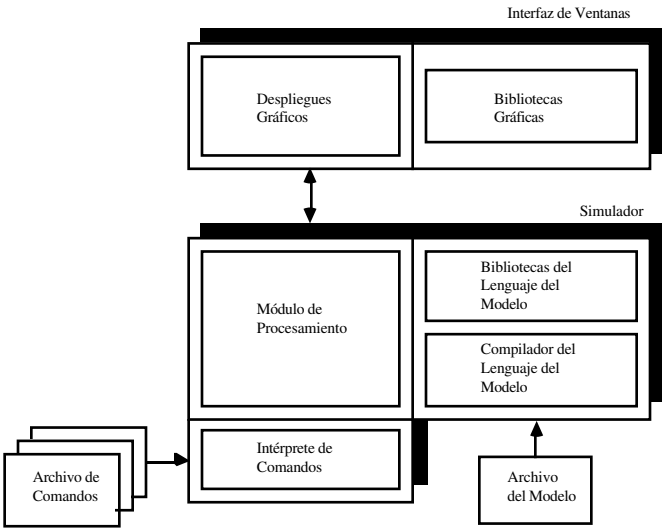

Figura 1. El sistema de simulación NSL está compuesto de dos unidades: (1) el Simulador, compuesto del Módulo de Procesamiento, el Compilador del Lenguaje del Modelo, las Bibliotecas del Lenguaje del Modelo, y el Intérprete de Comandos; (2) la Interfaz de Ventanas compuesta de la Unidad de Despliegues Gráficos y las Bibliotecas Gráficas.

3 Lenguaje NSL

Para poder modelar redes neuronales con NSL, es necesario describir (1) las neuronas que componen la red, (2) las interconexiones de las neuronas, y (3) la dinámica de las neuronas.

3.1 Neuronas

El modelo básico en NSL es de la neurona de un solo compartimiento, teniendo una salida y varias entradas, como se muestra en la Figura 2. El estado interno de la neurona es descrito por un solo valor, su potencial de membrana m, el cual depende de las entradas de la neurona y de su historia previa. La salida es descrita por otro valor, su *tasa de disparo* M, y puede servir como entrada a varias otras neuronas, incluyendo a sí misma. Al variar los valores de las entradas a una neurona, varía también su potencial de membrana y su tasa de disparo.

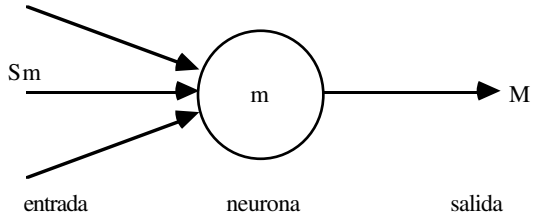

Figura 2. Modelo neuronal de un solo compartimiento es representado por un valor m correspondiente al potencial de membrana, y un valor M correspondiente a su tasa de disparo, la única salida de la neurona. S_m representa el conjunto de entradas a la neurona.

El potencial de membrana para m es descrito por la siguiente ecuación diferencial

$$\tau_m \frac{dm(t)}{dt} = f(S_m, m, t)$$

la cual depende de las entradas S_m, el valor previo de m, el parámetro temporal t, y la constante de tiempo τ_m. La elección de f depende del modelo neuronal particular. Por ejemplo, el *integrador con fugas* es descrito por

$$f(S_m, m, t) = -m(t) + S_m(t)$$

La *tasa de disparo M* (salida de la neurona) se obtiene aplicando una *función umbral* al potencial de la membrana de la neurona,

$$M(t) = \sigma(m(t))$$

donde σ es usualmente una función no lineal. Algunas de las funciones de *umbral* más comunes, como *rampa*, *escalón*, *saturación* y *sigmoide*, se muestran en la figura 3.

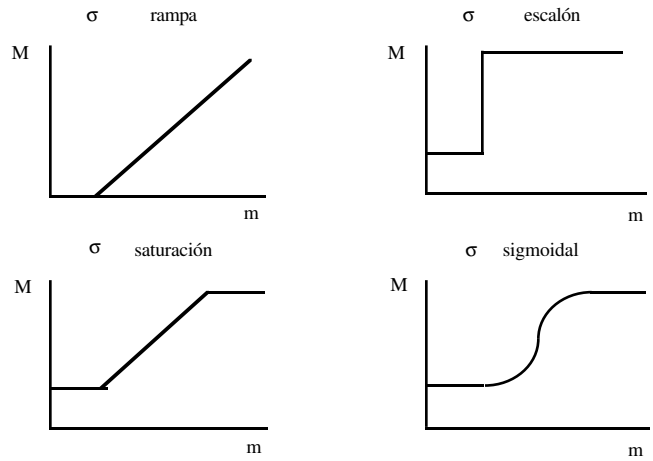

Figure 3. La figura muestra algunas funciones de umbral σ típicas.

En NSL dos estructuras **DATA** son requeridas para representar una neurona. Una **DATA** correponde al potencial de membrana y la otra a la tasa de disparo. La notación es la siguiente (con un punto y coma al final de cada oración):

```
DATA m;
DATA M:
```

El potencial de membrana *m* se representa por la ecuación diferencial

```
DIFF(m,τm) = f(Sm,m);
```

donde **DIFF** define una ecuación diferencial de primer orden para *m* con constante de tiempo τ_m, (el parámetro temporal *t* está implícito en la ecuación). El modelo del integrador con fugas corresponde a

```
DIFF(m,τm) = −m + Sm;
```

La tasa de disparo *M* se representa sencillamente con

```
M = σ(m);
```

donde σ representa la selección de función umbral.

3.2 Interconexiones

Cuando se construyen redes neuronales, la salida de una neurona sirve como entrada a otras neuronas. Las conexiones entre neuronas llevan un peso de conexión que describe cómo las neuronas están afectadas entre sí. Las conexiones son llamadas excitatorias o inhibitorias dependiendo de si su peso es positivo o negativo, respectivamente. La fórmula más común para la entrada a una neurona es

$$S_v = \sum_{i=1}^{n} w_i U_i(t)$$

donde $U_i(t)$ es la tasa de disparo de la neurona u_i cuya salida está conectada a la entrada *i* de la neurona *v*, y w_i es el peso de la conexión, como se muestra en la Figura 4. (*u* y *v* son análogas a *m*, mientras que *U* y *V* son análogas a *M*.)

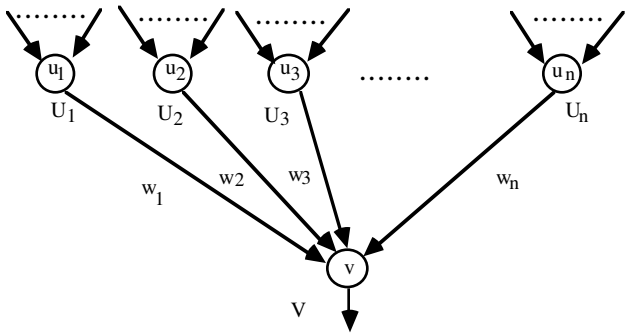

Figure 4. La neurona *v* recibe entradas de las neuronas $u_1,...,u_n$, correspondientes a sus tasas de disparo, $U_1,...,U_n$, multiplicadas por los pesos $w_1,...,w_n$, respectivamente.

Por ejemplo, la arquitectura de la red neuronal correspondiente al modelo del *Máximo Selector*, también conocido como el modelo *Didday* (ver Arbib [1989], se muestra en la Figura 5. *u* y *v* representan potenciales de membrana, *U* y *V* representan las tasas de disparo, S_v representa la entrada de la neurona *v*, S_{ui} representa la entrada de la neurona u_i, mientras que w_m, w_{ui}, y w_i representan los pesos de las conexiones, los cuales no son necesariamente iguales.

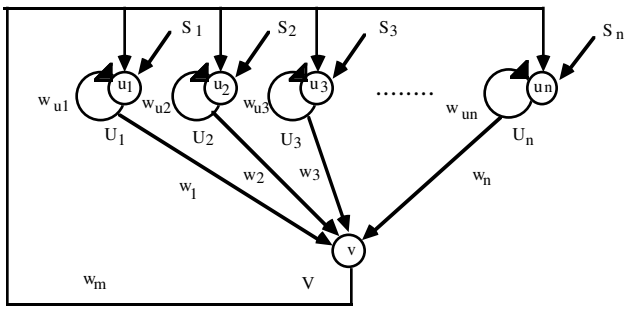

Figura 5. La red neuronal en la figura corresponde a la arquitectura del modelo Máximo Selector o Didday (ver Arbib [1989], donde Si representa las entradas a la red, ui y v representan potenciales de membrana mientras que Ui y V representan tasas de disparo. w_m, w_{ui}, y w_i corresponden a los pesos de las conexiones.

La entrada a la neurona *v* es dada por

$$S_v = w_1U_1 + w_2U_2 + ... + w_nU_n$$

mientras que la entrada a la neurona u_i es (existen *n* ecuaciones similares)

$$S_{ui} = w_mV + w_{ui}U_i + S_i$$

En NSL, estas expresiones, que describen interconexiones entre neuronas en la red neuronal, están descritas de forma similar. Por ejemplo, la entrada a la neurona *v*, representada por S_v, sería la suma de las salidas de todas las neuronas u_i multiplicadas (el operador '*') por el peso de conexión w_i correspondiente:

```
S_v = w_1*U_1 + w_2*U_2 + ... + w_n*U_n
```

La entrada a la neurona u_i, es representada por S_{ui} (existen *n* ecuaciones similares)

```
S_ui = w_m*V + w_ui*U_i + S_i
```

3.3 Capas y Máscaras

Cuando se modela miles de neuronas se vuelve extremadamente dificil nombrar a cada una de ellas. En el cerebro usualmente encontramos redes neuronales estructuradas en capas de dos dimensiones de neuronas homogéneas, con patrones de conexión regulares entre varias capas. Por lo tanto, extendemos la abstracción de la neurona básica a capas de neuronas y las conexiones sencillas a máscaras de conexión.

La ventaja computacional de introducir tales conceptos al describir la red neuronal es que las capas neuronales y las máscaras de interconexión pueden ser descritas de forma muy concisa como estructuras de alto nivel. Una interconexión entre neuronas puede ser procesada computando una *convolución* espacial entre una máscara y una capa. Por ejemplo, como se muestra en la Figura 6, si *A* representa un arreglo de salidas de una capa de neuronas, y *B* representa un arreglo de entradas a otra capa, y si la máscara $W(k,l)$ (para $-d \le k,l \le d$) representa el peso entre los elementos de $A(i+k,j+l)$ (para $-d \le k,l \le d$) y los elementos $B(i,j)$ para cada i y j, entonces tenemos

$$B = \sum_{k=-d}^{d} \sum_{l=-d}^{d} W(k,l)A(i+k, j+l)$$

que puede ser expresada por una operación sencilla

$$B = W * A$$

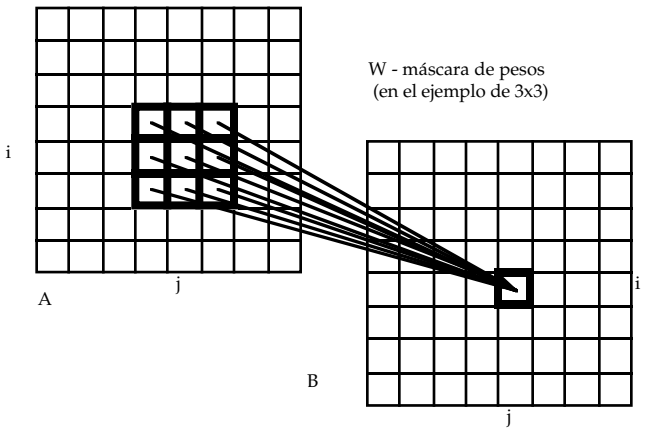

Figura 6. W representa la conexión o máscara de convolución entre las capas A y B correspondientes a la ecuación B=W*A. En este ejemplo W es una máscara de 3x3 que se superpone sobre una ventana en A del mismo tamaño, para obtener un valor de un elemento en B (en este ejemplo las capas A y B consisten de 8x8 neuronas cada una).

En término de sintaxis, las ecuaciones para la red serían exactamente las mismas si se describiera capas de neuronas o neuronas sencillas. La única diferencia es que ahora los factores que representan pesos de conexión serían mascaras guardando arreglos de pesos, y las variables que representan

neuronas ahora representarían capas de neuronas. El operador de multiplicación pasa a ser una convolución.

Para poder apoyar capas y máscaras, la estructura básica **DATA** en NSL es extendida con dos tipos de capas adicionales, **VECTOR**, y **MATRIX**, variando el número de dimensiones. Para simplificar las cosas, máscaras, que pueden tener cualquier forma rectangular, son también definidas en NSL como capas cuya actividad describe pesos de conexión en lugar de potenciales de membrana o tasas de disparo.

Por ejemplo, las capas de las neuronas mostradas en la Figura 5 serían descritas por

```
VECTOR(S,n);
VECTOR(u,n);
VECTOR(U,n);
DATA(v);
DATA(V);
```

3.4 Métodos Numéricos

Una característica común de muchos modelos de neuronas, y de los correspondientes métodos numéricos que los describen, es que son procesados como sistemas de tiempo *continuo*, pero con el estimado del estado de cada neurona actualizado durante tiempos discretos.

Por ejemplo, consideremos la ecuación diferencial de primer orden dada por

$$\tau_m \frac{dm(t)}{dt} = -m(t) + S_m(t)$$

donde $S_m(t)$ representa entradas externas, y τ_m es la constante de tiempo, con la correspondiente descripción en NSL

```
DIFF(m,tm) = -m+Sm;
```

La ecuación diferencial puede ser resuelta usando diferentes métodos numéricos. Por ejemplo, aplicando el método *Euler* a la ecuación diferencial genera

$$m(t+\Delta t) = (1-\frac{\Delta t}{\tau_m})m(t) + \frac{\Delta t}{\tau_m} S_m(t)$$

donde Δt es el paso de tiempo de integración, o representado internamente en NSL por

$$m = (1-\frac{step}{t_m})m + \frac{step}{t_m} S_m$$

donde *step* es el intervalo de tiempo de integración en NSL.

Usando un método diferente a Euler, como el método de *Interpolación*, la ecuación diferencial del modelo produciría

$$m(t+step) = (p)\ m(t) + (1 - p)\ S(t)$$

donde $p = \exp(-step/t_m)$.

4 Ejemplo

La descripción completa de las ecuaciones utilizados en el modelo del Máximo Selector, mostrado en la Figura 5, son

$$\tau_u \frac{du_i(t)}{dt} = -u_i + w_u f(u_i) - w_m g(v) - h_1 + S_i$$

$$1 \le i \le n$$

$$\tau_v \frac{dv}{dt} = -v + w_n \sum_{i=1}^{n} f(u_i) - h_2$$

donde w_{ui} es el peso de conexión para la autoalimentación de u_i, $w_{u1} = w_{u2} = ... = w_{un} = w_u$ y $w_1 = w_2 = ... = w_n$ (en este modelo particular los pesos de conexión son iguales), h_1 y h_2 son constantes (con la restricción de que $0 \le h_1$, y $0 \le h_2 < 1$) y las funciones umbral son un escalón para $f(u_i)$

$$f(u_i) = \begin{cases} 1 & u_i > 0 \\ 0 & u_i \le 0 \end{cases}$$

y una rampa para $g(v)$

$$g(v) = \begin{cases} v & v > 0 \\ 0 & v \le 0 \end{cases}$$

4.1 Descripción de la red en NSL

En NSL la descripción completa del modelo se organiza en dos tipos principales de módulos: **INIT_MODULE** conteniendo instrucciones de inicialización, y **RUN_MODULE**, conteniendo instrucciones a ejecutarse como parte de la simulación.

En este ejemplo, especificamos un **INIT_MODULE**, llamado *init*, y dos **RUN_MODULE**s, uno llamado *U* para especificar como las capas *u* y *U* son actualizadas, y otro llamado *V* para las capas *v* y *V*

```
INIT_MODULE(init)
{
    u = 0;
    v = 0;
    U = 0;
    V = 0;
}
```

```
RUN_MODULE(U)
{
   DIFF.eq(u,tu) = - u + wu*U
                - wm*V - h1 + S;
   U = NSLstep(u);
}
```

```
RUN_MODULE(V)
{
   DIFF.eq(v,tv) = - v
               + NSLsum(wn*U) - h2;
   V = NSLramp(v);
}
```

Se debe observar que S,u, yU son capas de vectores y todas las operaciones se aplican a las capas completas. **NSLsum**(w_n*U) primero multiplica el peso de la conexión w_n por la tasa de disparo U, que devuelve un valor correspondiente a la suma de los elementos de la capa. Esto es necesario ya que la capa v contiene un solo elemento. **NSLstep** y **NSLramp** son las correspondientes funciones umbral para las dos capas de neuronas en la red.

Es necesario añadir declaraciones en NSL para todas las estructuras utilizadas, incluyendo parámetros y constantes

```
DATA(tu);
DATA(tv);
DATA(h1);
DATA(h2);
DATA(wu);
DATA(wm);
DATA(wn);
```

Finalmente, es necesario designarle un nombre a la red por medio de la estructura **NETWORK**

```
NETWORK(DIDDAY);
```

4.2 Simulación de la red en NSL

La simulación de una red neuronal se efectúa en un ambiente interactivo. Este ambiente es apoyado por archivos que contienen parámetros de la simulación, valores de entradas de la red, y especificaciones de gráficos.

En la primera parte del archivo, la red apropiada es referenciada, en este caso 'DIDDAY'. Las otras entradas incluyen 'EULER' como método de integración, y la especificación de ejecutar el modelo comenzando del tiempo 0 al tiempo 20.0, usando un intervalo de tiempo, *delta*, de 0.1, dando un total de 200 iteraciones del modelo.

```
set network DIDDAY
set integration EULER
set delta 0.1
set end_time 20.0
```

Se deben asignar parámetros interactivos, o parámetros cuyos valores no fueron especificados originalmente en la red. Luego, especificamos la entrada a ser usada en la simulación. En este caso, se mantienen las entradas constantes durante toda la simulación. Todos los valores del arreglo S son 0 excepto 1.0 en la cuarta entrada (los arreglos comienzan con índice 0) y 0.5 en la sexta entrada.

```
set data_value tu 1.0
set data_value tv 1.0
set data_value wu 1.0
set data_value wm 1.0
set data_value wn 1.0
set data_value h1 0.1
set data_value h2 0.5

set data_value S
   0 0 0 0 1.0 0 0.5 0 0 0
```

Finalmente, se especifican las ventanas que serán usadas para desplegar las capas, por ejemplo, la entrada S y el potencial de membrana u. Se especifica, en este caso, dos despliegues contenidos en la ventana llamada *FW1*, que incluye dos secciones:

```
create window_interface
set frame_name FW1
set frame_width 500
set frame_height 350
set frame_rows 2
create display_frame
```

La primera fila contiene una ventana *WN11* que despliega el vector *S*, cada elemento mostrado como un gráfico de tipo **area_level_graph**; mientras que la segunda fila contiene una ventana *WN12* que despliega el vector *u*, cada elemento mostrado como un gráfico de tipo **temporal_graph**. La salida de la red se muestra en la Figura 7.

```
set window_name WN11
set window_layer S
set window_graph area_level_graph
set window_wymin 0
set window_wymax 2
create display_window

set window_name WN12
set window_layer u
set window_graph temporal_graph
set window_wymin -3
set window_wymax 3
set window_t1 20
create display_window
```

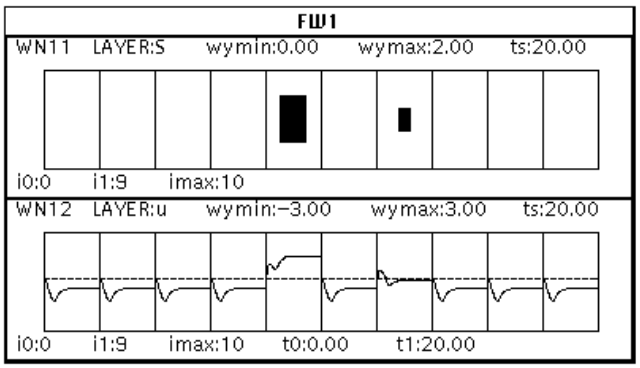

Figura 7. La salida del modelo Didday, muestra la capa de entrada *S*, en un gráfico de tipo **area_level_graph**, y el potencial de membrana de la capa *u*, en un gráfico de tipo **temporal_graph**.

5 Discusión

NSL ha sido utilizado ampliamente como una herramienta de simulación para redes neuronales biológicas y artificiales, donde diferentes tipos de aplicaciones han sido desarrolladas, como el modelo de *coordinación visuomotor* [Arbib and Lee, 1993], y el modelo de *generación de movimientos sacádicos* [Dominey and Arbib, 1992], entre otros. El principal desafío en el desarrollo de NSL, al igual que en otras herramientas de simulación es, por un lado proveer ambientes de simulación generales y amigables, mientras que por otro lado se debe ser lo más eficiente posible en la intensa tarea de simular redes neuronales.

La siguiente generación NSL3.0, ya en desarrollo, ofrecerá un ambiente distribuido y paralelo para la simulación de redes neuronales [Weitzenfeld and Arbib, 1991], y la integración de modelos de redes neuronales con sistemas de *esquemas*, como se describe en ASL, Abstract Schema Language, [Weitzenfeld, 1993]. El sistema ASL permitirá el desarrollo de redes neuronales jerárquicas y distribuidas, como en las aplicaciones de robótica para la *integración sensorimotor* [Fagg et al., 1992]. Más aún, se podrá integrar las redes neuronales con otras aplicaciones tales como algoritmos genéticos [Lewis et al., 1992].

Referencias

[Arbib, 1989] Arbib, M.A., 1989, The Metaphorical Brain 2: Neural Networks and Beyond, Wiley.

[Arbib and Lee, 1993] Arbib, M.A., Lee, H.B., 1993, Anuran Visuomotor Coordination for Detour Behavior: From Retina to Motor Schemas, in From Animals to Animats 2: Proc. of 2nd International Conference on Simulation of Adaptive Behavior (J.-A. Meyer, H.L. Roitblat, and S. Wilson, Eds), A Bradford Book/MIT Press :42-51.

[De Schutter, 1992] De Schutter, E., 1992, A Consumer Guide to Neuronal Modeling Software, in Trends in Neuroscience, 15(11):462-464.

[Dominey and Arbib, 1992] Dominey, P.F., Arbib, M.A., 1992, A Cortico-Subcortical Model for Generation of Spatially Accurate Sequential Saccades, Cerebral Cortex, 2:153-175.

[Fagg et al., 1992] Fagg, A.H., King, I.K., Lewis, M.A., Liaw, J.S., Weitzenfeld, A., 1992, A Neural Network Based Testbed for Modeling Sensorimotor Integration in Robotics Applications, Proc. of IJCNN '92, Baltimore, MD.

[Goddard et al., 1986] Goddard, N., Lynne, K.J., Mintz, T., 1986, Rochester Connectionist Simulator, User Manual, University of Rochester, Department of Computer Science.

[Hodgkin and Huxley, 1952] Hodgkin, A.L., Huxley, A.F., 1952, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. London, 117:500-544.

[Leighton, 1991] Leighton, R.R., 1991, The Aspirin/MIGRAINES Software Tools, User's Manual, V5.0, Technical Report, MITRE Washington Neural Network Group.

[Lewis et al., 1992] Lewis, M.A., Fagg, A.H., Solidum, A., 1992, A Genetic Approach to the Development of a Neural Network for the Control of a Walking Machine, Proc. of the IEEE Conference of Robotics and Automation, pp. 2618-23, Nice, France.

[Rall, 1959] Rall, W., 1959, Branching dendritic trees and motoneuron membrane resistivity, Exp. Neurol., 2: 503-532.

[Stroustrup, 1991] Stroustrup, B., 1991, The C++ Programming Language, 2nd. Ed., Addison-Wesley.

[Wegner, 1990] Wegner, P., 1990, Concepts and Paradigms of Object-Oriented Programming, OOPS Messenger, 1(1):7-87, Aug.

[Weitzenfeld, 1991] Weitzenfeld, A., 1991, NSL: Neural Simulation Language, Version 2.1, CNE-TR 91-05, University of Southern California, Center for Neural Engineering, Los Angeles, CA.

[Weitzenfeld and Arbib, 1991] Weitzenfeld, A., Arbib, M., 1991, A Concurrent Object-Oriented Framework for the Simulation of Neural Networks, Proceedings of ECOOP/OOPSLA '90 Workshop on Object-Based Concurrent Programming, OOPS Messenger, 2(2):120-124, April.

[Weitzenfeld, 1993] Weitzenfeld, A., 1993, A Hierarchical Computational Model for Distributed Heterogeneous Systems, TR 93-02, Center for Neural Engineering, University of Southern California, Los Angeles, California, May.

[Weitzenfeld, 1995] Weitzenfeld, A., 1995, NSL - Neural Simulation Language, en The Handbook of Brain Theory and Neural Networks, Editor M.A. Arbib, Bradford/MIT Press.

[Weitzenfeld and Arbib, 1994] Weitzenfeld, A., Arbib, M.A., 1994, NSL Neural Simulation Language, in Neural Network Simulation Environments, Ed. J. Skrzypek, Kluwer.

[Weitzenfeld and Arbib, 1995] Weitzenfeld, A., Arbib, M.A., 1995, NSL - Neural Simulation Language: System and Applications, MIT Press (en preparación).