A)$ definido como:

$$V = \{C \in C'\}$$

$A = \{(C, C')$ tal que existe una computación$\gamma \in COM(\Pi')$ con una transición $C \Rightarrow C'\}$

Nótese que así definido podrían haber configuraciones con neuronas con más de s spikes. Construimos el grafo asociado a Π a partir del grafo G haciendo los siguientes cambios:

1. Eliminamos aquellos nodos que correspondan a configuraciones con alguna neurona con más de s spikes.
   Nota: Una vez eliminados, puede haber configuraciones que no sean alcanzables desde C_0 (aquellas que tenían que pasar por una configuración con más de s spikes para llegar).
2. Se eliminan aquellos nodos C para los cuales no existe un camino desde C_0.

Es importante destacar que al ser un grafo finito esta construcción es efectiva.

A partir de la demostración del teorema anterior y las observaciones hechas se deduce que los nodos y caminos posibles en este grafo se corresponden a configuraciones y transiciones posibles en Π para computaciones s-acotadas.

Observación 3.1. Obsérvese que habiendo construido el grafo de esta manera, las computaciones de parada se corresponderán con nodos de los que no sale ninguna arista.

Una vez visto que el grafo asociado a las computaciones s-acotadas de un sistema SN P es finito podemos pasar a dar las construcciones de las gramáticas que adelantábamos.

Teorema 3.10. *$Spik^{\beta}_{\alpha}P_*(rule_*, cons_*, forg_*, bound_*) \subseteq NREG$ para todo $\alpha \in \{k, \underline{k} | k \geq 2\} \cup \{\omega, all\}, \beta \in \{a, ha, h\}$ u omitido.*

Demostración. Se comienza con la prueba de $Spik_{\underline{2}}P_*(rule_*, cons_*, forg_*, bound_*) \subseteq NREG$.

Caso $\alpha = \underline{2}, \beta = h$ u omitido:

Como ya se dijo a principios de la sección la técnica de la demostración será la siguiente:

Recuérdese que se tiene que $NREG$ se corresponde con la familia de conjuntos de longitudes de palabras de lenguajes regulares. Ahora bien, un lenguaje es regular si y solo si existe una gramática regular que lo genera. El objetivo será por tanto dado un sistema SN P Π construir una gramática regular G de forma que se cumpla que $\{|v| : v \in L(G_{\underline{2}})\} = N_{\underline{2}}(\Pi, s)$. De esta forma $N_{\underline{2}}(\Pi, s)$ será un conjunto de longitudes de palabras de un lenguaje regular y por tanto se tendrá que $N_{\underline{2}}(\Pi, s) \in NREG$. Como el sistema Π es arbitrario, se concluye el resultado.

Sea entonces Π un sistema SN P con un número acotado de spikes (s) en cada neurona. Por el Teorema 3.9 el número de configuraciones alcanzables por computaciones s-acotadas del sistema Π será finito. Denotemos por C al conjunto de todas las configuraciones alcanzables de Π por computaciones s-acotadas, y por C_0 a la configuración inicial. Sea $G_{\underline{2}} = (N, \{a\}, (C_0, 0), \Phi)$ una gramática donde $N = C \times \{0, 1, 2\}$ es el conjunto de símbolos no terminales, $\{a\}$ es el alfabeto, $(C_0, 0)$ es el símbolo inicial, y Φ es el conjunto de procesos dado por:

1. $(C, 0) \to (C', 0)$ por cada $C, C' \in C$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida no dispara ningún spike.
2. $(C, 0) \to (C', 1)$ por cada $C, C' \in C$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
3. $(C, 1) \to a(C', 1)$ por cada $C, C' \in C$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida no dispara ningún spike.
4. $(C, 1) \to a(C', 2)$ por cada $C, C' \in C$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
5. $(C, 2) \to \lambda$ por cada $C \in C$ tal que existe una computación de parada $C \Rightarrow C_i \Rightarrow C_i + 1 \Rightarrow \cdots \Rightarrow C_m$ en Π en la cual la neurona de salida no dispara o por cada C tal que existe una computación infinita empezando en la configuración C en la cual la neurona de salida no dispara.

Es importante destacar que la construcción de la gramática es efectiva porque todas las reglas de producción se pueden decidir algorítmicamente. Esto se puede demostrar fácilmente gracias al grafo construido en el corolario 3.1, denotado aquí por $G = (V, A)$.

Las reglas 1,2,3,4 se pueden decidir simplemente mirando las aristas del grafo y las transiciones asociadas. Veamos la regla 5.

Dado el símbolo no terminal $(C, 2)$ el problema de decidir si se añade el proceso $(C, 2) \to \lambda$ se puede decidir algorítmicamente.

Considérese el siguiente algoritmo, que tiene como entrada la configuración C.

1. Verificar si existe un camino desde el nodo C hasta una configuración de parada en el grafo G donde no se emite ningún spike. Si es que sí, añadir la producción.
2. Verificar si existe un ciclo en el grafo desde una cierta configuración C' (un bucle de transiciones $C' \Rightarrow \ldots, \Rightarrow C'$), donde no se emiten spikes, y si existe un camino desde C a C' en el cual no se emiten spikes. Si es que sí, añadir la producción.
3. En caso contrario no se añade.

Es interesante destacar que el paso 1 se corresponde con la computación de parada y el paso dos con la computación infinita. El paso 2 se corresponde con la computación infinita porque si existe tal ciclo en el grafo, por el no determinismo del sistema, siempre existirá una computación de Π que vaya de C a C' y que se quede dando vueltas constantemente en el ciclo, produciendo una computación infinita sin más spikes.

El hecho de poder saber si existen estos caminos o no de forma algorítmica se sigue de que el grafo es finito (un algoritmo de fuerza bruta valdría y pararía siempre).

Se tiene por tanto que así construida, se tiene que G es una gramática regular y que $\{|v| : v \in L(G_{\underline{2}})\} = N_2^{\beta}(\Pi)$ concluyendo que $N_2^{\beta}(\Pi)$ es semilineal.

Para el caso $\beta = h$ simplemente consideramos una gramática $G_{\underline{2}}^h$ con las mismas reglas de producción que $G_{\underline{2}}$, solo que la regla 5, $(C, 2) \to \lambda$, solo se añade si existe una computación de parada $C \Rightarrow C_i \Rightarrow C_i + 1 \Rightarrow \cdots \Rightarrow C_m$ en Π en la cual la neurona de salida no dispara.

Veamos un ejemplo de una computación de Π para ver cómo funciona. Supongamos que tenemos un sistema SN P Π y una computación γ de este tal que $st(\gamma) = \langle t, t{+}n\rangle$. Veamos que la longitud de la palabra generada por la derivación de ϕ asociada a la computación γ (por construcción de la gramática toda derivación, tiene asociado una computación de Π y viceversa), $(C_0, 0) \Rightarrow_{\phi}^{*} u$, cumple que $|u| = n$. Esto nos da una idea de por qué se tiene la igualdad $\{|v| : v \in L(G_{\underline{2}})\} = N_2^{\beta}(\Pi)$. Si denotamos por C_i a la configuración de Π tras aplicar el paso i de la computación γ, la derivación asociada a la computación γ será:

$$(C_0, 0) \to (C_1, 0) \to \cdots \to (C_t, 0) \to (C_{t+1}, 1) \to a(C_{t+2}, 1) \to a^2(C_{t+3}, 1) \to \cdots \to$$

$$\rightarrow a^{n-1}(C_{t+n}, 1) \rightarrow a^n(C_{t+n+1}, 2) \rightarrow a^n$$

Que claramente cumple que $|a^n| = n$.

Sigamos viendo el resto de casos. La técnica será similiar, se construirá una gramática G_α^β parecida a la anterior, para cada valor de α, β. Para la justificación de la construcción efectiva es clave usar el grafo construido en el Corolario 3.1.

Caso: $\alpha = \omega$ y β omitido:

Sea $G_\omega = (N, \{a\}, (C_0, 0), \Phi)$ una gramática donde $N = C \times \{0, 1, 2\}$ es el conjunto de símbolos no terminales, $\{a\}$ es el alfabeto, $(C_0, 0)$ es el símbolo inicial, y Φ es el conjunto de procesos dado por:

1. $(C_0, 0) \rightarrow (C, 0)$ por cada $C \in \mathcal{C}$ tal que existe una transición $C \implies C'$ en Π en la que la neurona de salida dispara un spike.
2. $(C, 0) \rightarrow (C', 1)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \implies C'$ en Π en la que la neurona de salida dispara un spike.
3. $(C, 1) \rightarrow a(C', 1)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \implies C'$ en Π en la que la neurona de salida no dispara ningún spike.
4. $(C, 1) \rightarrow a(C', 2)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \implies C'$ en Π en la que la neurona de salida dispara un spike.
5. $(C, 2) \rightarrow \lambda$ por cada $C \in \mathcal{C}$ por cada C tal que existe una computación infinita empezando en la configuración C en la cual la neurona de salida dispara infinitas veces.

Nótese que como $\mathcal{C}$ es el conjunto de todas las configuraciones alcanzables, entonces sabemos que para la regla 1 siempre existirá una computación s-acotada que va desde C_0 a C.

Además, nótese también que la regla 5 es dedicible algorítmicamente. Para decidirlo, basta ver si en el grafo G asociado a las computaciones s-acotadas de Π tiene un ciclo desde una cierta configuración C' (un bucle de transiciones $C' \Rightarrow \ldots, \Rightarrow C'$), donde se emite al menos un spike, y si existe un camino desde C a C'.

En caso de que pase esto, por el no determinismo del sistema, se puede construir una computación s-acotada con infinitos spikes, simplemente generando la computación a partir de la derivación de C_0 a C, tomar el camino C a C' y repetir el bucle

de transiciones infinitamente. De esta forma el intervalo generado por la gramática se incluiría en el conjunto $N_\omega(\Pi, s)$.

Es importante destacar que la condición de que exista un ciclo en el grafo en el que se emite al menos un spike es necesaria y suficiente para construir una computación s-acotada con infinitos spikes. Que es suficiente se justifica con el parrafo anterior.

Veamos que es necesaria. Por reducción al absurdo. Supongamos que existe un sistema SN P Π tal que su grafo no contiene ciclos donde se emiten spikes pero existe una computación s-acotada de Π que emite infinitos spikes.

Nótese que al ser C finito, las computaciones infinitas deben contener necesariamente bucles de transiciones. Nótese además que el numero de configuraciones desde las que se pueden emitir spikes es finito, digamos k. Construyamos un ciclo donde se emite un spike a partir de la computación con infinitos spikes.

Considérese la computación γ_s s-acotada con infinitos spikes, y sea t un instante en el que se emite un spike en una configuración, $C_1(C_1 \Rightarrow_t C')$. Como existen infinitos spikes en γ_s, considérese las configuraciones $C_2, \dots C_{k+1}$ en las que se emiten los siguientes k spikes. Como hay solo k configuraciones desde donde se pueden emitir spikes, entonces se tiene que existen $i, j \in \{1, \dots k+1\}$ tal que $C_j = C_i$. Si se considera el ciclo $C_i \Rightarrow_T C_i$, se tiene que en este se emite al menos un spike, llegando así a un absurdo.

Se concluye por tanto que la construcción de la gramática G_ω es efectiva y, análogamente al caso anterior, se tiene que $N_\omega(\Pi, s) \in NREG$.

En los siguientes casos daremos solo las gramáticas y se deja la deduccición de que los conjuntos pertenecen a $NREG$ como ejercicio al lector (de forma análoga al caso $\alpha = \underline{2}$)

Caso $\alpha = \omega$ y $\beta = a$

Análogamente a antes se considera $G^a_\omega = (N, \{a\}, (C_0, 0), \Phi)$ una gramática donde $N = C \times \{0, 1, 2\}$ es el conjunto de símbolos no terminales, $\{a\}$ es el alfabeto, $(C_0, 0)$ es el símbolo inicial, y Φ es el conjunto de procesos dado por los procesos de la gramática G_ω, a excepción de cambiar la regla número 1 por:

$(C_0, 0) \to (C, 0)$ por cada $C \in C$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike y tal que existe una computación parcial $C_0 \Rightarrow_\gamma C$ tal que $2k$ ($k \geq 0, k \in \mathbb{N}$) spikes han sido emitidos por la neurona de salida

durante γ.

Esto se puede determinar de manera efectiva a través del grafo, simplemente por algoritmo de fuerza bruta (al ser finito) y búsqueda de bucles de transiciones que emiten un número par de spikes (o un spike).

Caso $\alpha = k$ y β omitido

Sea $G_k = (N, \{a\}, (C_0, 0), \Phi)$ una gramática donde $N = C \times \{0, 1, 2\}$ es el conjunto de símbolos no terminales, $\{a\}$ es el alfabeto, $(C_0, 0)$ es el símbolo inicial, y Φ es el conjunto de procesos dado por:

1. $(C, 0, 0) \to (C, 0, l); (l = 0, 1, \dots k-2)$, por cada $C \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike y tal que existe una computación ("parcial") γ $C_0 \Rightarrow_\gamma C$ donde se han emitido l spikes.
2. $(C, 0, l) \to (C', 1, l+1); (l = 0, 1, \dots k-2)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
3. $(C, 1, l) \to a(C', 1, l); (l = 1, \dots k-1)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida no dispara ningún spike.
4. $(C, 1, l) \to a(C', 2, l+1); (l = 1, \dots k-1)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
5. $(C, 2, l) \to \lambda; (l = 2, \dots k)$ por cada $C \in \mathcal{C}$ tal que existe una computación de parada $C \Rightarrow C_i \Rightarrow C_i + 1 \Rightarrow \cdots \Rightarrow C_m$ en Π en la cual la neurona de salida dispara al menos $k - l$ spikes más o por cada C tal que existe una computación infinita empezando en la configuración C en la cual la neurona de salida dispara al menos $k - l$ spikes más.

En esta gramática se añade un tercer argumento para llevar el control de cuántos spikes se han emitido y aceptar aquellos que solo emitan k o más de k spikes.

Haciendo uso del grafo del Corolario 3.1 y usando apropiadamente los bucles de transición se llega a que la gramática construida es efectiva. (ojo que ahora hay que buscar bucles de transición que emitan m spikes con $m = 1$ o m un divisor de l, si es la regla 1, o m divisor de $k - l$, si es la regla 5.

Para los casos $\beta = a, h, ha$ basta aplicar las técnicas usadas en gramáticas anteriores en esta. Como idea, en $\beta = a$ se razona análogamente a $\alpha = \omega, \beta = a$, para $\beta = h$,

se razona igual que para $\alpha = \underline{2}, \beta = h$, y para $\beta = ha$ se mezclan las ideas de ambas gramáticas.

Para el caso $\alpha = \underline{k}$ y cualquier β basta tomar la gramática G_k^β cambiando la regla 5 para que en vez de considerar al menos $k - l$ spikes se considera para justo $k - l$ spikes.

De esta manera solo nos queda el caso $\alpha = all$. Se da la gramática asociada al caso β omitido y se deja al lector realizar los cambios necesarios basados en previas construcciones.

Caso $\alpha = all, \beta$ omitido:

Sea $G_{all} = (N, \{a\}, (C_0, 0), \Phi)$ una gramática donde $N = C \times \{0, 1, 2\}$ es el conjunto de símbolos no terminales, $\{a\}$ es el alfabeto, $(C_0, 0)$ es el símbolo inicial, y Φ es el conjunto de procesos dado por:

1. $(C_0, 0) \to (C, 0)$ por cada $C \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
2. $(C, 0) \to (C', 1)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
3. $(C, 1) \to a(C', 1)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida no dispara ningún spike.
4. $(C, 1) \to a(C', 2)$ por cada $C, C' \in \mathcal{C}$ tal que existe una transición $C \Rightarrow C'$ en Π en la que la neurona de salida dispara un spike.
5. $(C, 2) \to \lambda$ por cada $C \in \mathcal{C}$ siempre.

Nótese que como en $\alpha = all$ contamos todos los spikes de todas las computaciones no se añade ninguna restricción a la regla 5.

Tras este teorema nos queda probar que

$$NREG \subseteq Spik_\alpha^\beta P_*(rule_*, cons_q, forg_p, bound_*)$$

para $\alpha \in \{k, \underline{k} | k \geq 2\} \cup \{\omega, all\}, \beta \in \{a, ha, h\}$ u omitido. Para ver esta inclusión, basta ver que los sistemas SN P con spikes acotados son capaces de generar los conjuntos semilineales. Ahora bien, se tiene que cualquier conjunto semilineal de números es la unión de un conjunto finito con un número finito de progresiones aritméticas.

Además, un conjunto finito no es más que la unión de conjuntos unitarios. Así, gracias al teorema 3.5 que nos dice que podemos hacer la unión de dos sistemas SN P, basta demostrar que los conjuntos unitarios ($\{n\} n \geq 1$) y que las progresiones aritméticas ($\{ni | i \geq 1\}, \{r + ni | i \geq 1\}, r \geq 1$) son computables por nuestros sistemas SN P.

Para los sistemas SN P con $\beta \neq h, ha$, $\alpha \neq \underline{k}$ (es decir aquellos que no tienen por qué parar y $\underline{k}$), se dará una prueba constructiva (se construirá un sistema SN P que los genere). Tras esta construcción, se da un lema que nos permitirá pasar de las computaciones infinitas de estos, a computaciones con exactamente k spikes, terminando de demostrar la inclusión y por tanto el resultado buscado en la sección.

Teorema 3.11. *$\{n\} \in Spik^{\beta}_{\alpha}P_{14}(rule_2, cons_3, forg_2, bound_{2n+1})$ para todo $n \geq 1$ y $\alpha \in \{k | k \geq 2\} \cup \{\omega, all\}, \beta = a$ u omitido.*

Demostración. Se tiene que $\{n\} \in Spik^{\beta}_{\underline{2}}P_1(rule_2, cons_1, forg_0, bound_2)$, pues basta tomar el sistema SN P con una única neurona(la propia neurona de salida), conteniendo 2 spikes en el instante inicial, y con el conjunto de reglas $R = \{a^2/a \rightarrow a; 0, a \rightarrow a; n - 1\}$. De esta forma se manda un spike en el instante 1 y otro spike en el instante $2 + n - 1$, computando así n. Aplicando ahora el lema 3.1, construimos un tren de spikes infinito computando $n + 1$. de esta forma, se tiene que $\{n+1\} \in Spik^{\beta}_{\alpha}P_{14}(rule_2, cons_3, forg_2, bound_{2n+1})$. Para el caso $n = 1$, ya se vio en el teorema 4.1 un sistema que lo computaba.

Teorema 3.12. *$\{ni | i \geq 1\} \in Spik^{\beta}_{\alpha}P_m(rule_k, cons_q, forg_p, bound_s)$ para todo $\alpha \in \{k | k \geq 2\} \cup \{\omega, all\}, \beta = a$ u omitido.*

Demostración. Construyamos un sistema SN P que compute ese conjunto para todo α, β como en el enunciado. Considérese el sistema SN P dado por la figura 3.8

El sistema funciona de la siguiente manera. La neurona de salida dispara en el instante 1. Tras ese disparo, un spike llega a la neurona 1. Esta dispara y propaga el spike en tardando $n - 1$ instantes en llegar a las neuronas 0 y $n - 1$. Como ya se ha visto en la construcción de otros sistemas en la memoria, el comportamiento de Π dependerá de la regla que se escoja en la neurona 0 de manera no determinista. Si se escoge la regla $a \rightarrow a; 1$, entonces los spikes de las neuronas 0 y $n - 1$ llegarán con desfase de 1 a la neurona de salida, provocando que sean olvidadas. A la vez, la neurona n manda un spike a la neurona 1 volviendo a empezar el proceso .Si se escoge la regla $a \rightarrow a; 0$, entonces 2 spikes llegan a la neurona de salida, haciendo que esta dispare (devolviendo $n * i$, dependiendo de las veces que se aplique la regla $a \rightarrow a; 1$ antes). Tras el disparo de la neurona de salida, 2 spikes llegan a la neurona 1 que hace que vuelva a empezar.

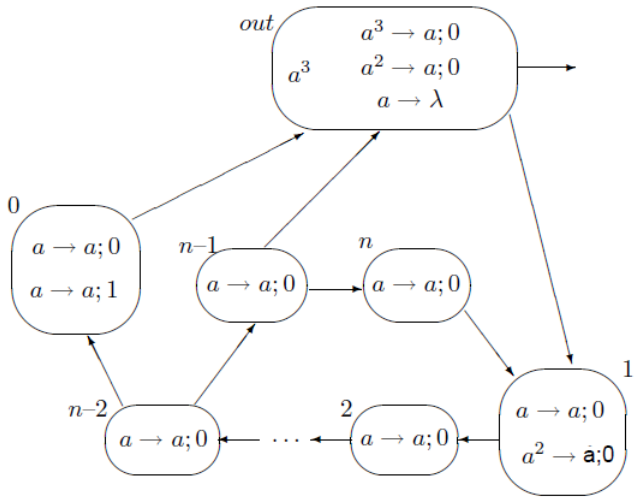


Figura 3.8: Representación del sistema SN P construido para la demostración del teorema 3.12

De esta forma, está claro que el conjunto computado por el sistema Π de la figura es $\{ni|i \geq 1\}$ y que la computación es infinita.

Teorema 3.13. $\{r + 2i|i \geq 1\} \in Spik^{\beta}_{\alpha}P_{r+4}(rule_2, cons_3, forg_2, bound_3)$ *con* $r \geq 2$ *y* $\alpha \in \{k|k \geq 2\} \cup \{\omega, all\}, \beta = a$ *u omitido.*

Demostración. La construcción de la prueba se ilustra con la figura 3.9 y su funcionamiento es análogo al ejemplo del final del capítulo 3, con la única diferencia de que una vez que la neurona 6 dispara pasan r instantes hasta que el spike llega a la neurona 1, y por eso es $r + 2i$ en vez de $1 + 2i$ como en el ejemplo.

Haciendo una variación de los teoremas anteriores podemos llegar a progresiones aritméticas de la forma $\{r + ni|i \geq 1\}$.

Teorema 3.14. $\{r + ni|i \geq 1\} \in Spik^{\beta}_{\alpha}P_{r+n+2}(rule_3, cons_3, forg_4, bound_3)$ *con* $r \geq 1, n \geq 3$ *y* $\alpha \in \{k|k \geq 2\} \cup \{\omega, all\}, \beta = a$ *u omitido.*

Demostración. El sistema SN P construido para la demostración de este teorema es una mezcla de las ideas de los teoremas anteriores. La construcción está dada en la figura 3.10. Se deja al lector los detalles del mismo.

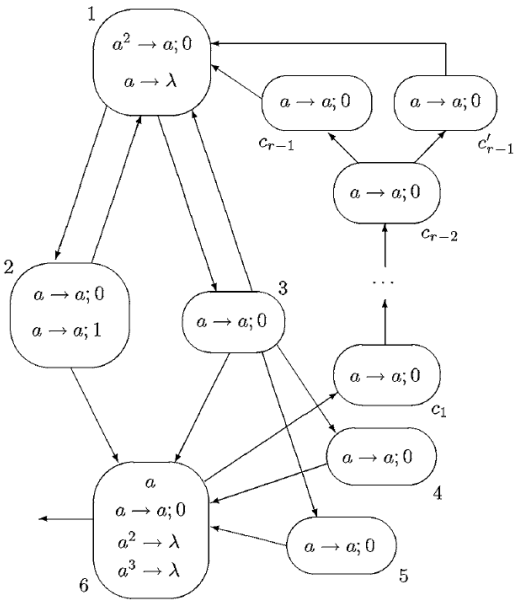


Figura 3.9: Figura de la construcción del sistema SN P para la demostración del teorema 3.13

Por el razonamiento dado a principios de la sección, gracias a estos teoremas y al teorema 3.5, se tiene el siguiente resultado (nótese que la cota de los spikes en las neuronas no es uniforme, y esto es por la forma en la que se computan los conjuntos unitarios):

Teorema 3.15. *$Spik^{\beta}_{\alpha}P_*(rule_k, cons_q, forg_p, bound_*) = NREG$ con $\alpha \in \{k|k \geq 2\} \cup \{\omega, all\}, \beta = a$ u omitido.*

Sin embargo, como ya se adelantó, los sistemas construidos en los teoremas de la sección computan con computaciones infinitas. Para el caso de aquellos conjuntos donde es necesario que la computación pare, las construcciones anteriores no son suficientes para demostrar el resultado buscado. Uno puede pensar en aplicar el resultado del lema 3.3, pero aquí solo acotamos el número de spikes, permitiendo al sistema SN P tener computaciones infinitas.

En el caso de tener sistemas SN P con spikes acotados, se puede demostrar el siguiente teorema, acotando no solo los spikes si no también haciendo las computaciones finitas. Esto nos proporciona la construcción necesaria para deducir el resultado para las combinaciones con $\alpha \in \{\underline{k}|k \geq 2\}, \beta \in ha, h$.

Teorema 3.16. *Dado un sistema SN P Π y una cota s en el número de spikes de*

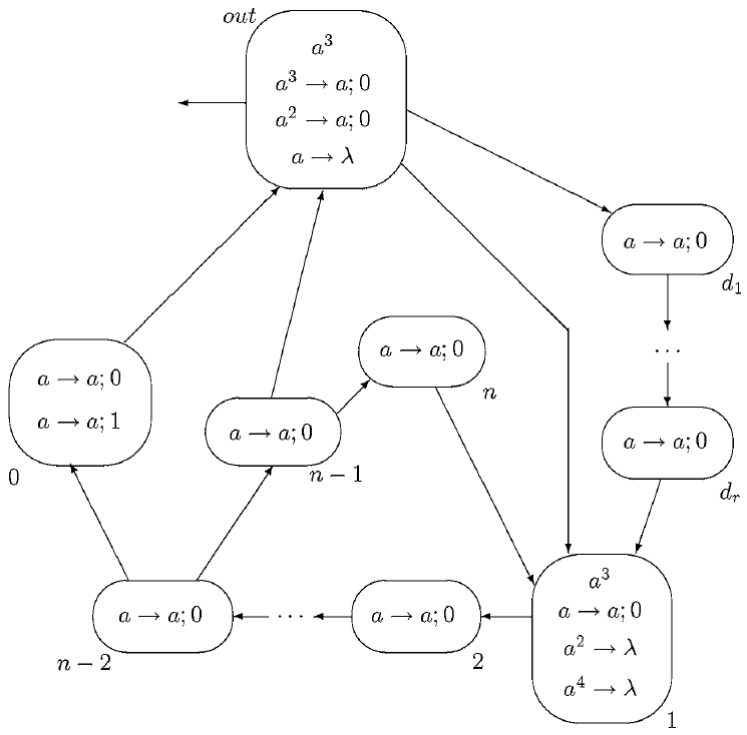


Figura 3.10: Sistema SN P construido para la demostración del teorema 3.14

cualquier neurona suya, se puede construir un sistema Π_k *de forma que:*

1. *Para cada computación* γ *s-acotada de* Π *con* $st(\gamma) = \langle t_1, \dots, t_j \rangle$, $j \leq k$, *existe una computación de parada* γ' $2s$*-acotada de* Π_k *tal que* $st(\gamma') = \langle t_1 + 2, \dots, t_j + 2 \rangle$.
2. *Para cada computación* γ *s-acotada de* Π *con* $st(\gamma) = \langle t_1, \dots, t_k, \dots \rangle$, *existe una computación de parada* γ' $2s$*-acotada de* Π_k *tal que* $st(\gamma') = \langle t_1 + 2, \dots, t_k + 2 \rangle$.
3. *Cada computación* γ' *de* Π_k *o bien nunca manda spikes, o bien* $st(\gamma') = \langle t_1 + 2, \dots, t_j + 2 \rangle$ *para alguna computación* γ *de* Π *donde* $st(\gamma) = \langle t_1, \dots, t_j \rangle$, $j \leq k$ *o bien* $st(\gamma') = \langle t_1 + 2, \dots, t_k + 2 \rangle$ *para alguna computación* γ *de* Π *donde* $st(\gamma) = \langle t_1, \dots, t_k, \dots \rangle$

Demostración. Sean $k \geq 1$ y Π un sistema SN P s-acotado. Sin pérdida de generalidad podemos asumir que las reglas de las neuronas de Π son todas de la forma $a^j/a^c \to x$ con $j \leq s$. Esto es porque como es s-acotado no podrá contener más de s spikes. Efectivamente, si una neurona σ_i contiene j spikes y la regla $E/a^c \to a; d$ y es aplicable en algún momento de la computación, por definición significará que $a^j \in L(E)$. Ahora bien, el contenido de la neurona σ_i está acotado por s, luego los únicos elementos

de $L(E)$ que pueden activar la regla pertnecen a$\{a, a^2, \dots, a^j, \dots, a^s\}$. Como es finito, podemos expresarlo como una sucesión de reglas de la forma $a^j/a^c \to x$.

Construyamos el sistema Π_k de la siguiente manera. Modificamos el sistema Π. Primero añadimos $s+2$ neuronas con las etiquetas *out* (neurona de salida de Π_k) y $1, \dots, s+1$, todas sin spikes al inicio de la computación. Estas neuronas son de la forma:

$$\sigma_{out} = (0, R_{out}); R_{out} = \{a \to a; 0\} \cup \{a^{s+i} \to \lambda | 1 \leq i \leq s+1\}$$
$$\sigma_i = (0, R_i); R_i = \{a^k/a \to a; 0\}$$

Además, a cada neurona del sistema SN P Π se le añade el conjunto de reglas

$$\{a^{s+i} \to \lambda | 1 \leq i \leq s+1\}$$

Se define por último si denotamos por syn a las conexiones sinápticas de Π y por i_0 a la su neurona de salida, se definen las conexiones sinápticas del sistema SN P Π_k como:

$$syn_k = syn \cup \{(i_0, out)\} \cup \{(i_0, i) | 1 \leq i \leq s+1\} \cup$$
$$\cup \{(i, out) | 1 \leq i \leq s+1\} \cup \{(i, l) | 1 \leq i \leq s+1, l \in syn\}$$

Así definido, el sistema Π_k manda spikes en su neurona de salida 2 instantes más tarde que la neurona de salida de Π (en los primeros k spikes). Además, cada vez que Π manda un spike, todas las neuronas etiquetadas $1, \dots, s+1$ lo reciben. Estas actúan como el contador de spikes del teorema 3.5 para rebosar todas las neuronas del sistema Π y la neurona *out* tras k spikes. Cuando se emite el k-ésimo spike, las neuronas $1, \dots, s+1$ todas disparan, haciendo a cada neurona recibir a^{s+1} spikes. Como se había asumido que no existían reglas que consumían más de s spikes en Π, y el contenido de las neuronas de Π está acotado por s, en el siguiente paso de computación en todas las neuronas una de las reglas del conjunto $\{a^{s+i} \to \lambda | 1 \leq i \leq s+1\}$ será aplicable, olvidándo así todas las spikes y parando la computación.

Así, el sistema SN P construido Π_k cumple las condiciones del teorema. ∎

Gracias a este teorema, los resultados del teorema 3.15 se pueden trasladar al resto de sistemas SN P, concluyendo la sección con el resultado buscado,

Teorema 3.17. *$Spik^{\beta}_{\alpha}P_*(rule_*, cons_q, forg_p, bound_*) = NREG$ con $\alpha \in \{k, \underline{k} | k \geq 2\} \cup \{\omega, all\}, \beta \in \{a, ha, h\}$ u omitido.*

Nótese que al construir los conjuntos unitarios, como ya se observó, no se pudo dar una cota uniforme al número de spikes en las neuronas. Como el número de reglas añadidas en la construcción del teorema 3.16 es mayor o igual que el número de spikes máximo por neurona de Π, en este resultado tampoco se puede garantizar la cota en el número de reglas del sistema.

4 | Apéndice

En el artículo original, en vez de presentar el resultado del Lema 4.1, se presentaba el siguiente sistema, que computa $n+2$ en vez de $n+1$. Se deja aquí para el lector interesado la explicación del sistema y la modificación del teorema posterior para adecuarse al cambio en el lema.

Lema 4.1. Sea Π un sistema SN P tal que cada computación γ de Π cumple que o bien $st(\gamma) = \emptyset$ o bien $st(\gamma) = \langle t, t+n \rangle$. Entonces existe un sistema SN P Π' cumpliendo:

1. Para cada computación γ de Π con $st(\gamma) = \langle t, t+n \rangle$ existe una computación γ' de Π' tal que $st(\gamma') = \langle t, t+(n+2), t+2(n+2), \dots \rangle$.
2. Cada computación γ' de Π' o bien no manda nunca spikes, o bien cumple $st(\gamma') = \langle t, t+(n+2), t+2(n+2), \dots \rangle$ con $\langle t, t+n \rangle$ siendo el tren de spikes de una computación de Π.

Demostración. Se dará la prueba del resultado apoyándonos en la figura 4.1.

El sistema SN P Π' comienza simulando una computación de Π (el sistema cumpliendo las condiciones del enunciado). Si la computación γ no para (y no emite spikes), entonces la computación γ' del sistema Π' no para y tampoco emite spikes. Si la computación γ del sistema Π para y manda 2 spikes, $st(\gamma) = \langle t, t+n \rangle$, entonces el sistema Π' generara una computación infinita γ' cumpliendo $st(\gamma') = \langle t, t+(n+2), t+2(n+2), \dots \rangle$. La idea detrás del sistema creado es la siguiente:

1. Primero se simula el sistema Π. Si no para es claro que Π' no para. Si el sistema Π para entonces se pasa al módulo de inicialización.
2. El módulo de inicialización consiste en aumentar la cantidad de spikes de salida, de Π, haciendolo impar. Más precisamente, si se tiene que la computación γ de Π cumple $st(\gamma) = \langle t, t+n \rangle$, entonces el módulo de inicialización se encargará de mandar $2n+1$ spikes a la neurona m_1. El hecho de que la cantidad de spikes

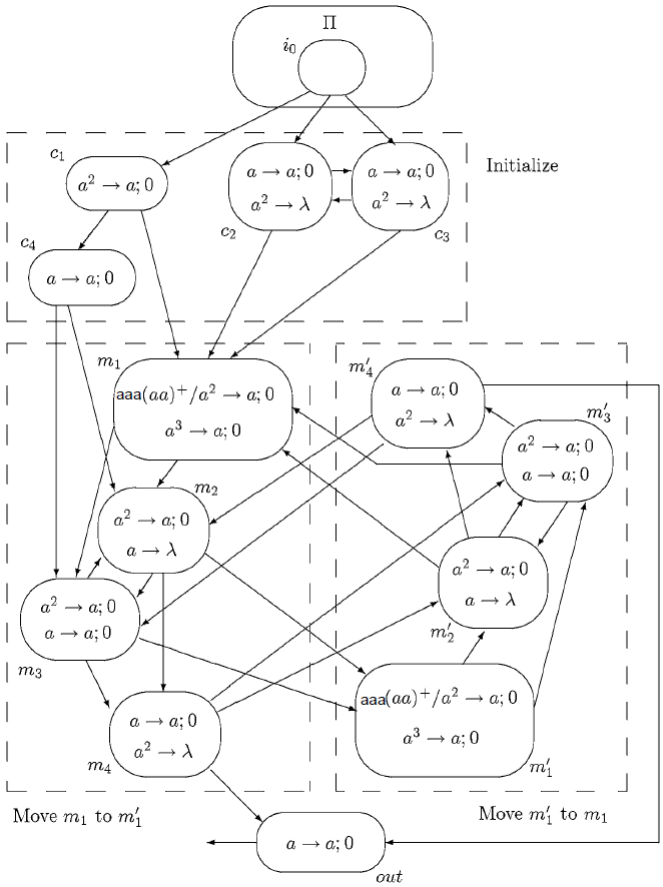


Figura 4.1: Idea de la construcción de la demostración del teorema 4.1

sea impar es importante porque las reglas de m_1 solo serán aplicables cuando haya un contenido impar de spikes mayor que 1. De esta forma, la manera de cargar el módulo será mandar primero $2n$ spikes de 2 en 2 y luego mandar el último.

3. Al mandar el último, el módulo mover m_1 a m'_1 es activado. Aquí, tras una serie de pasos, la neurona m'_1 contiene $2n+1$ spikes y la neurona de salida manda un spike.
4. A la vez que se manda el spike de salida se inicia el módulo mover m'_1 a m_1, que funciona de forma análoga al módulo mover m_1 a m'_1.
5. La distancia entre los spikes es $n+2$ y la computación de Π' es infinita y contiene infinitos spikes.

Veamos ahora el esquema general con algo más de detalle. Sea Π un sistema SN P

y γ una computación de este tal que $st(\gamma) = \langle t, t+n \rangle$. En los primeros $t-1$ pasos el sistema Π' se comporta igual que el sistema Π. En el paso t, i_0(la que sería la etiqueta de salida de Π), manda el primer spike. Esto hace que llegue un spike a c_1, c_2 y c_3. En el paso $t+1$ las neuronas c_2, c_3 se mandan entre ellas un spike, mandando también cada una un spike a m_1. De esta forma, m_1 recibirá 2 spike en cada paso de computación siguiente. Nótese que como se destacó en el resumen, las reglas de m_1 solo son aplicables cuando contiene un número impar de spikes. Por otro lado, la neurona c_1 necesita 2 spikes para que su regla sea aplicable, luego no se activará hasta que reciba otro spike (de la neurona i_0). Así, tras $n-1$ pasos de computación, m_1 tendrá $2n-2$ spikes, c_1, c_2, c_3 tendrán cada una un spike. En el paso $t+n$ la neurona i_0 dispara otro spike. De esta forma, c_1 ya tiene 2 spikes para disparar en el siguiente paso. A su vez, c_2, c_3 disparan en el paso $t+n$, dejando m_1 con $2n$ spikes y a ellas con 2 spikes cada una (uno recibido de i_0) y el otro que se han mandado entre ellas). Así, en el paso $t+n+1$ se tiene que c_2, c_3 se olvidan de los spikes que contenían, c_1 manda un spike a m_1 y a c_4, haciendo que m_1 tenga ahora un número impar de spikes y que sus reglas sean aplicables.

Comienza el módulo mover m_1 a m'_1. La neurona c_4 servirá como una ayuda para "poner en marcha" las neuronas m_2 y m_3 en el primer paso de este módulo. Este módulo consiste en pasar los contenidos de m_1 a m'_1. Al igual que m_1, las reglas de m'_1 solo son aplicables cuando contiene un número impar de spikes. Así, se pasarán los spikes de 2 en 2 de manera similar al paso anterior.

Continuamos por tanto en el paso $t+n+2$. En el primer paso la neurona m_1 manda un spike a m_2 y m_3 olvidando 2 spikes con la regla $aaa(aa)^+/a^2 \to a; 0$. Además, las neuronas m_2 y m_3 reciben un spike de c_4. En el siguiente paso, $t+n+3$, 2 spikes son mandados a la neurona m'_1, de parte de m_2 y m_3. Además, m_2 y m_3 se mandan spikes entre sí. La neurona m_1 vuelve a enviar spikes a m_2 y m_3, haciendo que en el siguiente paso vuelvan a tener 2 spikes y que se vuelvan a disparar (de esta forma aquellas spikes que se olvidan de m_1 pasan en el siguiente paso a m'_1). Nótese además que m_4 en cada paso recibe spikes de m_3 y m_2, olvidándolas en el siguiente paso. La computación sigue este curso hasta que la regla $a^3 \to a$ es aplicable en m_1. Este es justamente el paso $t+n+2+(n-1) = t+2n+1$. En este instante la neurona m'_1 contiene $2n-4$ spikes. Tras este paso, en el instante $t+2n+2$, se tiene que m'_1 contiene $2n-2$ spikes, las neuronas m_2, m_3 están preparadas para disparar de nuevo y la neurona m_1 está vacía. En el paso $t+2n+3$ la neurona m'_1 contiene $2n$ spikes (ha recibido 2 de m_2 y m_3), las neuronas m_2 y m_3 tienen ambas 1 spike. Así, en el paso $t+2n+4$ la neurona m_3 manda un spike a m_4 y a m'_1 (que ahora tiene un número impar de spikes), mientras que la neurona m_2 se olvida del suyo. Así, en el instante $t+2n+5$

se inicia el módulo mueve m_1' a m_1 y la neurona m_4 manda un spike a la neurona de salida (y actúa como actuaba c_4 en el primer paso de mueve m_1 a m_1'). Por último, en el paso $t + 2n + 6$ se emite un spike y se sigue con la computación del módulo mueve m_1' a m_1.

Por tanto, entre cada spike hay $t + 2n + 6 - (t + n + 4) = n + 2$ pasos. Por lo tanto, por cómo está construido el sistema y las observaciones hechas al principio, se deduce el resultado.

Teorema 4.1. *$Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q) = NRE$ para todo $k \geq 2, p \geq 3, q \geq 3$ y $\alpha \in \{\omega\} \cup \{k | k \geq 3\}, \beta = a$ o β omitido.*

Demostración. Como siempre, gracias a la tesis de Church-Turing basta probar que $NRE \subset Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$. Sea $Q \in NRE$ y denotemos por $Q' = \{n - 2 | n \in Q\} \in NRE$. Por el teorema 3.1 se tiene que existe un sistema SN P Π cumpliendo $N_{\underline{2}}^h(\Pi) = Q'$. Aplicando ahora el lema 4.1 al sistema Π se obtiene un sistema Π' cumpliendo $N_\omega(\Pi') = N_k(\Pi') = \{n + 2 | n \in N_{\underline{2}}^h(\Pi) = Q'\} = Q - \{1, 2\}$ para todo $k \geq 2$.

Caso 1: $1, 2 \notin Q$: Entonces se tiene que $Q = Q - \{1, 2\} \in Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$ demostrando así el teorema.

Caso 2: $1 \in Q, 2 \notin Q$: Para ver que Q está entonces en $Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$ basta construir un sistema SN P Π'' que compute el conjunto unitario $\{1\}$ y aplicar el teorema 3.5 (que nos dice que podemos hacer la unión finita de los dos sistemas) a Π' y a Π'', obteniendo así que el conjunto $Q = Q - 2 = \{1\} \cup Q - \{1, 2\}$ es computado por la unión de los sistemas Π'', Π' y por tanto $Q \in Spik_\alpha^\beta P_*(rule_k, cons_p, forg_q)$.

Este sistema SN P Π'' que computa el conjunto unitario $\{1\}$viene dado por dos neuronas conectadas entre sí $(syn = \{(1, 2), (2, 1)\}$ cumpliendo $R_i = \{a \to a; 0\}$ para $i = 1, 2$ con una configuración inicial dada por:

$$(\sigma_1 = (1, 0), \sigma_2 = (1, 0))$$

Donde es conveniente recordar que el 1 representa el número de spikes en la neurona σ_i y el 0 representa el tiempo que queda para que la neurona esté abierta.

Es fácil comprobar que este sistema SN P computa el conjunto unitario $\{1\}$.

Caso 3: $2 \in Q, 1 \notin Q$: Análogamente al caso anterior basta construir un sistema Π''' que compute el conjunto unitario $\{2\}$. Este sistema viene dado por $\Pi''' = (\{a\}, \sigma_1, \sigma_2, syn, 1)$ donde:

$$\sigma_i = (1, R_i), \quad R_i = \{a \to a; 1\}, i \in \{1, 2\}$$

$$syn = \{(1,2),(2,1)\}$$

De forma análoga al caso anterior es fácil comprobar que este sistema computa el conjunto unitario $\{2\}$.

Caso 4: $2,1 \in Q$: Basta considerar el sistema dado por la unión de los sistemas Π', Π'', Π''' razonando de forma análoga a los casos 2 y 3, acabando la demostración del teorema.

Bibliografía

[1] G. Rozenberg Gh. Păun M.J.Pérez-Jiménez. "Spike trains in spiking neural P systems". En: *International Journal of Foundations of Computer Science* 17.4 (2006), págs. 975-1002.

[2] T. Yokomori M. Ionescu Gh. Păun. "Spiking neural P systems". En: (2005).

[3] M. Minsky. *Computation - Finite and Infinite Machines*. Englewood Cliffs, NJ: Prentice Hall, 1967.