

Agradecimientos

Muy al principio de la maestría, Angélica García me presentó un artículo sobre redes de comportamiento para que lo revisara. Dos años más tarde, tuvo que soportar mi desordenada forma de trabajar y mis hábitos nocturnos como asesora de este trabajo. Hay mucho que agradecerle: sus consejos, sus clases, su trabajo como coordinadora de la maestría y sobre todo su amistad.

Quisiera agradecer a quienes revisaron este trabajo: al Dr. Christian Lemaître cuya revisión fue de gran ayuda para afinar detalles importantes en la redacción final. Debo agradecerle su insistencia en que implementáramos ambientes gráficos. La experiencia ganada en sus cursos hizo más fácil la programación de ABC. Al Dr. Manuel Martínez por sus comentarios durante las presentaciones en el seminario del proyecto monots y su disposición permanente a comentar las diferentes etapas de ABC y mi trabajo. Al Dr. Luciano García, cuya ayuda fue central para la presentación del modelo matemático de las redes de comportamiento.

A mis compañeros de andanzas: a Angélica Muñoz y al siempre genial Rubén de la Mora, quienes enriquecieron este trabajo enormemente gracias a las conversaciones sostenidas durante todo este tiempo; a Memo Hoyos siempre dispuesto a tender la mano.

A quienes contribuyeron con el trabajo, mi agradecimiento: a Esther Martínez por su ayuda con el mundo de cubos; a Fernando Montes por la red de forrajeo para el *allouta palliata*.

A Hilda, Mara y José Luis. Siempre pendientes de que recibos, certificados, notificaciones y archivos estén donde deben.

A quienes me ayudaron durante mi llegada a Xalapa: Paulina Gutiérrez y Chucho Arenzano. Ambos hicieron algo realmente admirable, soportarme en sus casas.

Al Dr. José Negrete hay muchas cosas que agradecerle: sus enseñanzas, su plática siempre inteligente, su humor punzante, la pasión con que hace su trabajo, sus contribuciones a este trabajo, pero sobre todo el que se halla decidido “a zarpar en compañía de tan entusiasta tripulación en el primer viaje del buque de la MIA de la UV”.

Este trabajo fue realizado con el apoyo del Consejo Nacional de Ciencia y Tecnología, a través de la beca con registro 70354

A Carmen y Carmelo que han hecho lo imposible para que este trabajo estuviera terminado, su apoyo ha sido definitivo y además, son unos padres excelentes. Su ejemplo y sus ganas de hacer cosas en esta vida son la mejor escuela que he tenido. Gracias.

A los de casa: Carmela, Aurora y María, las muchachas Berriel, siempre pendientes de mis cosas. Darles las gracias por todo lo que me han dado en la vida no es suficiente (algo se me ocurrirá). A Tato y a mi hermana Laura, por su cariño de siempre.

A Susana, por ser mi amor, mi cómplice y todo.

Índice

Introducción	i
Capítulo 1. Agentes	1
1.1 ¿Qué es un agente?.....	1
1.2 Clases de agentes	6
1.3 Problemas en el diseño e implementación de agentes autónomos	9
Capítulo 2. Selección de Acción	13
2.1 El problema de selección de acción	13
2.2 La Arquitectura de Subsumción.....	16
2.3 Redes de Comportamiento	18
2.4 Hamsterdam	19
2.5 La Sociedad de la Mente.....	20
Capítulo 3. Redes de Comportamiento	22
3.1 ¿Cómo hacer lo correcto?	22
3.2 Descripción del algoritmo	23
3.3 Modelo matemático.....	27
3.3.1 Módulo de habilidad.....	27
3.3.2 Agente	28
3.3.3 Proposiciones	28
3.3.4 Percepción	28
3.3.5 Metas globales del agente.....	29
3.3.6 Módulo de habilidad ejecutable	29
3.3.7 Parámetros	29
3.3.8 Dinámica de Activación / Inhibición	29
3.3.9 Módulo de habilidad activo	32
3.4 Propiedades y configuración de parámetros	32

3.4.1	<i>Capacidad de planificación</i>	32
3.4.2	<i>Comportamiento orientado a metas</i>	33
3.4.3	<i>Comportamiento guiado por el medio ambiente</i>	34
3.4.4	<i>Adaptabilidad</i>	34
3.4.5	<i>Tendencia a favorecer planes en curso</i>	35
3.4.6	<i>Prevención de conflictos</i>	35
3.4.7	<i>Previsión y velocidad</i>	35
3.5	Consideraciones	36
Capítulo 4. Implementación de un Algoritmo de Redes de Comportamiento		38
4.1	Definiendo un agente	39
4.2.1	<i>Creando la red de comportamiento</i>	40
4.3	Medio ambiente, metas y parámetros globales	41
4.4	El ciclo principal del agente	42
4.5	Ejemplo 1. El robot de Charniak	44
4.5.1	<i>Un agente trabajando</i>	45
4.5.2	<i>Portabilidad</i>	52
Capítulo 5. ABC		52
5.1	Presentación	52
5.2	Ventana principal	53
5.3	Ventana de diseño del agente	56
5.4	Ventana de activación/tiempo	58
5.5	Administración de archivos	59
Capítulo 6. Incorporación de Agentes al medio ambiente		62
6.1	Agentes y medio ambiente	62
6.2	Implementación de un medio ambiente	64
6.3	Percepción	65
6.3.1	<i>Sensor virtual clear</i>	66
6.3.2	<i>Sensor virtual on</i>	66
6.4	Efectores	67
6.4.1	<i>Efector take</i>	67
6.4.2	<i>Efector stack</i>	68
6.5	Modificaciones al algoritmo de selección de acción	69
6.6	Comportamiento del mundo de los cubos	71
Discusión y Trabajo Futuro		72
1.	Discusión	72

2. Trabajo futuro	74
2.1 Proyecto Monots	74
Referencias.....	77

Indice de Tablas

Tabla 1.1	5
Tabla 4.1	52

Indice de figuras

Figura 1.1	6
Figura 1.2	7
Figura 1.3	8
Figura 1.4	9
Figura 2.1	17
Figura 3.1	23
Figura 3.2	24
Figura 3.3	25
Figura 4.1	39
Figura 4.2	40
Figura 4.3	40
Figura 4.4	41
Figura 4.5	42
Figura 4.6	42
Figura 4.7	43
Figura 4.8	44
Figura 4.9	45
Figura 5.1	53
Figura 5.2	57
Figura 5.3	57
Figura 5.4	58
Figura 5.5	59
Figura 5.6	60
Figura 5.7	61
Figura 6.1	63
Figura 6.2	64
Figura 6.3	65

Figura 6.4	65
Figura 6.5	66
Figura 6.6	66
Figura 6.7	68
Figura 6.8	69
Figura 6.9	70
Figura 6.10	70
Figura 1	76

Introducción

Imaginemos una cebra en la sabana africana, inmóvil ante la presencia de un depredador y con suficiente sed como para buscar el próximo estanque tan pronto como el peligro pase. Al mismo tiempo, a miles de kilómetros sobre la cebra, un robot explora la superficie de la luna recolectando muestras de suelo. El robot debe contender con las irregularidades del paisaje lunar en su búsqueda de las muestras que le fueron encomendadas. Estos son dos ejemplos del problema de selección de acción: ¿Cómo seleccionamos las acciones más apropiadas a nuestras circunstancias? ¿Cómo es que un agente autónomo, que actúa en un medio ambiente dinámico puede seleccionar sus acciones, de tal forma que realice sus múltiples metas?

Este trabajo presenta el problema de selección de acción desde una perspectiva basada en el comportamiento. Este enfoque es resultado de recientes investigaciones en el área de agentes autónomos y la búsqueda de arquitecturas que pongan énfasis en un acoplamiento más directo entre la percepción de un agente y sus acciones, en mecanismos distribuidos y descentralizados de selección de acción, en una interacción dinámica entre el agente y su medio ambiente, y en la forma de contender con las limitaciones en los recursos computacionales y el conocimiento incompleto. En este sentido, el mecanismo de selección de acción que presentaremos busca verificar la siguiente hipótesis: la selección de acción puede ser modelada como una propiedad emergente de una dinámica de activación/inhibición entre las diferentes acciones que el agente puede realizar, evitando la presencia de formas globales de control que determinen el comportamiento del agente.

El capítulo uno presenta nuestra definición de agente autónomo, así como una perspectiva de la forma en que este término ha sido usado por la comunidad de Inteligencia Artificial. Así mismo, nos presenta diferentes clasificaciones de agentes autónomos y los problemas a enfrentar en el diseño e implementación de estos sistemas.

El capítulo dos profundiza en el problema de diseño e implementación que nos interesa, la selección de acción. Se presentan diferentes puntos de vista sobre la forma en que la selección de acción en agentes autónomos debería llevarse a cabo y se ofrece una serie de argumentos sobre el porque no usaremos un enfoque basado en planificación, tradicionalmente empleado en Inteligencia Artificial para resolver el problema de selección de acción.

El capítulo tres presenta el mecanismo de selección de acción propuesto por Pattie Maes conocido como Redes de Comportamiento. Discutimos el algoritmo de este mecanismo, así como su modelo matemático. De especial importancia, resulta la exposición de las propiedades que este mecanismo ofrece.

El capítulo cuatro presenta los aspectos relevantes de nuestra implementación del algoritmo de Redes de Comportamiento y ejemplifica en un agente, las propiedades discutidas en el capítulo anterior.

Ante los problemas encontrados en la configuración de parámetros de las Redes de Comportamiento, decidimos implementar una herramienta que constituyera un auxiliar en la experimentación con agentes basados en este mecanismo de selección de acción. El resultado es ABC, un ambiente gráfico que nos permite diseñar y simular agentes basados en Redes de Comportamiento. ABC es presentado en el capítulo cinco.

El capítulo seis, ejemplifica la forma en que podemos utilizar ABC para el diseño de agentes basados en Redes de Comportamiento y su posterior implementación. Se discute el diseño de sensores y efectores, así como las modificaciones a la implementación del algoritmo usada en ABC, para que el agente pueda contender con su medio ambiente.

Finalmente presentamos algunos resultados del uso de ABC en el diseño de agentes autónomos y algunos experimentos con agentes que ejemplifican las propiedades del algoritmo. Presentamos las posibles extensiones a esta herramienta para su aplicación en la simulación de la conducta de forrajeo del mono aullador (*allouta palliata*).

Capítulo 1

Agentes

1.1 ¿Qué es un agente?

La comunidad de Inteligencia Artificial ha propuesto una gran variedad de nociones sobre el término agente que reflejan distintas perspectivas sobre este tema. Para dar un panorama del uso de este término, presentaremos una serie de definiciones donde incrementalmente se plantean nuevas restricciones y posteriormente estableceremos la posición adoptada en este trabajo sobre las propiedades que debe poseer un agente.

La noción menos restrictiva del término agente guarda una estrecha relación con el uso común de esta palabra para referirse a la persona o cosa que actúa o es capaz de actuar: Un agente es un sistema capaz de percibir y actuar en un medio ambiente [Kaelbling 93] [Mataric 94]. En algunos trabajos, esta capacidad de percibir-actuar se identifica como *reactividad*. Observe que la única restricción sobre el sistema es su capacidad para percibir y actuar, así que podríamos nombrar agentes tan diversos como un ser humano, un animal, un robot ó un termostato. No se imponen restricciones sobre el medio ambiente donde percibe y actúa el agente, por lo que una macro de mi procesador de palabras que "percibe" una combinación de teclas que he presionado y "actúa" realizando algunas modificaciones en el formato de mi texto, es también un agente.

El medio ambiente donde actúa un agente puede tener diferentes propiedades [Russell y Norvig 95]. Si un agente tiene acceso al estado completo del medio ambiente a través de sus sensores, decimos que este medio ambiente es *accesible*. Un medio ambiente es *efectivamente accesible* si el agente tiene acceso a todos los aspectos del medio ambiente que intervienen en la selección de sus acciones. Si podemos predecir el próximo estado del medio ambiente a partir de su estado actual y la acción que ha seleccionado el agente para realizar, decimos que el medio ambiente es *determinista*. En un medio ambiente *episódico*, la existencia de un agente se divide en episodios de percepción-acción. La calidad en la selección de acción de un agente en estos medios, depende únicamente del episodio en curso. Si el medio ambiente sufre cambios mientras un agente está deliberando, decimos que es *dinámico*. Si el medio ambiente cambia únicamente por las acciones de un agente decimos que es *semidinámico*. Si existe un número limitado de percepciones y acciones bien definidas, decimos que el medio ambiente es *discreto*, de lo contrario decimos que es *continuo*. Nuestro interés está en agentes que actúan en un medio ambiente dinámico.

Algunos autores ponen énfasis en la capacidad del agente para llevar cabo metas [Maes 95] o satisfacer una agenda [Franklin y Graesser 96]. Las metas de un agente pueden tomar diversas formas: pueden ser *estados* particulares que el agente trata de alcanzar [Newell 81][Maes 89]; pueden ser un *reforzamiento selectivo* o recompensa que el agente trate de maximizar [Maes y Brooks 90]; pueden ser *necesidades internas* o motivaciones que el agente debe mantener dentro de ciertas zonas de viabilidad [Maes 91][Blumberg 94][Tyrell 93].

La noción de una meta como un estado a alcanzar está basado en el concepto de espacio de estados del problema [Newell 81], el cual consiste en un conjunto de estados y un conjunto de operadores que permiten pasar de un estado a otro. La actividad de un sistema con estas metas consiste en buscar el estado deseado en el espacio de estados del problema aplicando los operadores. Una meta de este tipo es un par: $G = \{s_c, S_f\}$, en donde: S es el conjunto de estados posibles, $s_c \in S$ es el estado actual del sistema, $S_f \subset S$ es el conjunto de estados finales y $s_c \notin S_f$ no es un estado final. Una meta de este tipo es una descripción del estado actual del sistema en el espacio de estados del problema y un conjunto de estados llamados estados finales. Implícita en la arquitectura del sistema, está la intención de pasar del estado actual a uno de los estados finales. Cuando el sistema se encuentra en uno de estos estados, el sistema marca esta meta como realizada. La única restricción en el estado actual del sistema es que no sea miembro del conjunto de estados finales como está definido para esa meta. Dos metas son consideradas como equivalentes si su conjunto de estados finales son idénticos. Observe que esta definición de meta excluye a las metas como estados del sistema que se mantienen dentro de ciertas zonas de viabilidad.

Las metas de un agente pueden ser *explícitas*, en cuyo caso pueden ser manipuladas por el agente y están bien definidas en términos de estados en un espacio de estados, ó *implícitas*, las cuales no cumplen con estas características. Un ejemplo de meta implícita, presente en agentes con capacidad de aprendizaje, es la de mejorar su desempeño sobre el tiempo. También es posible que un agente adquiera metas después de entrar en operación, en cuyo caso se llamaran metas *adquiridas*. Estas metas no necesariamente son submetas de las metas pre construidas del agente.

Todo agente guiado por metas incluye un conjunto de estas *pre establecidas*, las cuales guían su comportamiento. Estas metas pueden encontrarse embebidas en la estructura del sistema [Brooks 86], o descritas en términos de la arquitectura empleada [Maes 89]. En ambos casos, las reglas existen cuando el sistema entra en operación y en la mayor parte de los casos no cambian en el tiempo.

Otra distinción entre metas puede ser la forma en qué fueron creadas [Covrigaru y Lindsay 91]: Las metas *endógenas* son creadas por y dentro del agente. Estas metas pueden crearse como una reacción a un estímulo proveniente del medio ambiente o bien como submetas de las metas pre construidas. Las metas *exógenas* tienen su origen fuera del agente y se convierten en metas de éste cuando el agente es diseñado o a través de los sensores del agente. En cualquier caso ya están formuladas como metas. Las metas exógenas incluyen también las metas establecidas por otro agente mediante programación directa.

La forma de realización de las metas también marca diferencias entre estas. Las metas *realizables* son aquellas que tienen bien definido su conjunto de estados finales y una vez que estos estados son alcanzados por el agente se consideran realizadas. Por el contrario, las metas *homeostáticas* se encuentran permanentemente en realización. Esto

es, el agente puede alcanzar el conjunto de estados finales de la meta, sin embargo, cuando las circunstancias del agente hacen que su estado actual difiera del conjunto de estados finales de la meta, intentará realizarla nuevamente.

Una noción de agente en la cual la habilidad para llevar a cabo metas es central, proviene del área de interfaces inteligentes: los agentes son personajes que habitan en el ambiente de una computadora y ayudan al usuario a llevar a cabo su agenda [Laurel 90]. Este sentido de la palabra agente guarda una estrecha relación con otro uso común de la palabra agente, como aquel que actúa en representación de alguien más. Regresando a nuestro ejemplo de la macro, las consideraciones mencionadas no parecen restringir la noción como para dejarla fuera: Si la macro de mi procesador de palabras activa la corrección automática de ortografía, podríamos considerar que la meta del agente es la de eliminar errores ortográficos de un texto.

La autonomía es central en la definición de agentes que nos interesa. Un agente autónomo es aquel que no requiere de la intervención de otros agentes para llevar a cabo sus metas. El uso común de la palabra autonomía se refiere a cierta capacidad de auto gobierno, ó cierto grado de libertad con respecto a un control externo. Un agente autónomo es auto controlado, y por lo tanto debe "saber" que hacer para ejercer control sobre si mismo y debe "querer" ejercitar ese control de cierta forma y no de otra [McFarland 95].

Hay diferencias entre un sistema automático y uno autónomo. Revisemos un ejemplo muy simple: un termostato en un sistema de calefacción. En este sistema una variable registra la temperatura y cuando está baja de cierto valor la calefacción se enciende, al alcanzar cierta temperatura se apaga. Al definir un agente de esta forma seguimos la práctica de teoría de control y teoría de autómatas. Un conjunto de variables describen el estado del agente y proveen información que junto con el conocimiento de las ecuaciones que describen al agente, pueden ser usadas para calcular su funcionamiento futuro en términos de las entradas del agente o el estímulo del medio ambiente. De lo anterior se sigue que para poder controlar un agente de este tipo es necesario conocer sus variables de estado y las reglas que regulan su funcionamiento. Por otra parte el controlador debe querer que el agente se comporte de cierta manera, es decir, debe tener una motivación relevante al respecto. En este caso el agente no es autónomo, pues no tiene autogobierno, ni es independiente de un control externo, aunque puede funcionar automáticamente. En general un agente autónomo no sigue reglas, sino evalúa alternativas, por lo que su comportamiento no depende de su estado y las reglas que definen su comportamiento, sino de su estado y un proceso de evaluación basado en cierto criterio.

La autonomía tiene algunas implicaciones sobre la características de las metas que puede realizar un agente [Covrigaru y Lindsay 91]. Un agente autónomo debe ser capaz realizar un conjunto de metas, algunas de las cuales serán homeostáticas. Debe ser capaz de decidir cual de estas metas debe atender en un momento dado. La existencia de un agente autónomo va más allá del período necesario para el cumplimiento de sus metas, ya que siempre existe la posibilidad de una meta activa a realizar.

Un agente debe poseer un grado considerable de autonomía para exhibir un comportamiento verdaderamente inteligente [Beer 90], esto debido a que su diseñador está imposibilitado para prever todas las circunstancias que el agente deberá enfrentar durante una interacción larga y continua con su medio ambiente. Es deseable que un agente pueda ser capaz de responder a las contingencias que surgen momento a

momento de manera flexible, sin la necesidad de ser instruido explícitamente sobre que hacer en cada situación. La autonomía requiere que el agente esté en funcionamiento continuo: un agente autónomo percibe el mundo, selecciona una acción, la lleva a cabo y repite este proceso permanentemente. Para contar con un agente autónomo corrector de ortografía, éste debería percibir el texto creado, encontrar los errores y corregirlos involucrando lo menos posible al usuario; lo cual al parecer impone serias restricciones a nuestra macro que venía ejemplificando el concepto de agente.

Las propiedades de la noción de agente presentadas hasta ahora, como medio ambiente, metas, autonomía, funcionamiento continuo; parecen ser aceptados uniformemente en Inteligencia Artificial [Shoham 93]. De hecho, junto con habilidad social constituyen lo que se ha llamado noción débil de agente [Wooldrige y Jennings 95] en contraposición a una noción más contenciosa que revisaremos posteriormente.

La habilidad social se refiere a la capacidad de los agentes para interactuar con otros agentes. Para ello se han propuesto lenguajes de comunicación entre agentes [Genesereth y Ketchpel 94]. En el área de interfaces inteligentes se ha planteado además, la necesidad de una capacidad de discurso [Foner 93] ó cierta habilidad de comunicación [Etzioni y Weld 95] que permitan al agente obtener información o asistencia en la realización de sus tareas. Por el momento, la habilidad social no es relevante en nuestro trabajo.

Las nociones provenientes del área de interfaces inteligentes [Foner 93] [Etzioni y Weld 95] parecen ser acordes con la noción débil de agente. Desde esta perspectiva, un agente debe poseer la capacidad de adaptación suficiente para ajustar automáticamente su forma de trabajar, con base en la experiencia, a las preferencias de un usuario y ajustarse de igual forma, a los cambios en su medio ambiente. Para ello, idealmente, un agente debería poseer componentes de aprendizaje y memoria. Un agente debe ser lo suficientemente flexible para no obedecer ciegamente a comandos y estar en posibilidad de modificar las solicitudes que se le hacen, solicitar aclaraciones, o aún negarse a cumplir ciertas tareas. Un agente debe ser robusto, de tal forma que esté expuesto lo menos posible a una falla total de sus funciones. Si la mayor parte de una tarea se lleva a cabo a pesar de las fallas parciales en la funcionalidad del agente, el usuario incrementará su confianza en él. Es deseable que un agente presente la movilidad necesaria para transportarse de una máquina a otra, por si mismo, y a través de diferentes arquitecturas y plataformas.

A pesar de que la noción fuerte de agente es la dominante en Inteligencia Artificial, sólo la describiremos brevemente debido a que los conceptos provenientes de esta noción no son centrales para los intereses de este trabajo. La noción fuerte de agente, implica que además de poseer algunas de las propiedades mencionadas hasta el momento, un agente es un sistema computacional conceptualizado o implementado utilizando conceptos que aplicamos más comúnmente a los seres humanos, por ejemplo, mediante nociones mentalistas [Shoham 93] como conocimientos, creencias, intenciones y obligaciones; ó nociones intencionales [Dennett 87] como creencias y deseos; y aún emociones [Bates 94]. Diferentes argumentos sobre qué tan apropiado resulta emplear estas nociones para describir máquinas han sido propuestos, entre otros por John McCarthy [McCarthy 78] señalando que es legítimo atribuir estas nociones a máquinas, cuando la información que tal atribución expresa sobre la máquina, es la misma que la que expresaría sobre una persona. Esta atribución es útil cuando nos ayuda a entender la estructura de la máquina, su comportamiento pasado o futuro y cómo repararla o mejorarla. Aunque no es necesario, el empleo de estas cualidades mentales, ó cualidades isomorfas a ellas, son

de utilidad para describir brevemente el estado actual de una máquina. La atribución de nociones mentalistas ó intencionales es de mayor utilidad cuando se aplican en entidades cuya estructura no conocemos completamente.

Aunque el trabajo sobre el papel de las emociones en la implementación de agentes [Bates 94] es más cercano a la noción fuerte de agente, debo señalar que los componentes de generación, reconocimiento y expresión de emociones están acoplados a un algoritmo muy semejante al presentado en el capítulo 3 de este trabajo.

Nociones	R	Metas										Medio ambiente		A	Ro	Ap	M	HS	LC	NI
		e	r	n	e	i	p	a	r	h	F	real	virt							
		s	e	e	x	m	r	d	e	o	t	f	c							
Bates 94	•	•		•	•		•	•	•	•	•		•	•	•			•		•
Beer 90	•			•		•	•	•		•	•	•		•	•	•				
Blumberg 94	•			•		•	•	•		•	•		•	•	•					
Brooks 86	•			•		•	•	•		•	•	•		•	•		•			
Brustolini 91	•			•		•	•	•	•		•	•		•	•					
Etzioni y Weld 95	•	•			•		•	•	•		•		•	•	•		•	•		
Foner 93	•	•		•	•	•	•	•	•	•	•		•	•	•	•	•	•		
Franklin y Graesser 96	•	•		•	•	•	•	•	•	•	•		•	•	•	•				
Jennings y Wooldrige 95	•	•			•		•	•	•	•	•		•	•	•		•	•	•	•
Genesereth y Keptchel 94		•			•		•	•	•		•		•				•	•	•	
Kaebing 93	•	•				•	•	•		•	•		•	•	•	•				
Laurel 90	•	•			•	•	•	•	•	•	•		•				•			
Maes y Brooks 90a	•		•			•	•	•		•	•	•		•	•	•	•			
Maes 91	•			•	•		•	•	•	•	•		•	•	•					
Maes 95	•	•	•	•	•	•	•	•	•	•	•	•		•	•	•	•			
Mataric 94	•			•		•	•	•		•	•	•		•	•	•	•			
Rusell y Norvig 95	•	•		•	•	•	•	•	•	•	•		•	•	•	•				
Shoham 93	•	•			•		•	•	•		•		•				•			•

Tabla 1. 1 Resumen de las nociones presentadas en esta sección. Las leyendas en el encabezado son: R=reactividad, est=estados, ref=reforzamientos, nec=necesidades, exp=explícitas, imp=implícitas, pre=preconstruidas, adq=adquiridas, rea=realizables, hom=homeostáticas, F=funcionamiento continuo, A=adaptación, Ro=robustez, Ap=aprendizaje, M=movilidad, HS=habilidad social, LC=lenguaje de comunicación, NI=nociones intencionales. La columna de autonomía se omite ya que todas las nociones, salvo Genesereth y Keptchel, implican esta propiedad.

La tabla 1.1 resume las nociones discutidas hasta el momento y las propiedades presentes en ellas.

Nuestro interés se centra en agentes autónomos que tratan de realizar un conjunto de metas percibiendo y actuando en un medio ambiente. En especial, cómo es que un agente con estas propiedades, puede seleccionar su próxima acción a realizar.

Observe que las restricciones impuestas al medio ambiente no implican que nuestro agente deba actuar en el mundo real. Aunque algunos autores [Brooks 91a] [Brustolini 91] consideran que los agentes deben actuar en este nuestro mundo, nuestra posición,

coincidente con Etzioni [Etzioni y Weld 95] es que un ambiente virtual implementado en una computadora puede ser lo suficientemente dinámico como para experimentar el comportamiento de nuestros agentes.

He omitido intencionalmente hasta este momento, la noción de agente presentada en "La Sociedad de la Mente" [Minsky 86] pues es realmente el concepto agencia el que guarda una estrecha relación con nuestro concepto de agente. Para Minsky los agentes son entidades, que por sí mismas, pueden llevar a cabo tareas muy simples que no requieren de una mente o pensamiento. Es la unión de estos agentes en sociedades, ó agencias, la que puede dar origen a la verdadera inteligencia. Este tema será tratado más a fondo en el segundo capítulo, por el momento sólo mencionaremos que nuestros agentes están compuestos por entidades parecidas a los agentes de Minsky a las cuales llamaremos módulos de habilidad.

1.2 Clases de agentes

Las propiedades que constituyen las diferentes nociones de agente discutidas en la sección anterior, pueden ayudarnos a establecer una clasificación de agentes que nos sea de utilidad. Nuestra concepción de agente como autónomo, implica que al menos cuatro de las propiedades mencionadas deben estar presentes en un sistema para considerarlo como tal: reactividad, autonomía, capacidad de realizar metas y operación continua. La siguiente línea de la clasificación puede ser entre agentes autónomos naturales y artificiales. En el primer grupo estaríamos, entre otros, los seres humanos; y en el segundo los agentes que constituyen el objeto de estudio de este trabajo. Los agentes artificiales pudieran ser clasificados a su vez, con base en el medio ambiente en el que actúan. De esta forma podemos establecer las clases agente robótico, para aquellos agentes artificiales que actúan en el mundo real, y agentes computacionales, para los que actúan en un ambiente virtual implementado en una computadora. La figura 1.1 muestra el árbol correspondiente a esta primer clasificación.

Algunas de las propiedades mencionadas en la sección anterior fueron: adaptación, movilidad, flexibilidad, habilidad social. Es posible establecer nuevas clases de agentes a partir del conjunto de propiedades observadas por un agente. Así podemos hablar de agentes robóticos con capacidad de adaptación y sin ella, por citar un ejemplo.

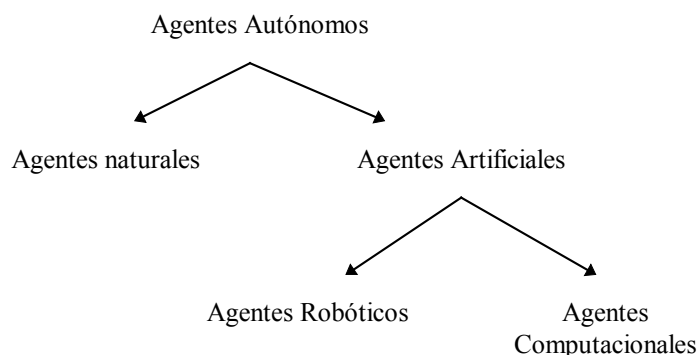


Figura 1. 1 Una posible clasificación de agentes autónomos podría establecerse a partir de este árbol de clasificación

Por supuesto que es posible establecer otros esquemas de clasificación. Brustolini [Brustolini 91], por ejemplo, establece una clasificación a partir de tres categorías: *agentes de regulación*, *agentes planificadores* y *agentes adaptativos* (figura 1.2). Un agente de regulación posee un conocimiento implícito en su estructura que le permite decidir que hacer al confrontar una situación. Las acciones que toma tienden a satisfacer sus motivaciones. Un agente planificador puede además, elaborar planes sobre la secuencia de acciones a ejecutar. Estos agentes pueden a su vez ser clasificados dependiendo del mecanismo de planificación que utilicen. Finalmente, los agentes adaptativos son aquellos que poseen la capacidad de adquirir conocimiento y por lo tanto, mejorar su desempeño.

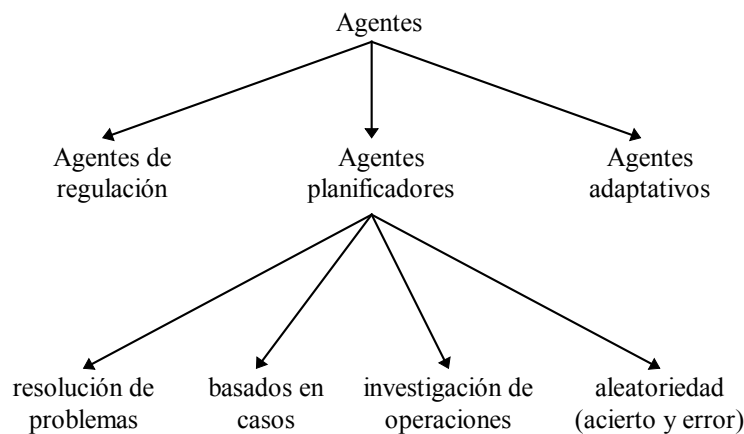


Figura 1. 2 Clasificación de agentes según Brustolini. Las clases no son excluyentes, así que, por ejemplo, podemos hablar de agentes adaptativos planificadores.

Rusell y Norvig [Rusell y Norvig 95] proponen una clasificación a partir de cuatro tipos de agentes: *agentes de reflejo simples*, *agentes que mantienen un estado del mundo*, *agentes basados en metas* y *agentes basados en utilidad* (figura 1.3). Los agentes de reflejo simples deciden que acción tomar únicamente con base en su percepción del medio ambiente. También se les conoce como *agentes tropísticos* [Genesereth y Nilsson 88]. Los agentes que mantienen un estado del mundo hacen uso de un estado interno en su elección de acciones. Estos agentes también son conocidos como *histeréticos* [Genesereth y Nilsson 88]. Mantener cierta información sobre el medio ambiente no siempre es suficiente para decidir qué hacer, por ello los agentes basados en metas deciden sus acciones guiados por la información que éstas proporcionan, como puede ser un estado al que se quiere llegar. Estos agentes pueden elaborar planes para la realización de las metas. Los agentes basados en utilidad están relacionados con la teoría del hombre económico en donde el agente es conceptualizado como una entidad racional que tiende a maximizar una noción cuantitativa llamada utilidad [Hendriks-Jansen 96]. Si un estado del mundo es preferido por un agente con relación a otro es porque, para ese agente, tiene mayor utilidad.

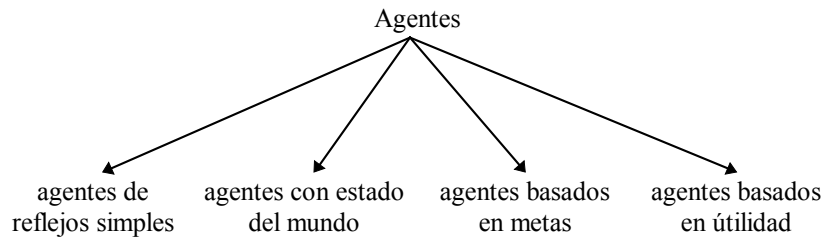


Figura 1. 3 Clasificación de agentes de Rusell y Norvig

Jennings [Jennings y Wooldrige 96] propone una clasificación de agente con base en el servicio que estos pueden prestar:

Un *agente de servicio* es aquel que lleva a cabo tareas bien definidas a petición del usuario, por ejemplo, encuentra el vuelo más barato de México DF a Boston entre el 20 y 23 de julio. Los llamados agentes de información podrían entrar en esta categoría. Oren Etzioni [Etzioni y Weld 95] ha identificado tres tipos de agente de información operando en Internet: Los *agentes guía* se encargan de orientar a las personas que navegan el World Wide Web. Web watcher es un agente que realiza recomendaciones sobre qué liga seguir en un documento de hipertexto y mejora su desempeño observando las reacciones del usuario ante sus recomendaciones. Existen *agentes constructores de índices* de sitios en la red como Lycos, WebCrawler e Infoseek. Estos agentes llevan a cabo de manera autónoma una búsqueda masiva en Internet, almacenando un índice de las palabras contenidas en millones de documentos, que será consultado por los usuarios. Una labor parecida realizan los *agentes localizadores de FAQs*. Los usuarios de Internet tienden a hacer las mismas preguntas, así que los grupos de discusión crean archivos FAQ (pregunta frecuente) con las respuestas a esas preguntas. Los agentes FAQ, como autoFAQ, buscan información en estos archivos a petición de un usuario.

Un *agente predictivo* provee información o servicios al usuario sin un requerimiento explícito para ello, por ejemplo, un agente que monitorea los grupos de discusión del Internet reportándome artículos que podrían ser de mi interés. Un ejemplo de agente predictivo es NewT [Sheth y Maes 93] que opera en el domino de los grupos de discusión news. Después de recibir un entrenamiento basado en ejemplos, NewT lleva a cabo recomendaciones de noticias en los news. Su desempeño mejora con base en la retroalimentación con el usuario. Dentro de este tipo de agentes podemos encontrar a UCEgo [Chin 91] el cual forma parte del sistema UC (UNIX Consultant) cuyo objetivo es auxiliar a un usuario del sistema operativo UNIX en sus tareas. Otro agente predictivo es E-mail [Maes y Kozierok 93], con la particularidad que este aprende sus reglas a partir de la observación del usuario. E-mail opera sobre el correo electrónico prediciendo las acciones de los usuarios sobre los mensajes recibidos. Este agente puede predecir los mensajes que serán leídos y en que orden, cuales serán borrados, donde serán archivados y cuales serán contestados.

Los *agentes cooperativos* [Wooldrige y Jennings 95] podrían constituir otro grupo en esta clasificación. La mayor parte de estos trabajos provienen del área de Inteligencia Artificial Distribuida, donde a diferencia de este trabajo, el énfasis es en el nivel social de los agentes ó el nivel multitudinario [Brooks 91b]. Los agentes en esta área de aplicación solucionan de manera cooperativa problemas como el control de tráfico

áereo, el control de aceleradores de partículas, la administración de plantas generadoras de energía eléctrica y el control de naves espaciales, entre otros.

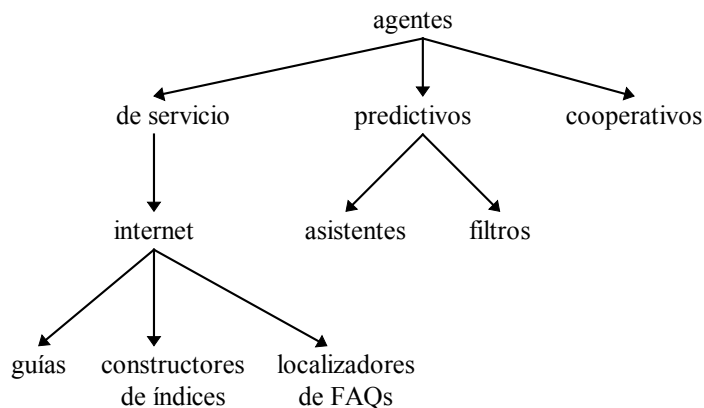


Figura 1. 4 Los dos niveles superiores de este árbol muestran la clasificación de agentes por Jennings, cuyo criterio de clasificación es la forma en que los agentes prestan su servicio.

1.3 Problemas en el diseño e implementación de agentes autónomos

El principal problema a resolver dentro de la investigación sobre agentes autónomos es el desarrollo de arquitecturas que resulten en agentes que demuestren ser adaptativos, robustos y efectivos [Maes 95]. Un agente adaptativo es aquel capaz de mejorar las habilidades para alcanzar sus metas. Un agente robusto es aquel que exhibe una degradación graciosa cuando alguno de sus componentes falla o hay cambios inesperados en el ambiente. Un agente efectivo es aquel que eventualmente realiza sus metas. El desarrollo de estas arquitecturas implica múltiples problemas de diseño e implementación que pueden ser analizados considerando tres niveles de abstracción: micro, macro y multitudinario [Brooks 91b].

El nivel micro estudia cómo es que distintos acoplamientos entre el agente y su medio ambiente se mantienen para alguna tarea en particular. Este nivel se interesa en las formas de percepción que serán necesarias en un agente y su relación con el estado interno y las acciones de éste. Un observador de la interacción del agente con su medio ambiente, debería identificar claramente que en cada momento, el agente realiza una tarea ó un conjunto de ellas en paralelo. Podemos referirnos a estas tareas como las conductas exhibidas por el agente. El nivel micro se ocupa de cada una de estas conductas en particular, independientemente de que el agente pueda exhibir una variedad de ellas en paralelo. Los retos a resolver en este nivel están relacionados con los siguientes aspectos: demostrar o probar la convergencia de un agente, es decir que el agente esté programado de tal forma que su conducta externa realice una tarea en particular con éxito; enfrentar la complejidad del medio ambiente haciendo uso de los aspectos relevantes de las sensaciones recibidas, en lugar de saturar al agente con multitud de datos; dada una tarea en particular, derivar el programa con el que el agente puede exhibir la conducta que bajo una convergencia demostrable, satisface esa tarea.

Una pregunta importante en este nivel es cómo debería emplear un agente su percepción, en particular cómo minimizar la incertidumbre inherente al uso de sensores para modelar el mundo real. Por modelar el mundo entendemos que el agente trata de

poseer alguna representación interna de una realidad objetiva externa. La incertidumbre en los sensores genera problemas al tratar de correlacionar las lecturas de estos con un modelo del mundo existente. Brooks opina que el problema radica en tratar de construir un modelo del mundo, por lo que evitar el uso de esos modelos reduce el problema de incertidumbre. Una alternativa es construir agentes que operen en una estrecha relación con el medio a través de su ciclo de percepción-acción. Otra alternativa es el uso de *índices funcionales* ó *representaciones directas* [Agre y Champan 90]. Estas alternativas nos obligan a enfrentar el problema de la incertidumbre durante el análisis del agente y no cuando éste se encuentra en operación, de tal forma que un agente no tiene que mantener medidas de incertidumbre durante su operación, pero como contraparte, su diseñador está obligado a considerar todos los mundos posibles de actuación del agente, los resultados de sus acciones y las posibles lecturas de percepción que estas acciones generarán.

El nivel micro está relacionado con dos áreas de estudio: la teoría de control y la etología. La teoría de control puede ayudar a demostrar la convergencia de las conductas en el caso de que estas puedan aislarse de otros elementos que intervienen en la interacción del agente con su medio ambiente y tengamos un conocimiento completo sobre los sensores y actuadores que componen al agente. La etología es un enfoque en el estudio del comportamiento animal que combina explicaciones causales y funcionales y se concentra en el comportamiento animal en su medio ambiente natural.

El nivel macro es central en nuestro trabajo. Una vez que hemos dotado a nuestro agente de múltiples comportamientos que manejan una variedad de circunstancias y realizan una serie de tareas, enfrentamos el problema de decidir que tarea o conjunto de tareas deberían estar activos en un momento dado. Este nivel estudia cómo es que las diversas micro percepciones y acciones se integran en un agente. De particular interés es saber hasta donde es posible que un comportamiento complejo emerja de reflejos simples. En este nivel el problema de selección de acción es muy importante. El problema de selección de acción está relacionado con las siguientes preguntas: ¿Cómo puede un agente decidir su próxima acción para progresar en la realización de sus múltiples metas que varían sobre el tiempo? ¿Cómo puede manejar las contingencias y oportunidades que se le presentan? ¿Cómo puede mediar sobre metas conflictivas? ¿Cómo puede manejar la presencia de incertidumbre en sensores y actuadores? ¿Cómo puede reaccionar a tiempo? Siendo la selección de acción el tema central de nuestro trabajo, estas preguntas serán abordadas en profundidad en los próximos dos capítulos.

El nivel multitudinario estudia la forma en que una multitud de agentes interactúan en un mismo medio ambiente. De especial interés resultan las sociedades al estilo de las colonias de insectos sociales, como abejas y hormigas, cuyo estudio plantea las siguientes cuestiones: ¿Cómo podemos predecir el comportamiento de una sociedad de agentes y en consecuencia determinar los efectos diferenciales que pequeños cambios en los individuos producirían a nivel social? ¿Cómo podemos sintetizar la programación de los individuos que conforman una sociedad a partir de la descripción de la tarea que deben realizar? ¿Cómo podemos determinar, dada una tarea, el número y las clases de agentes necesarios para llevarla a cabo? ¿Deben los agentes ser homogéneos para lograr robustez? ¿Bajo qué circunstancias la cooperación entre agentes mejora su desempeño? ¿Qué protocolos pueden establecerse para prevenir la interferencia entre agentes? ¿Cuándo es deseable prevenir esta interferencia? ¿Qué relación guardan la densidad de población y el consumo de recursos del medio ambiente con el comportamiento de la

sociedad? ¿De qué forma puede mejorarse el desempeño de los agentes incrementando la comunicación entre ellos?

El aprendizaje por experiencia como una fuente para mejorar el comportamiento, está relacionado con la capacidad de adaptación de los agentes. y aunque actualmente tiene un mayor impacto en el nivel macro, debería considerarse igualmente importante en los niveles micro y multitudinario. Este problema puede ser definido de la siguiente forma: Dados un agente y el conjunto de acciones que puede llevar a cabo, ciertos datos provenientes de sus sensores y múltiples metas que varían sobre el tiempo, ¿Cómo puede el agente mejorar su selección de acción con base en la experiencia? ¿Cómo puede un agente utilizar una posible retroalimentación con el medio ambiente, posterior a cada una de sus acciones, para mejorar su desempeño? [Maes 95]. Existen diversas formas en que una mejoría en el desempeño puede observarse, por ejemplo, una reducción en tiempo necesario para realizar una meta refleja un mejor desempeño.

Es deseable que el aprendizaje se lleve a cabo de forma incremental, sin la necesidad de fases separadas de actuación del agente y aprendizaje. Las metas del agente deben constituir un sesgo en el aprendizaje. El método de aprendizaje empleado por el agente debe enfrentar satisfactoriamente situaciones de incertidumbre. El método de aprendizaje utilizado debería ser no supervisado para mantener la autonomía del agente. En condiciones donde una parte del conocimiento que requiere el agente es fácil de obtener, el aprendizaje por parte del agente debería darse a partir de ese conocimiento previamente almacenado.

Las arquitecturas de aprendizaje propuestas en la literatura, asumen que el agente posee una colección de acciones preestablecidas y se concentran en aprender la forma adecuada de mediar entre ellas. Algunas de ellas son capaces de aprender nuevas acciones [Mahadevan y Conell 91][Drescher 92]. Los métodos utilizados pueden clasificarse en tres grupos: aprendizaje por reforzamiento [Maes y Brooks 90][Kaebling 93], sistemas de clasificación y aprendizaje de modelos [Maes 92]. En los primeros dos grupos el problema de aprendizaje puede describirse de la siguiente forma: dado un conjunto de acciones que el agente puede llevar a cabo y una señal de reforzamiento, el agente debe aprender una política de acción (establecer una correspondencia situación acción) que maximice la recompensa obtenida de la señal de reforzamiento. En el tercer grupo el agente aprende un modelo sobre el efecto de las acciones en su medio ambiente (como las acciones llevan de una situación a otra).

Para concluir este capítulo quisiéramos mencionar que la meta principal de la investigación en agentes autónomos es la comprensión de los principios y organización que subyacen en el comportamiento adaptativo, robusto y efectivo. Una meta secundaria, en la que se inscribe nuestro trabajo, es el desarrollo de herramientas, técnicas y algoritmos para construir agentes autónomos bajo los principios mencionados, es decir, arquitecturas para modelar agentes autónomos.

Capítulo 2

Selección de Acción

2.1 El problema de selección de acción

En este capítulo abordaremos uno de los problemas centrales al modelar agentes autónomos de cualquier complejidad, el cual está relacionado con la manera en que un agente debería llevar a cabo la elección de una conducta o acción a realizar. El problema de selección de acción puede describirse de la siguiente forma: dados un conjunto de metas que el agente debe realizar, un repertorio de acciones que el agente puede exhibir y datos de percepción específicos, ¿Cómo puede un agente decidir su próxima acción para progresar en la realización de sus múltiples metas? [Maes 95].

Rodney Brooks identifica el problema de selección de acción como el nivel macro de diseño [Brooks 91b], en contraposición al nivel micro relacionado con la implementación particular de las conductas. En su opinión, hay dos diferentes aspectos relativos a la selección de acción: qué acciones son potencialmente correctas a las circunstancias del agente y cómo resolver conflictos entre acciones. Para ello es necesario tomar en cuenta las siguientes consideraciones: las acciones seleccionadas en cierto momento deben ser apropiadas a las condiciones internas y externas del agente; los conflictos entre acciones deben resolverse de tal forma que acciones mutuamente conflictivas no sean activadas simultáneamente y el comportamiento del agente sea coherente; el paso de una acción a otra debe darse suavemente, de tal forma que tengan la persistencia necesaria; finalmente, la selección de acción debe operar de tal forma que la tarea para la que el agente ha sido diseñado sea llevada a cabo.

Brooks consideró dos preguntas centrales en el problema de selección de acción: ¿cuándo debe ser esta selección centralizada y cuando descentralizada? y ¿debe la resolución de conflictos seguir un esquema de prioridades fijas ó uno dinámicamente reconfigurable? Actualmente existen argumentos [Hendriks-Jensen 96] que señalan que por definición, la actividad situada implica mecanismos descentralizados y con resolución de conflictos dinámicamente reconfigurable.

Pattie Maes [Maes 90] resume el problema de selección de acción en la pregunta ¿Cómo hacer lo correcto? Esta pregunta nos sugiere, que el problema de hacer lo correcto está en

relación con el problema de desarrollar un mecanismo de control que opere sobre las posibles acciones de un agente. Esto es cierto pero resulta demasiado general. El punto central es como llevar a cabo el control de las acciones bajo las restricciones impuestas por la operación en un medio ambiente dinámico y complejo, con recursos computacionales limitados. De ahí que la decisión adoptada por el agente no pueda ser óptima y sin embargo es deseable que sea correcta.

Una selección de acción correcta debe estar orientada al cumplimiento de las metas del agente. Debe favorecer aquellas acciones encaminadas a la realización de las metas, y en particular, favorecer acciones que contribuyen con el mayor número de ellas. Debe ser oportunista, esto es, debe aprovechar las condiciones que le son favorables en el medio ambiente. Debe favorecer las acciones que son relevantes a la situación del agente en su medio ambiente y responder favorablemente a cambios imprevistos en este último. Debe favorecer la realización completa de aquellas secuencias de acciones que llevan al cumplimiento de las metas, en especial, debe evitar que la elección de acciones que pertenecen a secuencias distintas imposibilite la realización de las metas. Debe ser previsor y evitar que las acciones seleccionadas conduzcan al agente a situaciones donde le sea imposible realizar sus metas. Debe enfrentar satisfactoriamente el problema de metas y acciones conflictivas. Debe ser robusta, es decir, ante fallas en algunos elementos de la funcionalidad del agente, debe ser capaz de favorecer aquellas acciones que aún pueden llevarse a cabo y realizar el mayor número de metas posibles en esas condiciones.

El resto de este capítulo está dedicado a revisar algunos mecanismos de selección de acción basados en los principios mencionados hasta ahora y tomando en cuenta la naturaleza de las metas y sensores que utilizan, así como los mecanismos de arbitraje entre conductas y fusión de comandos. Es necesario advertir que estos mecanismos y las ideas en que están inspirados constituyen una perspectiva distinta del enfoque basado en planificación, tradicionalmente utilizado en Inteligencia Artificial para decidir las acciones que lleva a cabo un agente.

El planteamiento del problema de selección de acción desde la perspectiva de planificación es: dados un estado inicial del medio ambiente, un conjunto de operadores disponibles y sus efectos en el modelo del medio ambiente y un estado meta, encontrar la secuencia de operadores, el plan, que transforme el estado inicial en el estado meta [Nilsson 71]. Observe que existe cierta similitud entre esta definición del problema de selección de acción y la breve revisión de las metas como estados [Newell 81] presentada en el capítulo anterior.

Un proceso de planificación tradicional busca garantizar que un plan óptimo o cercano al óptimo será encontrado si es que existe. Este objetivo descansaba en los siguientes supuestos [Marks et.al. 88]: El medio ambiente es estable, y se comportará tal como se ha proyectado. El tiempo empleado en la planificación es independiente al tiempo que será necesario para la ejecución del plan, de tal forma que la eficiencia del planificador no tiene efectos secundarios en la factibilidad del plan construido. El planificador dispone de información completa y la ejecución del plan se dará sin errores. Cualquier plan correcto permanecerá correcto y podrá ser ejecutado posteriormente.

En un medio ambiente *dinámico*, en especial en el mundo real, los supuestos mencionados no son verdaderos. Esta situación ha dado origen a una serie de consideraciones y

problemas a resolver en el área de planificación. La complejidad del problema es una de ellas. Si un planificador realiza una búsqueda del plan correcto y seguro proyectando los efectos de todas sus acciones posibles, la complejidad computacional de este proceso puede ser muy elevada. De hecho Chapman [Agre y Chapman 90] ha demostrado que éste es un problema NP-duro e indecidible en el peor de los casos. Un segundo problema está relacionado con la ejecución del plan. Un plan que parecía correcto al momento de ser construido puede resultar incorrecto durante su ejecución. Esto se debe a que en un medio ambiente *dinámico*, las acciones del agente no son el único factor de cambio presente, así que las condiciones en el medio ambiente pueden haber cambiado al momento de ejecución, invalidando las precondiciones del plan. Por otra parte, si la planificación y la ejecución del plan son dos procesos independientes llevados a cabo secuencialmente por un mismo agente, el tiempo empleado en el proceso de planificación tiene efectos en la factibilidad de ejecución del plan. Así mismo, un agente que sigue esta estrategia no puede exhibir un comportamiento realmente oportunista.

Existen diversas propuestas para enfrentar los problemas de la planificación en ambientes *dinámicos*. Algunas de ellas tratan ganar flexibilidad a través del módulo ejecutor del plan. Estos planificadores conocidos como *intercalados* o *incrementales* [Wilkins 85], pueden alternar el control entre el planificador y el módulo de ejecución cuando este último está en problemas. La forma más común de saber que algo anda mal con el plan es monitoreando las precondiciones de los operadores. En caso de que estas no sean ya verdaderas, se regresa el control al planificador para trabajar en un nuevo plan. Aunque este enfoque resuelve el problema relacionado con cambios imprevistos en el medio ambiente, las dificultades asociadas a la construcción del plan y al aprovechamiento de oportunidades siguen siendo las mismas. Un segundo enfoque que busca mayor flexibilidad en la planificación es la *planificación reactiva* [Firby 87] en donde el módulo de ejecución dispone de una biblioteca de planes generada externamente. La idea es que el módulo ejecutor elija los planes de acuerdo a las circunstancias que se presentan en el medio ambiente. El problema con este enfoque parece ser una tensión entre las condiciones a corto plazo (reactividad) y los propósitos de largo plazo (planificación). Esto es, si los planes son muy cortos, difícilmente podríamos identificarlos como tales y por el contrario, si son muy largos, presentan todos los problemas de la planificación tradicional. Una combinación de *planificación reactiva* y *planificación basada en casos* es presentada en [Marks, et.al. 88].

Otra alternativa que hace un uso muy diferente del plan fue propuesta en Pengi y Sonja [Agre y Chapman 90], dos agentes cuya tarea es jugar Pengo. Pengi no construye plan alguno, pero aprovecha las contingencias y oportunidades que se presentan en su medio ambiente para improvisar diferentes maneras de lograr sus metas, en lo que se ha llamado *actividad situada*. Pengi utiliza una *representación directa* del mundo o *indexación funcional* en donde el agente establece relaciones causales con los objetos en su medio ambiente. De esta forma, Pengi confronta su medio ambiente a través de una constante interacción con éste, en lugar de construir y manipular un modelo del mundo. Agre y Chapman reconocen que hay muchos problemas en donde la *actividad situada* no es suficiente y proponen un uso del plan en donde éste no es visto como un programa a ejecutar que determina el comportamiento del agente, sino como una especie de dispositivo mnemónico, una fuente de recursos para mejorar el desempeño. Desde esta perspectiva el

plan deja de tener un papel central que ahora ocupa el agente, el cual tiene que llevar a cabo un esfuerzo interpretativo constante para determinar el “significado” de cada paso del plan bajo las condiciones presentes en el medio ambiente. Sonja, al igual que Pengi, hace uso de la actividad situada, pero a diferencia de éste, puede usar instrucciones (el plan) cuando estas le son dadas. Estas instrucciones no son interpretadas por Sonja como programas a ejecutar, sino como información que puede emplear en su *actividad situada*.

La actividad como el resultado de una interacción entre el agente y su medio ambiente, y no de un mecanismo de control, es el punto común entre esta última alternativa a la planificación tradicional y las arquitecturas de selección de acción que revisaremos en el resto de este capítulo.

2.2 La Arquitectura de Subsumción

Roodney Brooks propuso una arquitectura computacional [Brooks 86] que establece una estrecha conexión entre los elementos de percepción y acción de un agente. Esto se logra construyendo incrementalmente una serie de niveles que establecen una relación entre los elementos de percepción del agente y sus acciones. Cada nivel se conoce como espectro de habilidades y constituye una especificación informal de la clase de comportamiento que deseamos el agente exhiba en los medios ambientes que enfrente. Observemos que cada espectro de habilidades de un agente puede hacer algo por sí mismo. La idea central es que podemos construir espectros de habilidades, de tal forma que el comportamiento de un agente pueda mejorarse agregando un nuevo espectro de habilidad a los ya existentes, sin que estos deban ser modificados.

La unidad de construcción de la Arquitectura de Subsumción es una máquina de estado finito aumentada (Figura 2.1). Además de los elementos de una máquina de estado finito, tales como un alfabeto finito de entradas, un conjunto finito de salidas y una función de transición, una máquina de estado finito aumentada (AFSM) cuenta con elementos temporalizadores y un conjunto finito de registros, cada uno de los cuales recibe una de las entradas de la máquina de estado finito. Una AFSM puede producir una salida y/o cambiar de estado al recibir un mensaje de entrada o cuando cierto tiempo ha transcurrido, para esto los elementos temporalizadores se utilizan como cuentas regresivas. Los registros de una AFSM reciben señales que provienen de la salida de alguna AFSM o bien, de algún sensor del agente. Cada vez que un mensaje llega a un registro, reemplaza el contenido de éste. Una AFSM puede suprimir la entrada de otra, conectando su salida al registro de dicha entrada, en este caso el mensaje en el registro es sustituido por el mensaje supresor. Una AFSM puede inhibir la salida de otra conectando su salida a la salida de la otra AFSM. En este caso el mensaje original no se pierde, pero queda sin efecto durante un tiempo predeterminado. Estos mecanismos constituyen la esencia de la resolución de conflictos y establecen la prioridad de las conductas que componen un agente. Las AFSM no pueden compartir estados y en particular, una AFSM no puede leer los registros de otra. Temporalizadores y mensajes actúan de manera asíncrona. Un grupo coordinado de AFSM constituye una conducta y un conjunto de conductas establece un espectro de habilidades.

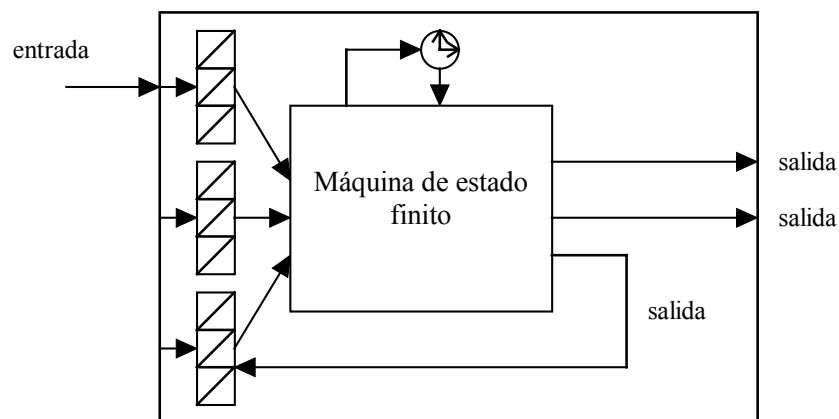


Figura 2. 1 Una máquina de estado finito aumentada (AFSM) está compuesta por una máquina de estado finito, un conjunto de registros y elementos temporalizadores.

Utilizando la arquitectura de Subsumción, Rodney Brooks y su equipo en MIT han construido diversos robots autónomos [Brooks 90]. Allen es el primero de ellos. Su objetivo es explorar su medio ambiente dirigiéndose hasta los lugares más distantes que perciba. Si un objeto se aproxima hacia Allen, éste huye de él evitando los obstáculos que encuentra en su camino. Un robot más complicado es Herbert cuyo objetivo es recoger las latas de refresco vacías que pudiera encontrar en los escritorios de un área de oficinas. Genghis es un robot de seis piernas que camina en terreno accidentado bajo el control de la arquitectura de subsumción.

Uno de los robots más interesantes que se han construido con la arquitectura de Subsumción es Toto [Mataric 89]. Toto fue construido para demostrar que la falta de estructuras de datos en la arquitectura de subsumción, no constituyen un obstáculo para hacer uso de mapas de navegación. Un nivel de habilidad de Toto le permite navegar evitando colisiones, seguir paredes y corredores de tal forma que puede explorar su medio ambiente. Un segundo nivel de habilidad trata de reconocer marcas particulares en el medio ambiente, como puertas y corredores. Cada vez que una marca nueva es reconocida, la conducta que reconoció la marca establece una correspondencia con la marca. Las conductas que corresponden a marcas adyacentes físicamente, establecen una relación de vecinos a través de ligas. Este proceso crea un grafo cuyos nodos son conductas, elementos computacionales especificados como AFSMs. Cuando el robot se mueve por su medio ambiente, las conductas tratan de identificar el lugar donde está. Los nodos que creen estar en correspondencia con el lugar donde se encuentra el robot incrementan su actividad. Un usuario puede pedir al robot que active conductas del tipo “ve a cierto lugar”, lo cual dispara un mecanismo de propagación de activación vía las ligas de vecino a partir de la conducta activada, lo cual hará que el robot alcance el lugar indicado.

La Arquitectura de Subsumción constituye un ejemplo de los mecanismos de selección de acción conocidos como *redes construidas manualmente*. El nombre se debe a que el diseñador debe establecer una a una, las relaciones entre las conductas (en este caso supresión e inhibición) para obtener el comportamiento deseado del agente. Las metas del agente, en estos mecanismos de selección de acción están implícitas en la estructura del

agente y pueden ser de muy diversa naturaleza. Los datos de percepción son de igual forma ilimitados. El arbitraje entre conductas se realiza por medio de los mecanismos de inhibición y supresión. Estos mecanismos no contemplan la fusión de comandos, esto es, no es posible que dos conductas se activen de manera simultánea.

Las *redes construidas manualmente* presentan algunas desventajas. La principal de ellas está relacionada con la escalabilidad. A medida que la complejidad de un agente aumenta, resulta muy difícil establecer las relaciones entre las conductas que componen el agente. Por otra parte, las metas del agente no pueden variar sobre el tiempo ya que se encuentran especificadas en la estructura de éste.

Otros ejemplos de este tipo de arquitecturas son las presentadas en [Beer 90] y [Chapman 92].

2.3 Redes de comportamiento

Las redes de comportamiento fueron propuestas por Pattie Maes en [Maes 89, 90a, 90b, 91] y constituyen un mecanismo no jerárquico, descentralizado y dinámicamente reconfigurable de selección de acción. Los nodos de una red de comportamiento representan las habilidades o acciones que un agente puede exhibir. Estos nodos están conectados por ligas de sucesor, predecesor y conflicto que posibilitan una dinámica de activación / inhibición entre las acciones de manera similar al mecanismo de propagación usado en Toto (ver sección anterior).

Cada habilidad o acción tiene asociado un nivel de activación, el cual es un número real, y un conjunto de condiciones que deben ser verdaderas para que la habilidad sea ejecutable. Las habilidades ejecutables cuyo nivel de activación rebasa cierto umbral son activadas. Sólo una conducta puede activarse a un mismo tiempo. Después de activada, el nivel de activación de una habilidad se vuelve cero.

Una habilidad que es ejecutable propaga activación a sus sucesores, promoviendo su ejecución y estableciendo de esta forma, secuencias de acciones. Una habilidad no ejecutable propaga activación a sus predecesores, buscando que al activarse estos, sus condiciones, por el momento falsas, se vuelvan verdaderas. Todas las habilidades decrementan el nivel de activación de las habilidades con quienes tienen conflictos.

La percepción se realiza mediante un conjunto de sensores virtuales, los cuales abstraen, a partir de los datos provenientes de los sensores reales, una serie de condiciones perceptuales binarias, tales como “pared a la izquierda”, “obstáculo a un metro”, etc. La ocurrencia de una condición perceptual incrementa el nivel de activación de los módulos donde aparece como condición.

Este mecanismo de selección de acción incorpora un conjunto de metas, las cuales incrementan el nivel de activación de las habilidades que pueden llevarlas a cabo. Es posible relacionar un peso con cada meta para convertirlas en motivaciones. Este peso representa la fuerza de la motivación y se incrementa con el paso del tiempo. Las conductas que satisfacen una motivación se llaman *consumatorias* y su activación reduce la fuerza de la motivación asociada. El resto de las habilidades se llaman *apetitivas* y su papel es contribuir a que las conductas consumatorias sean ejecutables.

El capítulo tres describe detalladamente este mecanismo de selección de acción, que será utilizado en la implementación de un simulador de agentes basados en redes de comportamiento.

Las redes de comportamiento constituyen un ejemplo del tipo de mecanismos conocido como *redes compiladas*. Este tipo de mecanismos facilita la síntesis de los agentes al construir de manera automática el circuito encargado del arbitraje entre conductas. Un diseñador de agentes debe especificar en algún formalismo las metas que el agente debe realizar, así como el conjunto de acciones que el agente puede llevar a cabo, en término de sus precondiciones y efectos esperados. A partir de esta especificación, el mecanismo de selección de acción generará un circuito que implementa el comportamiento deseado.

Las metas se definen explícitamente en la especificación del agente, por lo que estas pueden variar en tiempo de ejecución sin necesidad de recompilar la especificación del agente. Generalmente, los datos de percepción son binarios, lo cual, junto con el hecho de que cada uno de estos mecanismos de selección de acción utilizan un modelo de relevancia de acciones en particular, limita el tipo de agente que es posible implementar.

Otros ejemplos de redes compiladas son Rex/Gaps [Kaebling y Rosenschein 90] y los árboles teleo-reactivos [Nilsson 92].

2.4 Hamsterdam

Al tratar de implementar *animats*, animales artificiales que habitan medios ambientes dinámicos y complejos, Bruce Blumberg encontró que muchos de los mecanismos de selección de acción propuestos presentaban serias deficiencias al aumentar la complejidad del agente en términos de la cantidad de metas y acciones diferentes que se debían considerar. Por ello, propuso una nueva arquitectura, con una fuerte inspiración etológica, a la que llamó Hamsterdam [Blumberg 92]. En particular, él demostró que una estructura jerárquica de conductas, que permitiera cierto intercambio de información entre estas, podría ayudar considerablemente a decidir las acciones relevantes para un animat.

En su modelo, las acciones ó actividades están organizadas en una jerarquía con las acciones más generales en la parte superior y las más específicas en la inferior. Las acciones se representan como nodos de un árbol y cada nodo puede tener cero ó más hijos. El mecanismo de selección de acción consiste en determinar que hoja del árbol debe activarse a un tiempo dado, partiendo de la raíz del árbol. Los hijos de una acción se inhiben mutuamente y sólo uno de ellos puede estar activo a un tiempo dado. Si el nodo activo es una hoja, esto se ve reflejado en alguna acción del agente, de otra forma los hijos del nodo compiten por activarse y este proceso se repite hasta llegar a una hoja. La base de esta competencia es un valor asociado a cada acción del agente. Este valor está en función de mecanismos de disparo, variables endógenas y otros factores dependientes del dominio. Hamsterdam ofrece mecanismos explícitos de fatiga, lo cual es de utilidad al manejar persistencia en las acciones del agente. Los mecanismos de disparo reciben como entrada los datos de percepción del agente, identificando eventos y objetos relevantes en el medio ambiente. Su salida está en función de esta entrada y es expresada como una variable continua. El estado interno del agente se modela vía variables endógenas que se expresan como variables continuas cuyo valor está en función del valor de la variable endógena al

tiempo anterior, el valor de las actividades que afectan a la variable y un factor de cambio asociado, así como de una función de factores dependientes de dominio. Una acción puede depender de cualquier número de variables endógenas y mecanismos de disparo y en contra parte, estos pueden compartirse por cualquier número de acciones.

Las acciones perdedoras que se encuentran en la misma rama de la acción ganadora pueden emitir “recomendaciones” que están conformadas por el nombre de un comando y un peso asociado. Pesos negativos representan recomendaciones en contra de ese comando y positivos a favor. La acción correspondiente a una hoja ganadora, tomará en cuenta estas recomendaciones al exhibir su comportamiento.

Hamsterdam es un ejemplo de la clase de mecanismos de selección de acción conocidos como *redes jerárquicas construidas manualmente*. Como mencionamos, estos mecanismos están inspirados por modelos propuestos por etólogos como Lorenz y Tinbergen. Las metas o motivaciones que se emplean en estos mecanismos de selección de acción son mucho más complejas que en los modelos anteriores, de igual manera, la forma en que se utilizan los datos de percepción es mucho más sofisticada. El mecanismo de arbitraje entre conductas se basa en niveles de abstracción más altos para premiar acciones primitivas.

Otros ejemplos de redes jerarquizadas son los propuestos en [Tyrell 93] y [Rossenblatt y Payton 89].

2.5 La Sociedad de la Mente

La Sociedad de la Mente [Minsky 86] no es propiamente un mecanismo de selección de acción, sino una teoría de la mente con la que guardan cierta relación los orígenes de la arquitectura de subsumción y las redes de comportamiento.

En una aproximación sintética, Minsky intentará convencernos que es posible construir sistemas que exhiban verdadera inteligencia, a partir de pequeñas partes que no poseen una mente propia. A estas pequeñas partes las llama agentes y podemos pensar que son procedimientos que realizan tareas muy sencillas, que no requieren de una mente o pensamiento. Es la unión de estos agentes en sociedades lo que puede llevarnos a obtener la inteligencia. Debemos señalar que agentes y agencias son entidades abstractas cuya posible implementación no es abordada por Minsky.

Imaginemos un niño jugando con bloques de madera e imaginemos que la mente de este niño contiene una colección de mentes más pequeñas, a las cuales llamaremos agentes mentales. Imaginemos que uno de estos agentes llamado *constructor* tiene el control y su tarea es construir torres con los bloques de madera. Construir la torre es una tarea demasiado complicada para un solo agente, por lo que *constructor* debe pedir ayuda a otros agentes. De hecho, encontrar un bloque de madera y colocarlo en la cima de la torre es una tarea complicada que necesita de la intervención de otros agentes. Un agente llamado *agregar*, con esta funcionalidad, debería por lo tanto hacer uso de agentes para buscar, obtener y colocar los cubos de madera. A su vez los agentes encargados de colocar y obtener cubos de madera, harían uso de agentes como tomar, mover y soltar cubos de madera. Este ejemplo nos muestra como una tarea compleja puede descomponerse en tareas más sencillas, las cuales son realizadas por agentes. Observe que los agentes de Minsky son

muy semejantes a las conductas de la arquitectura de subsumción y a las habilidades de las redes de comportamiento.

Una vez establecida la función de cada uno de estos agentes es necesario entender la interacción que se da entre estos y cómo esas interacciones locales se combinan para dar origen a la funcionalidad del sistema. Al respecto, Minsky propone un cerebro-B con influencia sobre un cerebro-A que interactúa con el mundo. El cerebro-A está compuesto de agentes-sensores que perciben el mundo y agentes-motores que actúan sobre él. El cerebro-B únicamente tiene contacto con los agentes del cerebro-A y está compuesto por ejecutivos que dirigen, o al menos tiene influencia en la actividad del cerebro-A. Algunos ejemplos de esta relación son: si A parece entrar en un ciclo sin fin, B lo obliga a tratar algo diferente para salir del ciclo; si A hace algo del agrado de B, B hace que A lo recuerde; si A se encuentra demasiado involucrado con los detalles, B lo obliga a tomar una perspectiva de más alto nivel y viceversa. La noción de cerebro-B provee un mecanismo de control abstracto de alto nivel. Las redes de comportamiento [Maes 89] puede considerarse un mecanismo de control de bajo nivel como el cerebro-A.

En opinión de Minsky, los cerebros se componen jerárquicamente. En la base tenemos los agentes con su propia jerarquía. En el siguiente nivel encontramos sociedades, organizaciones de agentes. En el siguiente nivel encontramos capas de sociedades. La sociedad de la mente se compone como una serie de capas de sociedades. Cada capa aprende a explotar las habilidades adquiridas por la capa previa. Esta organización es muy semejante a los espectros de habilidad de la arquitectura de subsumción [Brooks 86].

Más allá de las coincidencias y diferencias mencionadas, la importancia de la sociedad de la mente, en relación con los mecanismos de selección discutidos, radica en su visión de la mente como una colección de procesos no unificados y heterogéneos cuya interacción hace posible la emergencia de la verdadera inteligencia.

Capítulo 3

Redes de Comportamiento

3.1 ¿Cómo hacer lo correcto?

Hemos elegido las redes de comportamiento para implementar un mecanismo de selección de acción y un ambiente de simulación que nos permita experimentar con diversos agentes autónomos y la forma en que estos realizan sus metas eligiendo correctamente sus acciones.

Las propiedades exhibidas por las redes de comportamiento fueron un factor decisivo en esta elección. Este mecanismo exhibe un comportamiento guiado por metas y relevante a la situación del agente, es adaptativo, tolerante ante fallas, persistente en el plan de acción en curso, reactivo y rápido [Maes 89,90b,90c,91] [Franklin95]. Además, las redes de comportamiento son un mecanismo de selección de acción descentralizado y dinámicamente reconfigurable, lo que las hace viables para implementar agentes con actividad situada [Hendriks-Jansen 96].

En este capítulo presentaremos un algoritmo de activación / inhibición en una red de comportamiento, así como el modelo matemático de este mecanismo de selección de acción. Posteriormente, estudiaremos la forma en que una red de comportamiento puede ser configurada a través de sus parámetros globales y terminaremos presentando algunas limitaciones presentes en este mecanismo.

Es importante que durante la lectura de este capítulo tengamos presente que este mecanismo de selección de acción busca verificar la siguiente hipótesis: la selección de acción puede ser modelada como una propiedad emergente de una dinámica de activación / inhibición entre las diferentes acciones que el agente puede realizar, evitando la presencia de módulos burocráticos que funcionen como formas globales de control determinando que acción resultará activada o inhibida.

3.2 Descripción del algoritmo

El algoritmo para elegir la acción correcta basado en redes de comportamiento define un agente como un conjunto de habilidades o destrezas representados como módulos de habilidad. Un módulo de habilidad m_i puede ser descrito por la tupla $(c_i, a_i, d_i, \alpha_i)$. La lista c_i

contiene las precondiciones que deben ser observadas para que el módulo de habilidad m_i se active. Las listas a_i y d_i representan los efectos esperados al activarse el módulo de habilidad m_i : La lista a_i contiene las condiciones que se espera sean verdaderas después de activarse el módulo y la lista d_i las que se espera dejen de serlo. El valor α_i representa el nivel de activación que ha acumulado el módulo. La figura 3.1 representa un módulo de habilidad. Como puede observar, los módulos de habilidad se asemejan a los operadores empleados en los sistemas de planificación clásicos [Fikes y Nilsson 71].

Cuando todas las precondiciones de un módulo de habilidad se observan verdaderas, decimos que éste es *ejectuable*. Ser ejecutable es una condición necesaria para que un módulo de habilidad pueda activarse.

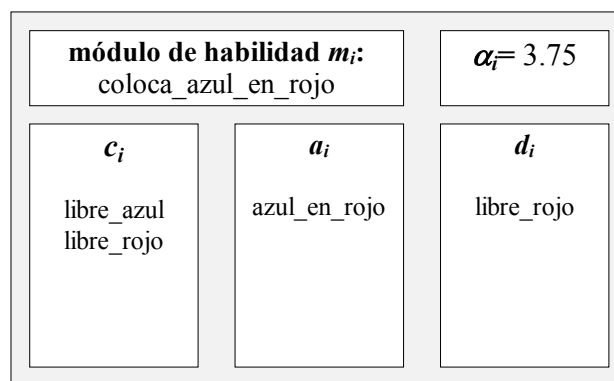


Figura 3.1 Imagine un agente que actúa en un medio ambiente donde puede encontrar cubos de diferentes colores. La idea es que este agente sea capaz de construir torres con esos cubos. Una destreza deseable en este agente es que sea capaz de colocar un cubo sobre otro. En este caso el módulo de habilidad m_i se encargará de colocar un cubo azul sobre uno rojo. Para ello es necesario que el cubo azul y el rojo estén libres. Estas precondiciones se encuentran en la lista c_i . Si el módulo de habilidad m_i se activa, el resultado esperado de sus acciones es que el cubo azul este sobre el rojo (a_i) y que el cubo rojo deje de percibirse como libre (d_i). El nivel de activación acumulado en este módulo es de 3.75 (α_i).

Los módulos de habilidad que conforman un agente se organizan en una red, relacionados por medio de tres tipos de ligas: *sucesor*, *predecesor* y *conflicto*. La composición de cada módulo de habilidad m_i en términos de c_i , a_i , y d_i definen esta red de la siguiente forma:

Hay una *liga de sucesor* (figura 3.2) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in a_i \cap c_j$. Decimos que el módulo de habilidad m_i tiene como sucesor al módulo m_j debido a que las precondiciones p de m_j serían verdaderas de activarse m_i , por lo que existe la posibilidad de que m_j se active después de m_i . Observe que es posible que existan más de una liga del mismo tipo entre dos módulos de habilidad.

Hay una *liga de predecesor* (figura 3.3) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in c_i \cap a_j$. Decimos que el módulo de habilidad m_i tiene como predecesor al módulo m_j , debido a que las precondiciones p de m_i serían verdaderas

de activarse m_j , por lo que existe la probabilidad de que m_j se active antes de m_i . Observe que por cada liga predecesor, existe una de sucesor en sentido contrario.

Hay una *liga de conflicto* (figura 3.4) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in c_i \cap d_j$. Decimos que el módulo m_i tiene como módulo conflictivo al módulo m_j debido a que las precondiciones p de m_i dejarían de observarse verdaderas de activarse m_j , por lo que m_i no podría activarse. Observe que pueden existir conflictos mutuos entre dos módulos.

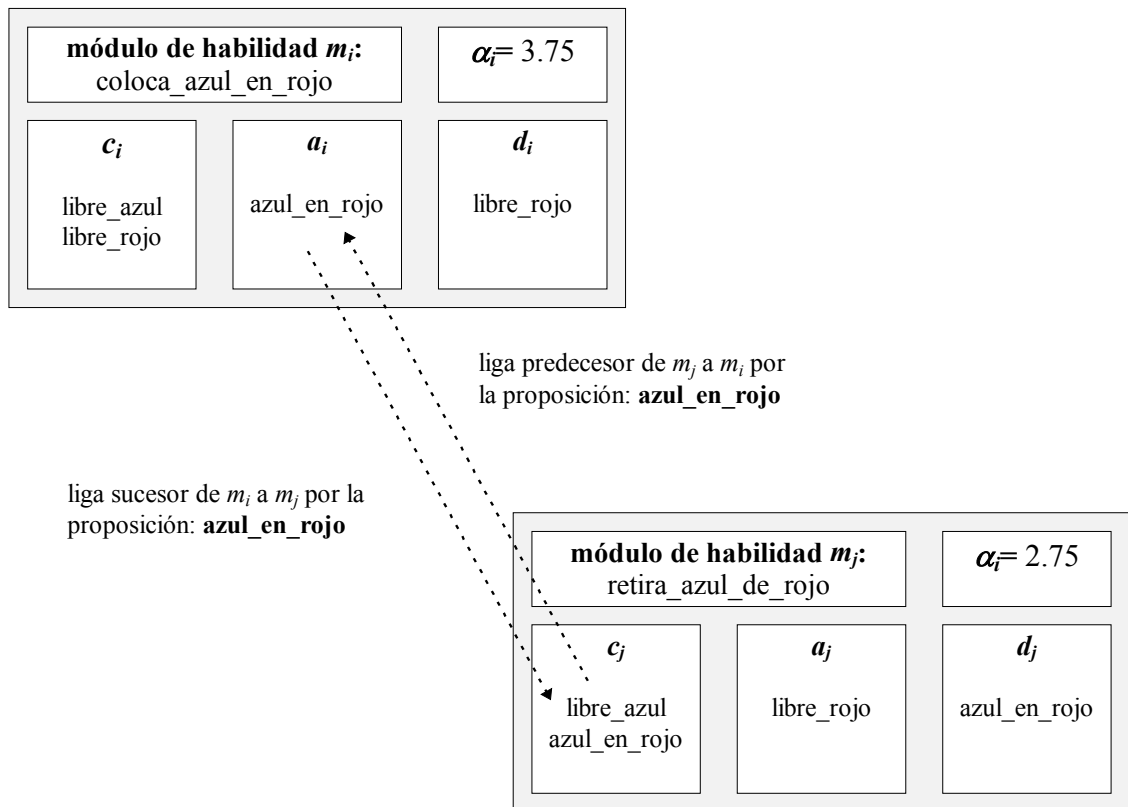


Figura 3.2 La existencia de una liga de sucesor implica la existencia de una liga de predecesor en sentido contrario. En este caso, hay una liga de predecesor del módulo m_j al módulo m_i por la proposición azul_en_rojo. Observe que también la proposición libre_rojo define ligas entre estos dos módulos. Por claridad, únicamente se muestran las ligas originadas por azul_en_rojo.

La idea detrás de estas ligas es que permitan una dinámica de activación/inhibición entre los módulos de habilidad que componen un agente. Esta dinámica tiene efecto sobre el nivel de activación de los módulos, de tal forma que después de un tiempo, aquellos módulos que representan la acciones adecuadas a la situación del agente y las metas que debe llevar a cabo acumulen una mayor activación. Si el nivel de activación de un módulo

ejecutable ha rebasado cierto umbral, entonces se activará desplegando sus habilidades. En esta dinámica de activación/inhibición intervienen los siguientes factores:

Activación proveniente del medio ambiente. Decimos que un módulo de habilidad m_i está en correspondencia parcial con el medio ambiente, si al menos una de sus precondiciones c_i se observa verdadera. Los módulos de habilidad que están en correspondencia parcial con el medio ambiente se estimulan, incrementando su nivel de activación, con la intención de que los módulos que son relevantes a una situación en el medio ambiente sean los que acumulen un mayor nivel de activación. La situación actual del medio ambiente se obtiene utilizando un conjunto de sensores virtuales, los cuales utilizan los datos generados por sensores reales para determinar cuando una proposición es verdadera. La situación actual puede considerarse como la unión de las observaciones de todos los sensores virtuales.

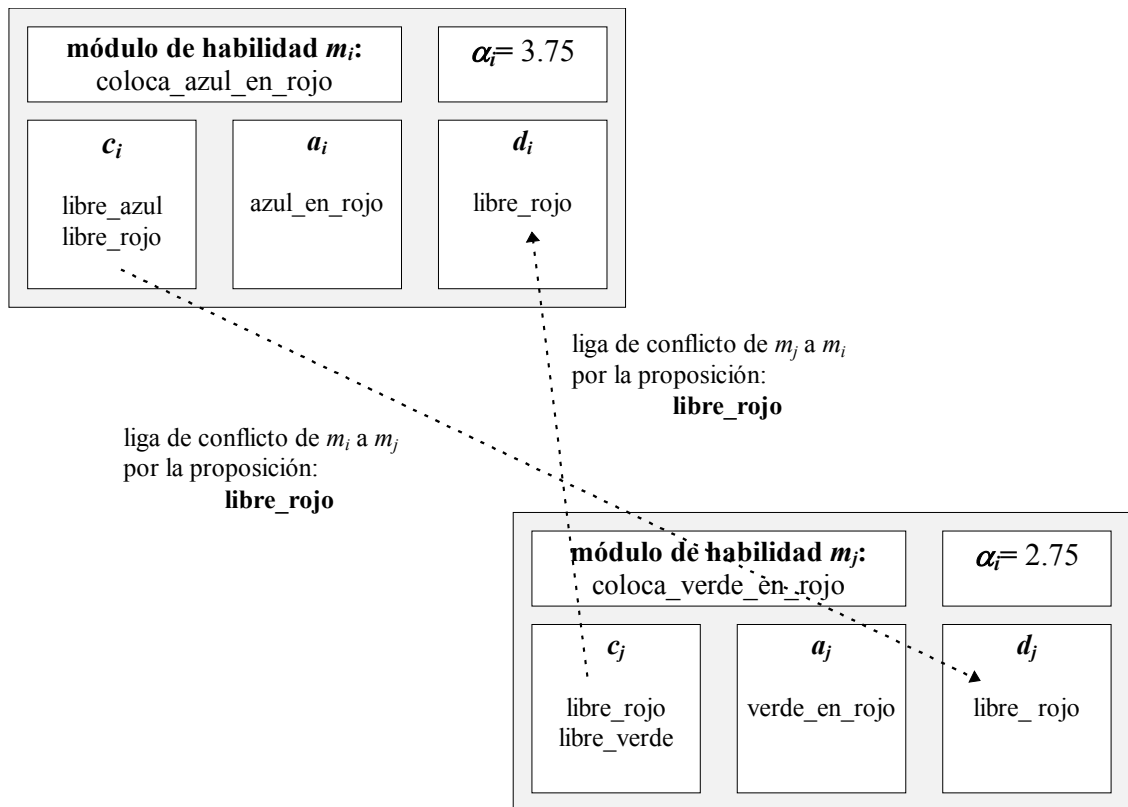


Figura 3.3 Al agregar un nuevo módulo de habilidad para colocar un cubo verde sobre un cubo rojo podemos observar un conflicto mutuo con el módulo coloca_azul_en_rojo: Si m_j se activa, el cubo rojo ya no estará libre y m_i no podrá activarse. De igual forma, si m_i se activa el cubo rojo ya no estaría libre y m_j no podría activarse.

Activación proveniente de las metas globales del agente. Un módulo de habilidad m_i puede llevar a cabo una meta global del agente, si ésta es miembro de su lista a_i . Los módulos de habilidad que pueden realizar alguna de las metas globales del agente se estimulan incrementando su activación. Las metas globales pueden ser de dos tipos:

permanentes, en cuyo caso estimularan al agente permanentemente; y temporales, las cuales una vez realizadas dejan de estimular al agente.

Inhibición proveniente de las metas realizadas. Decimos que un módulo de habilidad m_i puede deshacer una meta global del agente, si ésta es miembro de su lista d_i . Los módulos de habilidad que con sus acciones pueden deshacer una meta previamente realizada se inhiben, por lo que su nivel de activación sufre un decremento.

Los tres factores mencionados se consideran continuamente, de tal forma que la presencia de cambios imprevistos en el medio ambiente y el planteamiento de nuevas metas al agente, se ven reflejados inmediatamente en los niveles de activación de cada módulo de habilidad.

Además de la influencia del medio ambiente y de las metas globales del agente, la red de comportamiento presenta una dinámica entre los módulos de habilidad que obedece a los siguientes factores:

Activación de sucesores. Un módulo de habilidad ejecutable m_i estimula a sus sucesores m_j por cada proposición falsa p que define la liga de sucesión entre ellos, esto es $p \in a_i \cap c_j$. Esta propagación hacia adelante de energía de activación es deseable, pues la activación del módulo m_i contribuirá a que las precondiciones p de sus sucesores sean verdaderas, incrementando la posibilidad de que estos lleguen a ser ejecutables.

Activación de predecesores. Un módulo de habilidad m_i que no es ejecutable estimula a sus predecesores m_j por cada proposición falsa p que define la liga de predecesor entre ellos, esto es $p \in c_i \cap a_j$. Esta propagación hacia atrás de energía de activación es deseable, pues la activación de los sucesores m_j haría que las precondiciones p del módulo m_i cambiasen de falsas a verdaderas.

Inhibición por conflictos. Todo módulo de habilidad m_i , ejecutable o no, inhibe a aquellos módulos $m_j \neq m_i$ con los que tiene conflictos, por cada proposición p que es verdadera y define la liga de conflicto entre ellos, esto es $p \in c_i \cap d_j$. Decimos que $m_j \neq m_i$ si $c_j \neq c_i \vee a_j \neq a_i \vee d_j \neq d_i$. Para estas listas (c, a y d) se cumple que $l_j \neq l_i$ si $(\exists p) [(p \in l_j \wedge p \notin l_i) \vee (p \in l_j \wedge p \notin l_i)]$. Observe que los módulos `coloca_azul_en_rojo` y `coloca_verde_en_rojo` de la figura 3.3 presentarían conflicto con ellos mismos por la proposición `libre_rojo`, es importante señalar que un módulo no puede inhibirse a si mismo por conflicto dado que $m_j \neq m_i$. En el caso de presentarse un conflicto mutuo entre dos módulos de habilidad, el módulo con mayor activación inhibe al de menor activación, lo cual evita que se presente el fenómeno de mutua eliminación entre módulos relevantes. En caso de querer simular el comportamiento irrelevante [Maes 91] que se presenta en algunos animales, la inhibición entre módulos con conflictos mutuos se da en ambos sentidos.

El algoritmo consiste en un ciclo en el que se llevan a cabo los siguientes cálculos para todos los módulos de habilidad que componen al agente:

1. Se calcula el impacto del medio ambiente, las metas y las metas realizadas.
2. Se calcula el impacto de la dinámica de activación/inhibición entre los módulos
3. Se normalizan los niveles de activación para mantener su suma constante
4. El módulo de habilidad que cumple con los siguientes condiciones se activa, su nivel de activación se fija en cero y el umbral regresa a su valor inicial:

- a) es ejecutable
 - b) su nivel de activación sobrepasa cierto umbral
 - c) tiene el nivel de activación más alto entre los módulos que cumplen a y b (en caso de empate se elige uno aleatoriamente)
5. Si no hay un módulo de habilidad que se active, el umbral se decrementa en un 10%

Cuatro parámetros globales pueden ser utilizados para ajustar la dinámica de activación/inhibición entre los módulos de habilidad y, por lo tanto, la selección de acciones por parte del agente:

- θ el umbral que deben superar los módulos de habilidad para activarse. Su valor disminuye un 10% en cada tiempo del ciclo principal en que ningún módulo de habilidad entra en actividad y regresa a su valor original cuando un módulo se activa. Este parámetro está relacionado con π que es el nivel promedio de activación de los módulos que componen un agente.
- σ la cantidad de energía que una proposición que se observa verdadera inyecta a la red
- γ la cantidad de energía que una meta inyecta a la red
- δ la cantidad de energía que las metas realizadas retiran de la red

Algunas de las propiedades globales que resulta interesante observar son: la secuencia de módulos de habilidad que entran en actividad, la optimalidad de esa secuencia (su cálculo es dependiente de cada problema) y la velocidad con que las metas son realizadas (la relación entre el número de tiempos que se ha repetido el ciclo y la cantidad de módulos de habilidad que han sido activados).

3.3 Modelo matemático

3.3.1 Módulo de habilidad

Un módulo de habilidad m_i se define como la tupla: $m_i = (c_i, a_i, d_i, \alpha_i)$ donde:

- c_i Conjunto de proposiciones que necesitan observarse verdaderas para que m_i pueda activarse (precondiciones).
- a_i Conjunto de proposiciones que se observaran verdaderas al activarse m_i .
- d_i Conjunto de proposiciones que dejaran de observarse verdaderas al activarse m_i .
- α_i Nivel de activación del m_i , $\alpha_i \in \mathfrak{R}^+$.

3.3.2 Agente

Un agente A , capaz de exhibir n habilidades ó acciones, se define como: $A = \{m_1 \dots m_n\}$

3.3.3 Proposiciones

Sea $p_i = c_i \cup a_i \cup d_i$ el conjunto de proposiciones utilizados para definir el módulo de habilidad m_i .

Decimos que $P_A = p_1 \cup p_2 \cup \dots \cup p_n$ es el conjunto de proposiciones utilizado para definir al agente A .

Algunas proposiciones $p \in P_A$ están relacionadas con un sensor virtual que determina su valor de verdad con base en la información proveniente de sensores reales.

Dado un agente A , definimos:

$M(p) = \{m_i: m_i \in A \wedge p \in c_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de c_i .

$A(p) = \{m_i: m_i \in A \wedge p \in a_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de a_i .

$D(p) = \{m_i: m_i \in A \wedge p \in d_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de d_i .

3.3.4 Percepción

La función $S(t) = \{p: p \in P_A \wedge p \text{ es verdadera al tiempo } t\}$ regresa el conjunto de proposiciones que se observan como verdaderas en el medio ambiente al tiempo t .

Esta función es dependiente del dominio del agente y descansa en la existencia de sensores virtuales que, como mencionamos anteriormente, determinan el valor de verdad de su proposición asociada con base en la información proveniente de un sensor real.

Sea $S_p(t)$ el sensor virtual asociado a la proposición $p \in P_A$ y $SR_p(t)$ el sensor real que provee la información necesaria para determinar el valor de verdad de p , entonces:

$$S_p(t) = \begin{cases} \{p\} & \text{Si la información de } SR_p(t) \text{ determina que } p \text{ es verdadera} \\ \{\} & \text{En cualquier otro caso} \end{cases}$$

Desde este punto de vista la función $S(t)$ regresa el conjunto formado por la unión de todos los sensores virtuales que posee el agente A .

3.3.5 Metas globales del agente

La función $G(t) = \{p: p \in P_A \wedge p \text{ es una meta del agente al tiempo } t\}$ regresa el conjunto de proposiciones que son metas globales de la sociedad de módulos de habilidad que conforman al agente A . Esta función es dependiente del dominio.

La función $GR(t) = \{S(t) \cap G(t)\}$ regresa el conjunto de metas que el agente ha realizado al tiempo t .

3.3.6 Módulo de habilidad ejecutable.

La función:

$$E(m_i, t) = \begin{cases} 1 & \text{Si } c_i \supset S(t) \\ 0 & \text{en cualquier otro caso} \end{cases}$$

regresa 1 si todas las precondiciones del módulo m_i se observan verdaderas al tiempo t , en cuyo caso decimos que el módulo m_i es ejecutable.

3.3.7 Parámetros.

π es el nivel de activación promedio. Dado un agente A_n :

$$\sum_{i=1..n} \alpha_i(t) = \pi \cdot n, \forall t$$

θ el umbral de activación. Para que un módulo de habilidad m_i pueda activarse es necesario que $\alpha_i > \theta$ y que $E(m_i, t) = 1$

σ es un valor representando la influencia del medio ambiente en el agente ó la cantidad de energía que una proposición que se observa verdadera inyecta en la red de comportamiento.

γ es un valor representando la influencia de las metas en el agente ó la cantidad de energía que una meta inyecta en la red de comportamiento.

δ es un valor representando la influencia de las metas realizadas y conflictos en el agente ó la cantidad de energía que una meta realizada retira de la red de comportamiento.

Los parámetros $\pi, \theta, \sigma, \gamma, \delta \in \mathbb{R}^+$

3.3.8 Dinámica de Activación / Inhibición

Activación por influencia del medio ambiente. La cantidad de energía que recibe un módulo m_i del medio ambiente, por correspondencia parcial con éste es:

$$entrada_por_estado(m_i, t) = \sum_{\forall p \in P} \sigma \frac{1}{|M(p)|} \frac{1}{|c_i|}$$

donde $P = S(t) \cap c_i$ y $|X|$ indica la cardinalidad del conjunto X .

Activación por influencia de las metas. La cantidad de energía que un módulo m_i recibe debido a su capacidad de realizar alguna meta del agente es:

$$entrada_por_metas(m_i, t) = \sum_{\forall p \in P} \gamma \frac{1}{|A(p)|} \frac{1}{|a_i|}$$

donde $P = G(t) \cap a_i$

Inhibición por metas realizadas. La cantidad de energía que es retirada de un módulo m_i cuya acción puede afectar una meta previamente realizada es:

$$salida_por_metas_realizadas(m_i, t) = \sum_{\forall p \in P} \delta \frac{1}{|D(p)|} \frac{1}{|d_i|}$$

donde $P = GR(t) \cap d_i$

Propagación hacia adelante. La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de predecesor es:

$$factor_fwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{\sigma}{\gamma} \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 1 \\ 0 & \text{Si } E(m_i, t) = 0 \end{cases}$$

donde $P = \{p: p \in c_i \cap a_j \wedge p \notin S(t)\}$

De esta forma, la cantidad de energía que un módulo m_i , recibe por propagación hacia adelante es:

$$fwp(m_i, t) = \sum_{\forall m \in A} factor_fwp(m_i, m, t)$$

Propagación hacia atrás. La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de sucesor es:

$$factor_bwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 0 \\ 0 & \text{Si } E(m_i, t) = 1 \end{cases}$$

donde $P = \{p: p \in a_i \cap c_j \wedge p \in S(t)\}$

De esta forma, la cantidad de energía que un módulo m_i , recibe por propagación hacia adelante es:

$$bwp(m_i, t) = \sum_{\forall m \in A} factor_bwp(m_i, m, t)$$

Inhibición por conflicto. La cantidad de energía que es retirada de un módulo m_i por presentar conflictos con otro módulo m_j a través de sus ligas de conflicto es:

$$factor_confp(m_i, m_j, t) = \begin{cases} 0 & \text{Si } \alpha_i(t-1) > \alpha_j(t-1) \wedge (\exists p)(p \in S(t) \cap c_i \cap d_j) \\ \max \left[\sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|D(p)|} \frac{1}{|d_i|}, \alpha_i(t-1) \right] & \text{En cualquier otro caso} \end{cases}$$

donde $P = \{p: p \in d_i \cap c_j \wedge p \in S(t)\}$

De esta forma, la cantidad de energía que es retirada de un módulo m_i , por inhibición debida a conflictos es:

$$confp(m_i, t) = \sum_{\forall m \in A \wedge m \neq m_i} factor_confp(m_i, m, t)$$

Nivel de activación. El nivel de activación del módulo m_i al tiempo t se define como:

$$\alpha(m_i, t) = \begin{cases} 0 & \text{Si } t = 0 \\ normaliza \left[\begin{array}{l} \alpha(m_i, t-1) + \\ entrada_por_estado(m_i, t) + \\ entrada_por_metas(m_i, t) - \\ salida_por_metas_realizadas(m_i, t) + \\ fwp(m_i, t) + bwp(m_i, t) - confp(m_i, t) \end{array} \right] & \text{Si } t > 0 \end{cases}$$

donde *normaliza* es una función que se encarga de que la activación total del agente A_n permanezca constante en π .

3.3.9 Módulo de habilidad activo

El módulo de habilidad m_i que se activa al tiempo t está determinado por:

$$activo(t, m_i) = \begin{cases} 1 & \text{Si } \begin{cases} \alpha(m_i, t) > \theta \wedge \\ E(m_i, t) \wedge \\ \alpha(m_i, t) > \alpha(m_{j \neq i}, t) \end{cases} \\ 0 & \text{En cualquier otro caso} \end{cases}$$

3.4 Propiedades y configuración de parámetros

El algoritmo de selección de acciones presentado exhibe propiedades interesantes que discutiremos en esta sección, entre ellas: capacidad de planificación, comportamiento orientado a metas, comportamiento guiado por el medio ambiente, adaptabilidad, tendencia a favorecer los planes en curso, prevención de conflictos, previsión y velocidad.

De especial importancia es la posibilidad de configurar la red, mediante los parámetros globales, para poner énfasis en aquellas propiedades que deseamos sean exhibidas durante la selección de acción.

3.4.1 Capacidad de planificación

Las redes de comportamiento exhiben una capacidad de planificación al considerar, de cierta forma, el efecto de una secuencia de acciones antes de llevarlas a cabo. Si una secuencia de módulos de habilidad es capaz de transformar la situación actual en la que se encuentra el agente, en aquella donde es posible dar cumplimiento a las metas globales, entonces los módulos de habilidad en esa secuencia resultarán altamente estimulados. Por el contrario, si una secuencia de acciones implica efectos negativos, los módulos de habilidad en esa secuencia serán inhibidos.

La relevancia de las metas se obtiene por la activación por metas globales del agente y la propagación hacia atrás a partir de los módulos de habilidad que pueden realizar las metas globales del agente. La relevancia de la situación actual y el comportamiento oportunista del agente, se obtienen por la activación por el medio ambiente y la propagación hacia adelante entre los módulos de habilidad que tiene correspondencia parcial con el medio ambiente. La prevención de conflictos y la relevancia de metas interactuantes se lleva a cabo por la inhibición proveniente de metas realizadas y la inhibición entre módulos conflictivos. Los máximos locales en el proceso de selección de acción pueden ser evitados si se permite que la dinámica de activación/inhibición se lleve a cabo durante el tiempo adecuado, fijando el umbral de activación θ lo suficientemente alto, de tal forma que la red evolucione hacia mejores patrones de activación. Finalmente, existe un sesgo a favor de la secuencia de módulos de habilidad que está siendo estimulada, ó plan en curso, debido a los niveles de activación acumulados.

Algunas diferencias con los sistemas de planificación construidos previamente en Inteligencia Artificial [Fikes y Nilsson 71], [Sacerdoti 74] son: Una red de comportamiento no construye una representación explícita del plan a ejecutar, pero expresa la “intención” de llevar a cabo ciertas acciones mediante altos niveles de activación en los módulos de habilidad correspondientes. Una red de comportamiento no tiene un módulo central que lleve a cabo un proceso de búsqueda, la dinámica de activación/inhibición entre los

operadores va resultando en la ejecución de los operadores apropiados. El resultado es que algunos de los problemas asociados a los árboles de búsqueda, como información duplicada en varias partes del árbol, crecimiento exponencial en relación con las dimensiones del problema, representación demasiado estricta, no están presentes en las redes de comportamiento. Como contra parte, las decisiones que produce una red de comportamiento son menos “racionales” que las producidas por los planificadores deliberativos tradicionales en Inteligencia Artificial.

3.4.2 Comportamiento orientado a metas.

El algoritmo presentado contribuye a que las metas globales del agente sean realizadas. Si g es una meta global del agente, entonces una cantidad γ de energía de activación es inyectada a aquellos módulos de habilidad que pueden realizar esa meta. A su vez, estos módulos llevarán a cabo una propagación de energía hacia aquellos predecesores que pueden hacer que sus precondiciones falsas sean verdaderas. Esta propagación hacia atrás contribuye a que los módulos de habilidad involucrados en el cumplimiento de g sean favorecidos con altos niveles de activación.

La dinámica de activación/inhibición favorece a aquellos módulos que contribuyen con diferentes metas globales del agente. De igual forma, favorece el cumplimiento de metas que tiene asociadas las cadenas más cortas entre los módulos que tienen correspondencia parcial con el medio ambiente y aquellos que realizan las metas. Los módulos que tienen menos competencia también resultan favorecidos, esto es, si un sólo módulo puede satisfacer la meta $g1$ y dos módulos pueden satisfacer la meta $g2$, es muy probable que $g1$ se lleve a cabo primero que $g2$. Esto se debe a que los módulos asociados a $g2$ tienen que compartir la cantidad de energía de activación γ . Observe que las precondiciones falsa de un módulo de habilidad pueden ser consideradas como submetas del agente, y por lo tanto presentan estas mismas propiedades.

La relevancia de las metas en la selección de acción puede ajustarse variando la tasa entre γ y σ . Si $\sigma = 0$ la selección de acciones por parte del agente está completamente guiada por las metas, lo cual produce un mecanismo de encadenamiento hacia atrás tradicional. En ese caso, el agente tomará menos oportunidad de las situaciones que se le presenten, será menos reactivo y la tendencia a favorecer el plan en curso será menor, lo cual repercute en la velocidad con que alcanza sus metas el agente. Idealmente deseamos que el agente tenga una fuerte influencia de las metas y tome ventaja de las oportunidades que se le presentan. Esto se logra eligiendo $\gamma > \sigma > 0$. La tasa óptima es dependiente de cada problema.

3.4.3 Comportamiento guiado por el medio ambiente

El algoritmo incrementa el nivel de activación de los módulos que son relevantes a las condiciones observadas en el medio ambiente. Esto es resultado del estímulo que experimentan aquellos módulos que tiene correspondencia parcial con la percepción del medio por parte del agente y la posterior propagación de energía hacia sus sucesores. La relevancia del medio ambiente en el proceso de selección de acción puede ajustarse mediante el parámetro σ . Al incrementar el valor de σ la influencia del medio ambiente es mayor. La propagación de energía de activación hacia adelante hace que los módulos con

mayor probabilidad de activarse, dadas las condiciones actuales del medio ambiente, sean estimulados incrementando su nivel de activación.

La capacidad de aprovechar oportunidades ha sido reconocida como central en el diseño de agentes autónomos [Maes 95].

3.4.4 Adaptabilidad

El proceso de selección de acciones presentado en este capítulo es completamente abierto en el sentido de que el medio ambiente y las metas pueden cambiar en tiempo de ejecución, repercutiendo rápidamente, en un cambio en los patrones de activación/inhibición para ajustarse a la nueva situación. Debido a un proceso continuo de percepción del medio ambiente y re-evaluación de qué metas han sido realizadas, el agente puede adaptarse fácilmente a situaciones imprevistas. El nivel de activación de los módulos juega un papel muy importante en cuanto a la adaptabilidad del agente. En cierta forma, el nivel de activación, representa la relevancia de un módulo a través del tiempo, lo cual permite que las situaciones imprevistas no impliquen una replanificación a partir de cero, y un ajuste de los niveles de activación acorde a la nueva situación sea suficiente. Es muy probable que esta adecuación lleve a la ejecución de otra cadena de módulos de habilidad, o plan alternativo.

Cuando γ y σ son relativamente pequeños en relación con π el sistema es menos adaptativo y presenta una marcada tendencia a favorecer el plan en curso. Esto se debe a que la dinámica de activación/inhibición entre los módulos de habilidad tiene un mayor impacto que la influencia del medio ambiente y de las metas. La elección de los valores adecuados para los parámetros debe evitar que el agente salte entre planes que contribuyen a diferentes metas sin lograr llegar a realizarlas.

La naturaleza distribuida de este algoritmo ofrece otro tipo de adaptabilidad debido a la tolerancia ante fallas. Si un módulo de habilidad presenta fallas, el agente es capaz de utilizar el resto de los módulos para tratar de realizar sus metas.

3.4.5 Tendencia a favorecer los planes en curso

El algoritmo exhibe una tendencia implícita a favorecer los planes en curso, con excepción de situaciones que requieren urgentemente que el agente realice una meta diferente a la del plan en curso. La presencia de este sesgo, puede explicarse en el hecho de que el nivel de activación de los módulos no es reinicializado cada vez que un módulo entra en activación. Como una consecuencia, la historia de su activación en el pasado juega un papel importante en la selección de acción, en particular cuando el impacto del medio ambiente y de las metas es relativamente pequeño en comparación con el nivel promedio de activación π . Pero aún cuando no es este el caso, el algoritmo exhibe dos tipos de sesgo. El sesgo horizontal se refiere a que el algoritmo presenta una tendencia a trabajar con una de sus metas a la vez hasta que la realiza, y sólo entonces considera otra meta. En general, hay una tendencia en atender primero las metas que tienen asociadas secuencias de módulos cortas. Los módulos de habilidad que contribuyen a que la meta en turno sea realizada tienden a ser estimulados. El sesgo vertical se refiere a que el algoritmo favorece la ejecución de módulos que satisfacen submetas del agente, precondiciones falsas, que contribuyen a la misma meta global.

3.4.6 Prevención de conflictos

Activar módulos de habilidad en un orden que no es el adecuado, puede incrementar dramáticamente la cantidad de acciones necesarias para llegar a la realización de las metas, e inclusive puede llevar a situaciones donde ya no es posible cumplir algunas de ellas. Por lo tanto, cualquier mecanismo de selección de acción debe ser capaz de mediar entre acciones conflictivas. Las reglas de inhibición de este algoritmo realizan esta mediación. Los módulos de habilidad que pueden deshacer una meta ya realizada, son inhibidos y su nivel de activación se reduce en un factor de δ . Si este parámetro es lo suficientemente alto, la selección de acciones protegerá metas globales y submetas que hallan sido realizadas. Es necesario establecer un equilibrio entre una selección de acciones que no toma en cuenta las metas que se han realizado y una que es incapaz de deshacer metas realizadas y por lo tanto puede llegar a generar situaciones donde ningún módulo de habilidad se activará, pues la única posibilidad es deshacer una meta ya realizada.

3.4.7 Previsión y velocidad

La selección de acción puede hacerse más ó menos previsor, variando el valor del parámetro θ . Si el umbral de activación θ es lo suficientemente alto como para permitir que la dinámica de activación/inhibición se de por más tiempo, la selección de acciones es más elaborada, pues se evitan máximos locales. Por otra parte, si el umbral es muy alto, la selección de acciones tomará demasiado tiempo, lo cual no es deseable, sobre todo en medios ambientes que cambian rápidamente. En contra parte, si el umbral de activación es lo suficientemente bajo, la velocidad en que el sistema realiza sus metas se incrementa, pero las acciones elegidas son menos orientadas a las metas, menos oportunistas y están más expuestas a los conflictos.

3.5 Consideraciones

Un factor importante a considerar en nuestra presentación del algoritmo es el relacionado con la complejidad del cómputo que éste lleva a cabo. Podemos observar similitudes entre la selección de acciones en una red de comportamiento y los procesos de búsqueda empleados tradicionalmente en IA, por lo cual podríamos suponer que el rendimiento del algoritmo se ve reducido en la medida que el número de módulos de habilidad de un agente aumentan. A continuación expondremos algunos argumentos [Maes 90] que nos hacen creer que las redes de comportamiento no presentan este problema.

El máximo número de pasos que el sistema necesita para seleccionar una acción está acotado por el ancho de la red de comportamiento multiplicado por alguna función lineal del umbral de activación. El ancho de la red es la distancia, en términos de módulos de habilidad, entre los módulos ejecutables y los módulos que realizan metas. Además, el algoritmo evalúa diferentes cursos de acción en paralelo, de tal forma que no es necesario reiniciar el proceso cuando un curso de acción no llega a realizarse, simplemente se da un cambio a otro de los cursos de acción que están siendo evaluados. Lo anterior, aunado al hecho de que el sistema no construye un árbol de búsqueda, ni mantiene representaciones de planes parciales, nos hace pensar que el cómputo requerido por las redes de comportamiento es menor al empleado en técnicas de búsqueda tradicionales.

La experiencia en el diseño de agente autónomos parece indicar que una red de comportamiento crece en profundidad y no en amplitud. Típicamente un agente autónomo debe enfrentar muchas metas que requiere cursos de acción de poca amplitud, en lugar de enfrentar metas que requieren de un número considerable de acciones para ser realizadas y por lo tanto, requieren más “planificación”. Esto nos hace pensar que el impacto del número de módulos de habilidad que componen un agente en su eficiencia no es tan grave. Por otra parte, aún cuando la red de comportamiento tuviera cursos de acción muy amplios, es posible que el algoritmo seleccione cursos de acción alternativos ya que no espera convergencia en los niveles de activación y el umbral de activación disminuye con el paso del tiempo. Debemos señalar que en este último caso, el comportamiento del agente no tendería al óptimo.

El hecho de que las mismas reglas de activación/inhibición se apliquen por igual a todos los módulos de habilidad, que además, están conectados por ligas fijas representadas localmente, abre interesantes oportunidades de paralelizar el algoritmo, lo cual implicaría una mejora considerable en la velocidad de éste.

Un segundo factor a considerar en esta sección es el relacionado con la ausencia de variables. El algoritmo de redes de comportamiento no incorpora variables. En su lugar emplea una *representación directa* como la utilizada en Pengi [Agre y Chapman 90] presentada en el capítulo anterior. Una consecuencia de la ausencia de variables en el algoritmo es que todos los módulos de habilidad ó operadores deben estar instanciados. Así mismo, no es posible plantear metas del *estilo muevete-a(x,y)*. Aunque esto pudiera parecer una limitación sería, en el capítulo anterior presentamos como el robot Toto [Mataric 89] hace uso de mapas de navegación utilizando *representación directa* que posibilitan metas de tipo *muevete-a-la-puerta*. Es posible incorporar variables en el algoritmo creando copias de un módulo de habilidad cada vez que este es instanciado con valores de variables distintos, sin embargo esta opción requeriría una cantidad mucho mayor de cómputo. La tendencia es utilizar *representación directa* y replantearnos la forma en que un agente puede ser “instruido” sobre las metas a realizar.

Debemos mencionar que el algoritmo presenta algunas limitaciones. Esporádicamente, la selección de acción de un agente puede caer en ciclos. Esto se debe a que el sistema no mantiene una historia de las acciones que ha realizado en el pasado. El hecho de que los ciclos se presenten en contadas ocasiones se debe a la interacción del agente con su medio ambiente. En un medio ambiente *dinámico* es muy probable que las condiciones del medio cambien constantemente, lo cual tiene un impacto en la activación propagada a través de la red de comportamiento y en consecuencia la selección de acción podría salir del ciclo en que se encuentra. Una posible solución es incorporar un mecanismo de *habitación* con el cual cada módulo de habilidad que se activa es menos propenso a activarse en el futuro, esto pueden lograrse usando umbrales de activación locales que varían en el tiempo.

Un segundo problema está relacionado con la selección de parámetros. No es claro cómo podemos seleccionar un conjunto de parámetros dado una aplicación específica. Esta selección depende de las propiedades que deseamos exhiba la red de comportamiento, pero además depende de factores como el tamaño y la estructura de la red de comportamiento. Este problema y sus consecuencias en el estudio y enseñanza de las redes de

comportamiento nos sugirieron la necesidad de implementar una herramienta como ABC descrita en capítulo 5.

Capítulo 3

Redes de Comportamiento

3.1 ¿Cómo hacer lo correcto?

Hemos elegido las redes de comportamiento para implementar un mecanismo de selección de acción y un ambiente de simulación que nos permita experimentar con diversos agentes autónomos y la forma en que estos realizan sus metas eligiendo correctamente sus acciones.

Las propiedades exhibidas por las redes de comportamiento fueron un factor decisivo en esta elección. Este mecanismo exhibe un comportamiento guiado por metas y relevante a la situación del agente, es adaptativo, tolerante ante fallas, persistente en el plan de acción en curso, reactivo y rápido [Maes 89,90b,90c,91] [Franklin95]. Además, las redes de comportamiento son un mecanismo de selección de acción descentralizado y dinámicamente reconfigurable, lo que las hace viables para implementar agentes con actividad situada [Hendriks-Jansen 96].

En este capítulo presentaremos un algoritmo de activación / inhibición en una red de comportamiento, así como el modelo matemático de este mecanismo de selección de acción. Posteriormente, estudiaremos la forma en que una red de comportamiento puede ser configurada a través de sus parámetros globales y terminaremos presentando algunas limitaciones presentes en este mecanismo.

Es importante que durante la lectura de este capítulo tengamos presente que este mecanismo de selección de acción busca verificar la siguiente hipótesis: la selección de acción puede ser modelada como una propiedad emergente de una dinámica de activación / inhibición entre las diferentes acciones que el agente puede realizar, evitando la presencia de módulos burocráticos que funcionen como formas globales de control determinando que acción resultará activada o inhibida.

3.2 Descripción del algoritmo

El algoritmo para elegir la acción correcta basado en redes de comportamiento define un agente como un conjunto de habilidades o destrezas representados como módulos de habilidad. Un módulo de habilidad m_i puede ser descrito por la tupla $(c_i, a_i, d_i, \alpha_i)$. La lista c_i

contiene las precondiciones que deben ser observadas para que el módulo de habilidad m_i se active. Las listas a_i y d_i representan los efectos esperados al activarse el módulo de habilidad m_i : La lista a_i contiene las condiciones que se espera sean verdaderas después de activarse el módulo y la lista d_i las que se espera dejen de serlo. El valor α_i representa el nivel de activación que ha acumulado el módulo. La figura 3.1 representa un módulo de habilidad. Como puede observar, los módulos de habilidad se asemejan a los operadores empleados en los sistemas de planificación clásicos [Fikes y Nilsson 71].

Cuando todas las precondiciones de un módulo de habilidad se observan verdaderas, decimos que éste es *ejectuable*. Ser ejecutable es una condición necesaria para que un módulo de habilidad pueda activarse.

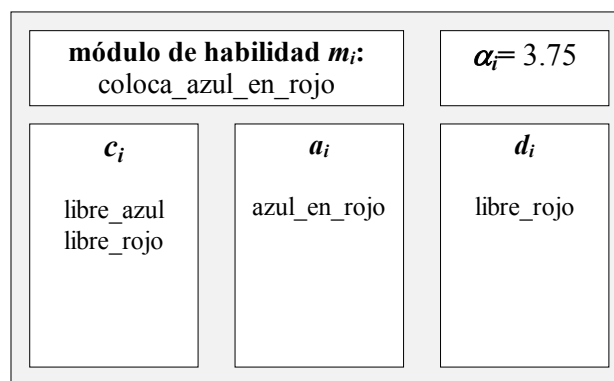


Figura 3.1 Imagine un agente que actúa en un medio ambiente donde puede encontrar cubos de diferentes colores. La idea es que este agente sea capaz de construir torres con esos cubos. Una destreza deseable en este agente es que sea capaz de colocar un cubo sobre otro. En este caso el módulo de habilidad m_i se encargará de colocar un cubo azul sobre uno rojo. Para ello es necesario que el cubo azul y el rojo estén libres. Estas precondiciones se encuentran en la lista c_i . Si el módulo de habilidad m_i se activa, el resultado esperado de sus acciones es que el cubo azul este sobre el rojo (a_i) y que el cubo rojo deje de percibirse como libre (d_i). El nivel de activación acumulado en este módulo es de 3.75 (α_i).

Los módulos de habilidad que conforman un agente se organizan en una red, relacionados por medio de tres tipos de ligas: *sucesor*, *predecesor* y *conflicto*. La composición de cada módulo de habilidad m_i en términos de c_i , a_i , y d_i definen esta red de la siguiente forma:

Hay una *liga de sucesor* (figura 3.2) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in a_i \cap c_j$. Decimos que el módulo de habilidad m_i tiene como sucesor al módulo m_j debido a que las precondiciones p de m_j serían verdaderas de activarse m_i , por lo que existe la posibilidad de que m_j se active después de m_i . Observe que es posible que existan más de una liga del mismo tipo entre dos módulos de habilidad.

Hay una *liga de predecesor* (figura 3.3) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in c_i \cap a_j$. Decimos que el módulo de habilidad m_i tiene como predecesor al módulo m_j , debido a que las precondiciones p de m_i serían verdaderas

de activarse m_j , por lo que existe la probabilidad de que m_j se active antes de m_i . Observe que por cada liga predecesor, existe una de sucesor en sentido contrario.

Hay una *liga de conflicto* (figura 3.4) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in c_i \cap d_j$. Decimos que el módulo m_i tiene como módulo conflictivo al módulo m_j debido a que las precondiciones p de m_i dejarían de observarse verdaderas de activarse m_j , por lo que m_i no podría activarse. Observe que pueden existir conflictos mutuos entre dos módulos.

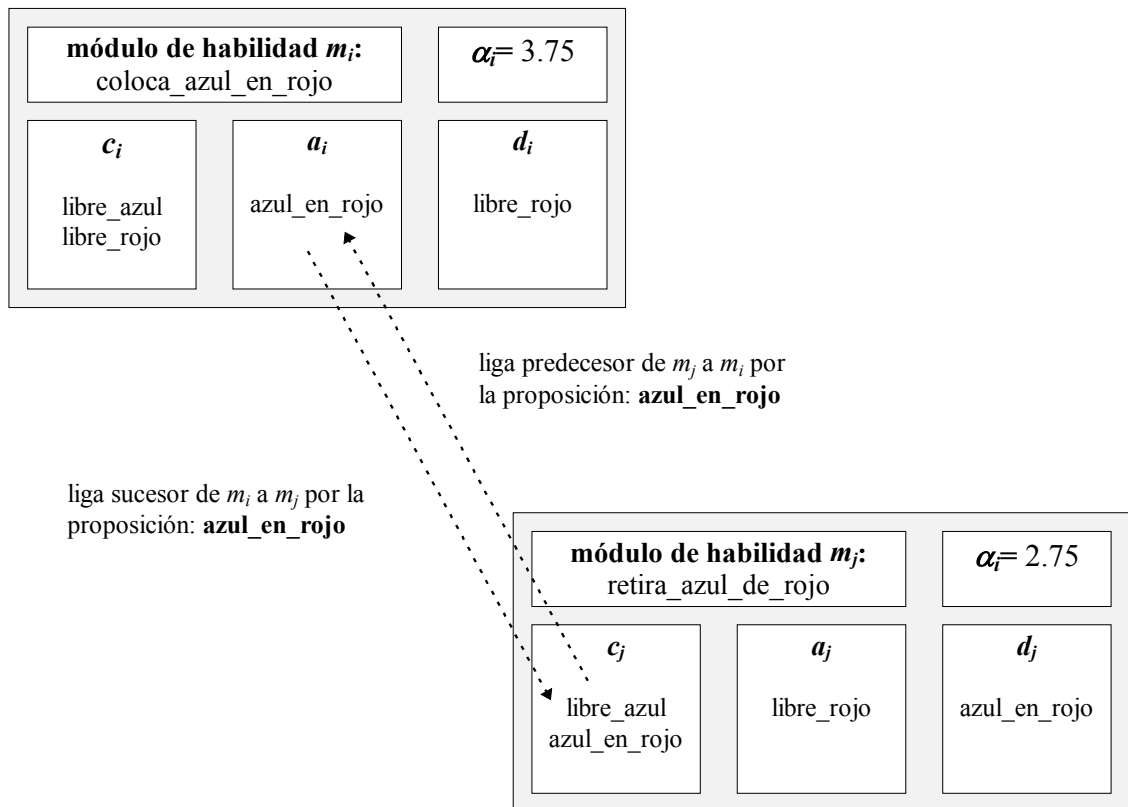


Figura 3.2 La existencia de una liga de sucesor implica la existencia de una liga de predecesor en sentido contrario. En este caso, hay una liga de predecesor del módulo m_j al módulo m_i por la proposición azul_en_rojo. Observe que también la proposición libre_rojo define ligas entre estos dos módulos. Por claridad, únicamente se muestran las ligas originadas por azul_en_rojo.

La idea detrás de estas ligas es que permitan una dinámica de activación/inhibición entre los módulos de habilidad que componen un agente. Esta dinámica tiene efecto sobre el nivel de activación de los módulos, de tal forma que después de un tiempo, aquellos módulos que representan la acciones adecuadas a la situación del agente y las metas que debe llevar a cabo acumulen una mayor activación. Si el nivel de activación de un módulo

ejecutable ha rebasado cierto umbral, entonces se activará desplegando sus habilidades. En esta dinámica de activación/inhibición intervienen los siguientes factores:

Activación proveniente del medio ambiente. Decimos que un módulo de habilidad m_i está en correspondencia parcial con el medio ambiente, si al menos una de sus precondiciones c_i se observa verdadera. Los módulos de habilidad que están en correspondencia parcial con el medio ambiente se estimulan, incrementando su nivel de activación, con la intención de que los módulos que son relevantes a una situación en el medio ambiente sean los que acumulen un mayor nivel de activación. La situación actual del medio ambiente se obtiene utilizando un conjunto de sensores virtuales, los cuales utilizan los datos generados por sensores reales para determinar cuando una proposición es verdadera. La situación actual puede considerarse como la unión de las observaciones de todos los sensores virtuales.

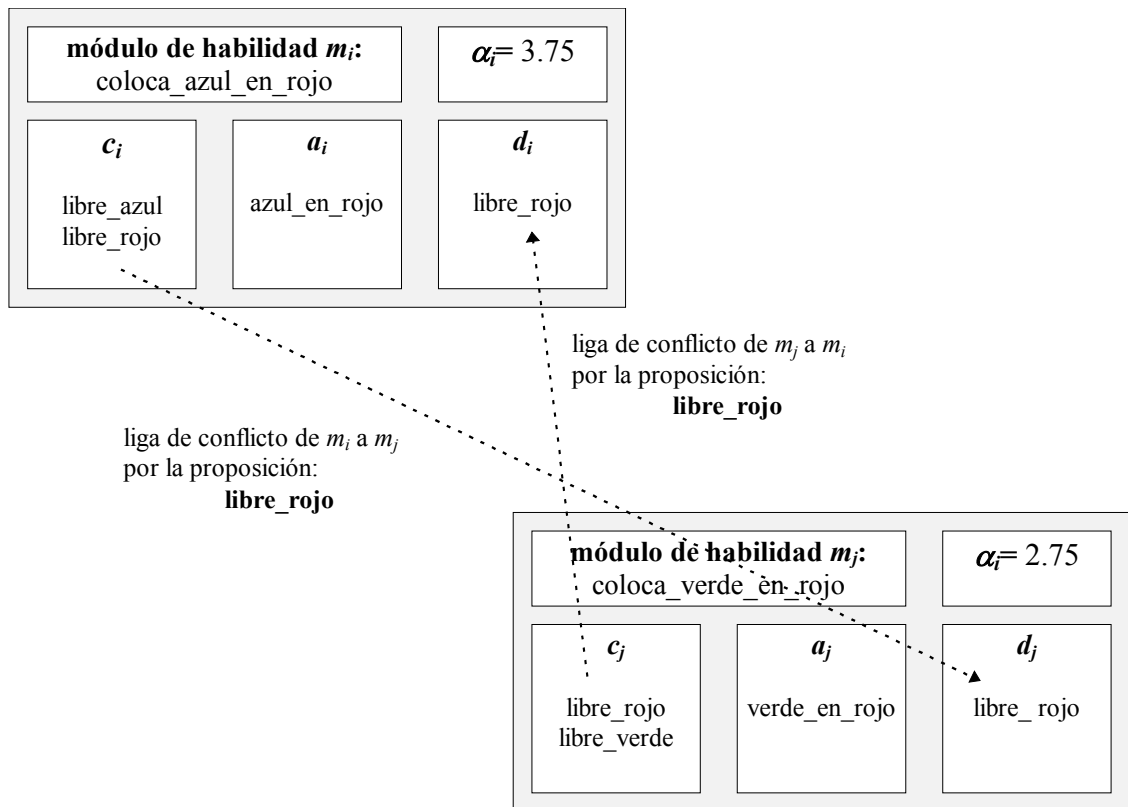


Figura 3.3 Al agregar un nuevo módulo de habilidad para colocar un cubo verde sobre un cubo rojo podemos observar un conflicto mutuo con el módulo coloca_azul_en_rojo: Si m_j se activa, el cubo rojo ya no estará libre y m_i no podrá activarse. De igual forma, si m_i se activa el cubo rojo ya no estaría libre y m_j no podría activarse.

Activación proveniente de las metas globales del agente. Un módulo de habilidad m_i puede llevar a cabo una meta global del agente, si ésta es miembro de su lista a_i . Los módulos de habilidad que pueden realizar alguna de las metas globales del agente se estimulan incrementando su activación. Las metas globales pueden ser de dos tipos:

permanentes, en cuyo caso estimularan al agente permanentemente; y temporales, las cuales una vez realizadas dejan de estimular al agente.

Inhibición proveniente de las metas realizadas. Decimos que un módulo de habilidad m_i puede deshacer una meta global del agente, si ésta es miembro de su lista d_i . Los módulos de habilidad que con sus acciones pueden deshacer una meta previamente realizada se inhiben, por lo que su nivel de activación sufre un decremento.

Los tres factores mencionados se consideran continuamente, de tal forma que la presencia de cambios imprevistos en el medio ambiente y el planteamiento de nuevas metas al agente, se ven reflejados inmediatamente en los niveles de activación de cada módulo de habilidad.

Además de la influencia del medio ambiente y de las metas globales del agente, la red de comportamiento presenta una dinámica entre los módulos de habilidad que obedece a los siguientes factores:

Activación de sucesores. Un módulo de habilidad ejecutable m_i estimula a sus sucesores m_j por cada proposición falsa p que define la liga de sucesión entre ellos, esto es $p \in a_i \cap c_j$. Esta propagación hacia adelante de energía de activación es deseable, pues la activación del módulo m_i contribuirá a que las precondiciones p de sus sucesores sean verdaderas, incrementando la posibilidad de que estos lleguen a ser ejecutables.

Activación de predecesores. Un módulo de habilidad m_i que no es ejecutable estimula a sus predecesores m_j por cada proposición falsa p que define la liga de predecesor entre ellos, esto es $p \in c_i \cap a_j$. Esta propagación hacia atrás de energía de activación es deseable, pues la activación de los sucesores m_j haría que las precondiciones p del módulo m_i cambiasen de falsas a verdaderas.

Inhibición por conflictos. Todo módulo de habilidad m_i , ejecutable o no, inhibe a aquellos módulos $m_j \neq m_i$ con los que tiene conflictos, por cada proposición p que es verdadera y define la liga de conflicto entre ellos, esto es $p \in c_i \cap d_j$. Decimos que $m_j \neq m_i$ si $c_j \neq c_i \vee a_j \neq a_i \vee d_j \neq d_i$. Para estas listas (c, a y d) se cumple que $l_j \neq l_i$ si $(\exists p) [(p \in l_j \wedge p \notin l_i) \vee (p \in l_j \wedge p \notin l_i)]$. Observe que los módulos `coloca_azul_en_rojo` y `coloca_verde_en_rojo` de la figura 3.3 presentarían conflicto con ellos mismos por la proposición `libre_rojo`, es importante señalar que un módulo no puede inhibirse a si mismo por conflicto dado que $m_j \neq m_i$. En el caso de presentarse un conflicto mutuo entre dos módulos de habilidad, el módulo con mayor activación inhibe al de menor activación, lo cual evita que se presente el fenómeno de mutua eliminación entre módulos relevantes. En caso de querer simular el comportamiento irrelevante [Maes 91] que se presenta en algunos animales, la inhibición entre módulos con conflictos mutuos se da en ambos sentidos.

El algoritmo consiste en un ciclo en el que se llevan a cabo los siguientes cálculos para todos los módulos de habilidad que componen al agente:

1. Se calcula el impacto del medio ambiente, las metas y las metas realizadas.
2. Se calcula el impacto de la dinámica de activación/inhibición entre los módulos
3. Se normalizan los niveles de activación para mantener su suma constante
4. El módulo de habilidad que cumple con los siguientes condiciones se activa, su nivel de activación se fija en cero y el umbral regresa a su valor inicial:

- a) es ejecutable
 - b) su nivel de activación sobrepasa cierto umbral
 - c) tiene el nivel de activación más alto entre los módulos que cumplen a y b (en caso de empate se elige uno aleatoriamente)
5. Si no hay un módulo de habilidad que se active, el umbral se decrementa en un 10%

Cuatro parámetros globales pueden ser utilizados para ajustar la dinámica de activación/inhibición entre los módulos de habilidad y, por lo tanto, la selección de acciones por parte del agente:

- θ el umbral que deben superar los módulos de habilidad para activarse. Su valor disminuye un 10% en cada tiempo del ciclo principal en que ningún módulo de habilidad entra en actividad y regresa a su valor original cuando un módulo se activa. Este parámetro está relacionado con π que es el nivel promedio de activación de los módulos que componen un agente.
- σ la cantidad de energía que una proposición que se observa verdadera inyecta a la red
- γ la cantidad de energía que una meta inyecta a la red
- δ la cantidad de energía que las metas realizadas retiran de la red

Algunas de las propiedades globales que resulta interesante observar son: la secuencia de módulos de habilidad que entran en actividad, la optimalidad de esa secuencia (su cálculo es dependiente de cada problema) y la velocidad con que las metas son realizadas (la relación entre el número de tiempos que se ha repetido el ciclo y la cantidad de módulos de habilidad que han sido activados).

3.3 Modelo matemático

3.3.1 Módulo de habilidad

Un módulo de habilidad m_i se define como la tupla: $m_i = (c_i, a_i, d_i, \alpha_i)$ donde:

- c_i Conjunto de proposiciones que necesitan observarse verdaderas para que m_i pueda activarse (precondiciones).
- a_i Conjunto de proposiciones que se observaran verdaderas al activarse m_i .
- d_i Conjunto de proposiciones que dejaran de observarse verdaderas al activarse m_i .
- α_i Nivel de activación del m_i , $\alpha_i \in \mathfrak{R}^+$.

3.3.2 Agente

Un agente A , capaz de exhibir n habilidades ó acciones, se define como: $A = \{m_1 \dots m_n\}$

3.3.3 Proposiciones

Sea $p_i = c_i \cup a_i \cup d_i$ el conjunto de proposiciones utilizados para definir el módulo de habilidad m_i .

Decimos que $P_A = p_1 \cup p_2 \cup \dots \cup p_n$ es el conjunto de proposiciones utilizado para definir al agente A .

Algunas proposiciones $p \in P_A$ están relacionadas con un sensor virtual que determina su valor de verdad con base en la información proveniente de sensores reales.

Dado un agente A , definimos:

$M(p) = \{m_i: m_i \in A \wedge p \in c_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de c_i .

$A(p) = \{m_i: m_i \in A \wedge p \in a_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de a_i .

$D(p) = \{m_i: m_i \in A \wedge p \in d_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de d_i .

3.3.4 Percepción

La función $S(t) = \{p: p \in P_A \wedge p \text{ es verdadera al tiempo } t\}$ regresa el conjunto de proposiciones que se observan como verdaderas en el medio ambiente al tiempo t .

Esta función es dependiente del dominio del agente y descansa en la existencia de sensores virtuales que, como mencionamos anteriormente, determinan el valor de verdad de su proposición asociada con base en la información proveniente de un sensor real.

Sea $S_p(t)$ el sensor virtual asociado a la proposición $p \in P_A$ y $SR_p(t)$ el sensor real que provee la información necesaria para determinar el valor de verdad de p , entonces:

$$S_p(t) = \begin{cases} \{p\} & \text{Si la información de } SR_p(t) \text{ determina que } p \text{ es verdadera} \\ \{\} & \text{En cualquier otro caso} \end{cases}$$

Desde este punto de vista la función $S(t)$ regresa el conjunto formado por la unión de todos los sensores virtuales que posee el agente A .

3.3.5 Metas globales del agente

La función $G(t) = \{p: p \in P_A \wedge p \text{ es una meta del agente al tiempo } t\}$ regresa el conjunto de proposiciones que son metas globales de la sociedad de módulos de habilidad que conforman al agente A . Esta función es dependiente del dominio.

La función $GR(t) = \{S(t) \cap G(t)\}$ regresa el conjunto de metas que el agente ha realizado al tiempo t .

3.3.6 Módulo de habilidad ejecutable.

La función:

$$E(m_i, t) = \begin{cases} 1 & \text{Si } c_i \supset S(t) \\ 0 & \text{en cualquier otro caso} \end{cases}$$

regresa 1 si todas las precondiciones del módulo m_i se observan verdaderas al tiempo t , en cuyo caso decimos que el módulo m_i es ejecutable.

3.3.7 Parámetros.

π es el nivel de activación promedio. Dado un agente A_n :

$$\sum_{i=1..n} \alpha_i(t) = \pi \cdot n, \forall t$$

θ el umbral de activación. Para que un módulo de habilidad m_i pueda activarse es necesario que $\alpha_i > \theta$ y que $E(m_i, t) = 1$

σ es un valor representando la influencia del medio ambiente en el agente ó la cantidad de energía que una proposición que se observa verdadera inyecta en la red de comportamiento.

γ es un valor representando la influencia de las metas en el agente ó la cantidad de energía que una meta inyecta en la red de comportamiento.

δ es un valor representando la influencia de las metas realizadas y conflictos en el agente ó la cantidad de energía que una meta realizada retira de la red de comportamiento.

Los parámetros $\pi, \theta, \sigma, \gamma, \delta \in \mathfrak{R}^+$

3.3.8 Dinámica de Activación / Inhibición

Activación por influencia del medio ambiente. La cantidad de energía que recibe un módulo m_i del medio ambiente, por correspondencia parcial con éste es:

$$entrada_por_estado(m_i, t) = \sum_{\forall p \in P} \sigma \frac{1}{|M(p)|} \frac{1}{|c_i|}$$

donde $P = S(t) \cap c_i$ y $|X|$ indica la cardinalidad del conjunto X .

Activación por influencia de las metas. La cantidad de energía que un módulo m_i recibe debido a su capacidad de realizar alguna meta del agente es:

$$entrada_por_metas(m_i, t) = \sum_{\forall p \in P} \gamma \frac{1}{|A(p)|} \frac{1}{|a_i|}$$

donde $P = G(t) \cap a_i$

Inhibición por metas realizadas. La cantidad de energía que es retirada de un módulo m_i cuya acción puede afectar una meta previamente realizada es:

$$salida_por_metas_realizadas(m_i, t) = \sum_{\forall p \in P} \delta \frac{1}{|D(p)|} \frac{1}{|d_i|}$$

donde $P = GR(t) \cap d_i$

Propagación hacia adelante. La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de predecesor es:

$$factor_fwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{\sigma}{\gamma} \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 1 \\ 0 & \text{Si } E(m_i, t) = 0 \end{cases}$$

donde $P = \{p: p \in c_i \cap a_j \wedge p \notin S(t)\}$

De esta forma, la cantidad de energía que un módulo m_i , recibe por propagación hacia adelante es:

$$fwp(m_i, t) = \sum_{\forall m \in A} factor_fwp(m_i, m, t)$$

Propagación hacia atrás. La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de sucesor es:

$$factor_bwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 0 \\ 0 & \text{Si } E(m_i, t) = 1 \end{cases}$$

donde $P = \{p: p \in a_i \cap c_j \wedge p \in S(t)\}$

De esta forma, la cantidad de energía que un módulo m_i , recibe por propagación hacia adelante es:

$$bwp(m_i, t) = \sum_{\forall m \in A} factor_bwp(m_i, m, t)$$

Inhibición por conflicto. La cantidad de energía que es retirada de un módulo m_i por presentar conflictos con otro módulo m_j a través de sus ligas de conflicto es:

$$factor_confp(m_i, m_j, t) = \begin{cases} 0 & \text{Si } \alpha_i(t-1) > \alpha_j(t-1) \wedge (\exists p)(p \in S(t) \cap c_i \cap d_j) \\ \max \left[\sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|D(p)|} \frac{1}{|d_i|}, \alpha_i(t-1) \right] & \text{En cualquier otro caso} \end{cases}$$

donde $P = \{p: p \in d_i \cap c_j \wedge p \in S(t)\}$

De esta forma, la cantidad de energía que es retirada de un módulo m_i , por inhibición debida a conflictos es:

$$confp(m_i, t) = \sum_{\forall m \in A \wedge m \neq m_i} factor_confp(m_i, m, t)$$

Nivel de activación. El nivel de activación del módulo m_i al tiempo t se define como:

$$\alpha(m_i, t) = \begin{cases} 0 & \text{Si } t = 0 \\ normaliza \left[\begin{array}{l} \alpha(m_i, t-1) + \\ entrada_por_estado(m_i, t) + \\ entrada_por_metas(m_i, t) - \\ salida_por_metas_realizadas(m_i, t) + \\ fwp(m_i, t) + bwp(m_i, t) - confp(m_i, t) \end{array} \right] & \text{Si } t > 0 \end{cases}$$

donde *normaliza* es una función que se encarga de que la activación total del agente A_n permanezca constante en π .

3.3.9 Módulo de habilidad activo

El módulo de habilidad m_i que se activa al tiempo t está determinado por:

$$activo(t, m_i) = \begin{cases} 1 & \text{Si } \begin{cases} \alpha(m_i, t) > \theta \wedge \\ E(m_i, t) \wedge \\ \alpha(m_i, t) > \alpha(m_{j \neq i}, t) \end{cases} \\ 0 & \text{En cualquier otro caso} \end{cases}$$

3.4 Propiedades y configuración de parámetros

El algoritmo de selección de acciones presentado exhibe propiedades interesantes que discutiremos en esta sección, entre ellas: capacidad de planificación, comportamiento orientado a metas, comportamiento guiado por el medio ambiente, adaptabilidad, tendencia a favorecer los planes en curso, prevención de conflictos, previsión y velocidad.

De especial importancia es la posibilidad de configurar la red, mediante los parámetros globales, para poner énfasis en aquellas propiedades que deseamos sean exhibidas durante la selección de acción.

3.4.1 Capacidad de planificación

Las redes de comportamiento exhiben una capacidad de planificación al considerar, de cierta forma, el efecto de una secuencia de acciones antes de llevarlas a cabo. Si una secuencia de módulos de habilidad es capaz de transformar la situación actual en la que se encuentra el agente, en aquella donde es posible dar cumplimiento a las metas globales, entonces los módulos de habilidad en esa secuencia resultarán altamente estimulados. Por el contrario, si una secuencia de acciones implica efectos negativos, los módulos de habilidad en esa secuencia serán inhibidos.

La relevancia de las metas se obtiene por la activación por metas globales del agente y la propagación hacia atrás a partir de los módulos de habilidad que pueden realizar las metas globales del agente. La relevancia de la situación actual y el comportamiento oportunista del agente, se obtienen por la activación por el medio ambiente y la propagación hacia adelante entre los módulos de habilidad que tiene correspondencia parcial con el medio ambiente. La prevención de conflictos y la relevancia de metas interactuantes se lleva a cabo por la inhibición proveniente de metas realizadas y la inhibición entre módulos conflictivos. Los máximos locales en el proceso de selección de acción pueden ser evitados si se permite que la dinámica de activación/inhibición se lleve a cabo durante el tiempo adecuado, fijando el umbral de activación θ lo suficientemente alto, de tal forma que la red evolucione hacia mejores patrones de activación. Finalmente, existe un sesgo a favor de la secuencia de módulos de habilidad que está siendo estimulada, ó plan en curso, debido a los niveles de activación acumulados.

Algunas diferencias con los sistemas de planificación construidos previamente en Inteligencia Artificial [Fikes y Nilsson 71], [Sacerdoti 74] son: Una red de comportamiento no construye una representación explícita del plan a ejecutar, pero expresa la “intención” de llevar a cabo ciertas acciones mediante altos niveles de activación en los módulos de habilidad correspondientes. Una red de comportamiento no tiene un módulo central que lleve a cabo un proceso de búsqueda, la dinámica de activación/inhibición entre los

operadores va resultando en la ejecución de los operadores apropiados. El resultado es que algunos de los problemas asociados a los árboles de búsqueda, como información duplicada en varias partes del árbol, crecimiento exponencial en relación con las dimensiones del problema, representación demasiado estricta, no están presentes en las redes de comportamiento. Como contra parte, las decisiones que produce una red de comportamiento son menos “racionales” que las producidas por los planificadores deliberativos tradicionales en Inteligencia Artificial.

3.4.2 Comportamiento orientado a metas.

El algoritmo presentado contribuye a que las metas globales del agente sean realizadas. Si g es una meta global del agente, entonces una cantidad γ de energía de activación es inyectada a aquellos módulos de habilidad que pueden realizar esa meta. A su vez, estos módulos llevarán a cabo una propagación de energía hacia aquellos predecesores que pueden hacer que sus precondiciones falsas sean verdaderas. Esta propagación hacia atrás contribuye a que los módulos de habilidad involucrados en el cumplimiento de g sean favorecidos con altos niveles de activación.

La dinámica de activación/inhibición favorece a aquellos módulos que contribuyen con diferentes metas globales del agente. De igual forma, favorece el cumplimiento de metas que tiene asociadas las cadenas más cortas entre los módulos que tienen correspondencia parcial con el medio ambiente y aquellos que realizan las metas. Los módulos que tienen menos competencia también resultan favorecidos, esto es, si un sólo módulo puede satisfacer la meta $g1$ y dos módulos pueden satisfacer la meta $g2$, es muy probable que $g1$ se lleve a cabo primero que $g2$. Esto se debe a que los módulos asociados a $g2$ tienen que compartir la cantidad de energía de activación γ . Observe que las precondiciones falsa de un módulo de habilidad pueden ser consideradas como submetas del agente, y por lo tanto presentan estas mismas propiedades.

La relevancia de las metas en la selección de acción puede ajustarse variando la tasa entre γ y σ . Si $\sigma = 0$ la selección de acciones por parte del agente está completamente guiada por las metas, lo cual produce un mecanismo de encadenamiento hacia atrás tradicional. En ese caso, el agente tomará menos oportunidad de las situaciones que se le presenten, será menos reactivo y la tendencia a favorecer el plan en curso será menor, lo cual repercute en la velocidad con que alcanza sus metas el agente. Idealmente deseamos que el agente tenga una fuerte influencia de las metas y tome ventaja de las oportunidades que se le presentan. Esto se logra eligiendo $\gamma > \sigma > 0$. La tasa óptima es dependiente de cada problema.

3.4.3 Comportamiento guiado por el medio ambiente

El algoritmo incrementa el nivel de activación de los módulos que son relevantes a las condiciones observadas en el medio ambiente. Esto es resultado del estímulo que experimentan aquellos módulos que tiene correspondencia parcial con la percepción del medio por parte del agente y la posterior propagación de energía hacia sus sucesores. La relevancia del medio ambiente en el proceso de selección de acción puede ajustarse mediante el parámetro σ . Al incrementar el valor de σ la influencia del medio ambiente es mayor. La propagación de energía de activación hacia adelante hace que los módulos con

mayor probabilidad de activarse, dadas las condiciones actuales del medio ambiente, sean estimulados incrementando su nivel de activación.

La capacidad de aprovechar oportunidades ha sido reconocida como central en el diseño de agentes autónomos [Maes 95].

3.4.4 Adaptabilidad

El proceso de selección de acciones presentado en este capítulo es completamente abierto en el sentido de que el medio ambiente y las metas pueden cambiar en tiempo de ejecución, repercutiendo rápidamente, en un cambio en los patrones de activación/inhibición para ajustarse a la nueva situación. Debido a un proceso continuo de percepción del medio ambiente y re-evaluación de qué metas han sido realizadas, el agente puede adaptarse fácilmente a situaciones imprevistas. El nivel de activación de los módulos juega un papel muy importante en cuanto a la adaptabilidad del agente. En cierta forma, el nivel de activación, representa la relevancia de un módulo a través del tiempo, lo cual permite que las situaciones imprevistas no impliquen una replanificación a partir de cero, y un ajuste de los niveles de activación acorde a la nueva situación sea suficiente. Es muy probable que esta adecuación lleve a la ejecución de otra cadena de módulos de habilidad, o plan alternativo.

Cuando γ y σ son relativamente pequeños en relación con π el sistema es menos adaptativo y presenta una marcada tendencia a favorecer el plan en curso. Esto se debe a que la dinámica de activación/inhibición entre los módulos de habilidad tiene un mayor impacto que la influencia del medio ambiente y de las metas. La elección de los valores adecuados para los parámetros debe evitar que el agente salte entre planes que contribuyen a diferentes metas sin lograr llegar a realizarlas.

La naturaleza distribuida de este algoritmo ofrece otro tipo de adaptabilidad debido a la tolerancia ante fallas. Si un módulo de habilidad presenta fallas, el agente es capaz de utilizar el resto de los módulos para tratar de realizar sus metas.

3.4.5 Tendencia a favorecer los planes en curso

El algoritmo exhibe una tendencia implícita a favorecer los planes en curso, con excepción de situaciones que requieren urgentemente que el agente realice una meta diferente a la del plan en curso. La presencia de este sesgo, puede explicarse en el hecho de que el nivel de activación de los módulos no es reinicializado cada vez que un módulo entra en activación. Como una consecuencia, la historia de su activación en el pasado juega un papel importante en la selección de acción, en particular cuando el impacto del medio ambiente y de las metas es relativamente pequeño en comparación con el nivel promedio de activación π . Pero aún cuando no es este el caso, el algoritmo exhibe dos tipos de sesgo. El sesgo horizontal se refiere a que el algoritmo presenta una tendencia a trabajar con una de sus metas a la vez hasta que la realiza, y sólo entonces considera otra meta. En general, hay una tendencia en atender primero las metas que tienen asociadas secuencias de módulos cortas. Los módulos de habilidad que contribuyen a que la meta en turno sea realizada tienden a ser estimulados. El sesgo vertical se refiere a que el algoritmo favorece la ejecución de módulos que satisfacen submetas del agente, precondiciones falsas, que contribuyen a la misma meta global.

3.4.6 Prevención de conflictos

Activar módulos de habilidad en un orden que no es el adecuado, puede incrementar dramáticamente la cantidad de acciones necesarias para llegar a la realización de las metas, e inclusive puede llevar a situaciones donde ya no es posible cumplir algunas de ellas. Por lo tanto, cualquier mecanismo de selección de acción debe ser capaz de mediar entre acciones conflictivas. Las reglas de inhibición de este algoritmo realizan esta mediación. Los módulos de habilidad que pueden deshacer una meta ya realizada, son inhibidos y su nivel de activación se reduce en un factor de δ . Si este parámetro es lo suficientemente alto, la selección de acciones protegerá metas globales y submetas que hallan sido realizadas. Es necesario establecer un equilibrio entre una selección de acciones que no toma en cuenta las metas que se han realizado y una que es incapaz de deshacer metas realizadas y por lo tanto puede llegar a generar situaciones donde ningún módulo de habilidad se activará, pues la única posibilidad es deshacer una meta ya realizada.

3.4.7 Previsión y velocidad

La selección de acción puede hacerse más ó menos previsor, variando el valor del parámetro θ . Si el umbral de activación θ es lo suficientemente alto como para permitir que la dinámica de activación/inhibición se de por más tiempo, la selección de acciones es más elaborada, pues se evitan máximos locales. Por otra parte, si el umbral es muy alto, la selección de acciones tomará demasiado tiempo, lo cual no es deseable, sobre todo en medios ambientes que cambian rápidamente. En contra parte, si el umbral de activación es lo suficientemente bajo, la velocidad en que el sistema realiza sus metas se incrementa, pero las acciones elegidas son menos orientadas a las metas, menos oportunistas y están más expuestas a los conflictos.

3.5 Consideraciones

Un factor importante a considerar en nuestra presentación del algoritmo es el relacionado con la complejidad del cómputo que éste lleva a cabo. Podemos observar similitudes entre la selección de acciones en una red de comportamiento y los procesos de búsqueda empleados tradicionalmente en IA, por lo cual podríamos suponer que el rendimiento del algoritmo se ve reducido en la medida que el número de módulos de habilidad de un agente aumentan. A continuación expondremos algunos argumentos [Maes 90] que nos hacen creer que las redes de comportamiento no presentan este problema.

El máximo número de pasos que el sistema necesita para seleccionar una acción está acotado por el ancho de la red de comportamiento multiplicado por alguna función lineal del umbral de activación. El ancho de la red es la distancia, en términos de módulos de habilidad, entre los módulos ejecutables y los módulos que realizan metas. Además, el algoritmo evalúa diferentes cursos de acción en paralelo, de tal forma que no es necesario reiniciar el proceso cuando un curso de acción no llega a realizarse, simplemente se da un cambio a otro de los cursos de acción que están siendo evaluados. Lo anterior, aunado al hecho de que el sistema no construye un árbol de búsqueda, ni mantiene representaciones de planes parciales, nos hace pensar que el cómputo requerido por las redes de comportamiento es menor al empleado en técnicas de búsqueda tradicionales.

La experiencia en el diseño de agente autónomos parece indicar que una red de comportamiento crece en profundidad y no en amplitud. Típicamente un agente autónomo debe enfrentar muchas metas que requiere cursos de acción de poca amplitud, en lugar de enfrentar metas que requieren de un número considerable de acciones para ser realizadas y por lo tanto, requieren más “planificación”. Esto nos hace pensar que el impacto del número de módulos de habilidad que componen un agente en su eficiencia no es tan grave. Por otra parte, aún cuando la red de comportamiento tuviera cursos de acción muy amplios, es posible que el algoritmo seleccione cursos de acción alternativos ya que no espera convergencia en los niveles de activación y el umbral de activación disminuye con el paso del tiempo. Debemos señalar que en este último caso, el comportamiento del agente no tendería al óptimo.

El hecho de que las mismas reglas de activación/inhibición se apliquen por igual a todos los módulos de habilidad, que además, están conectados por ligas fijas representadas localmente, abre interesantes oportunidades de paralelizar el algoritmo, lo cual implicaría una mejora considerable en la velocidad de éste.

Un segundo factor a considerar en esta sección es el relacionado con la ausencia de variables. El algoritmo de redes de comportamiento no incorpora variables. En su lugar emplea una *representación directa* como la utilizada en Pengi [Agre y Chapman 90] presentada en el capítulo anterior. Una consecuencia de la ausencia de variables en el algoritmo es que todos los módulos de habilidad ó operadores deben estar instanciados. Así mismo, no es posible plantear metas del *estilo muevete-a(x,y)*. Aunque esto pudiera parecer una limitación sería, en el capítulo anterior presentamos como el robot Toto [Mataric 89] hace uso de mapas de navegación utilizando *representación directa* que posibilitan metas de tipo *muevete-a-la-puerta*. Es posible incorporar variables en el algoritmo creando copias de un módulo de habilidad cada vez que este es instanciado con valores de variables distintos, sin embargo esta opción requeriría una cantidad mucho mayor de cómputo. La tendencia es utilizar *representación directa* y replantearnos la forma en que un agente puede ser “instruido” sobre las metas a realizar.

Debemos mencionar que el algoritmo presenta algunas limitaciones. Esporádicamente, la selección de acción de un agente puede caer en ciclos. Esto se debe a que el sistema no mantiene una historia de las acciones que ha realizado en el pasado. El hecho de que los ciclos se presenten en contadas ocasiones se debe a la interacción del agente con su medio ambiente. En un medio ambiente *dinámico* es muy probable que las condiciones del medio cambien constantemente, lo cual tiene un impacto en la activación propagada a través de la red de comportamiento y en consecuencia la selección de acción podría salir del ciclo en que se encuentra. Una posible solución es incorporar un mecanismo de *habitación* con el cual cada módulo de habilidad que se activa es menos propenso a activarse en el futuro, esto pueden lograrse usando umbrales de activación locales que varían en el tiempo.

Un segundo problema está relacionado con la selección de parámetros. No es claro cómo podemos seleccionar un conjunto de parámetros dado una aplicación específica. Esta selección depende de las propiedades que deseamos exhiba la red de comportamiento, pero además depende de factores como el tamaño y la estructura de la red de comportamiento. Este problema y sus consecuencias en el estudio y enseñanza de las redes de

comportamiento nos sugirieron la necesidad de implementar una herramienta como ABC descrita en capítulo 5.

Capítulo 3

Redes de Comportamiento

3.1 ¿Cómo hacer lo correcto?

Hemos elegido las redes de comportamiento para implementar un mecanismo de selección de acción y un ambiente de simulación que nos permita experimentar con diversos agentes autónomos y la forma en que estos realizan sus metas eligiendo correctamente sus acciones.

Las propiedades exhibidas por las redes de comportamiento fueron un factor decisivo en esta elección. Este mecanismo exhibe un comportamiento guiado por metas y relevante a la situación del agente, es adaptativo, tolerante ante fallas, persistente en el plan de acción en curso, reactivo y rápido [Maes 89,90b,90c,91] [Franklin95]. Además, las redes de comportamiento son un mecanismo de selección de acción descentralizado y dinámicamente reconfigurable, lo que las hace viables para implementar agentes con actividad situada [Hendriks-Jansen 96].

En este capítulo presentaremos un algoritmo de activación / inhibición en una red de comportamiento, así como el modelo matemático de este mecanismo de selección de acción. Posteriormente, estudiaremos la forma en que una red de comportamiento puede ser configurada a través de sus parámetros globales y terminaremos presentando algunas limitaciones presentes en este mecanismo.

Es importante que durante la lectura de este capítulo tengamos presente que este mecanismo de selección de acción busca verificar la siguiente hipótesis: la selección de acción puede ser modelada como una propiedad emergente de una dinámica de activación / inhibición entre las diferentes acciones que el agente puede realizar, evitando la presencia de módulos burocráticos que funcionen como formas globales de control determinando que acción resultará activada o inhibida.

3.2 Descripción del algoritmo

El algoritmo para elegir la acción correcta basado en redes de comportamiento define un agente como un conjunto de habilidades o destrezas representados como módulos de habilidad. Un módulo de habilidad m_i puede ser descrito por la tupla $(c_i, a_i, d_i, \alpha_i)$. La lista c_i

contiene las precondiciones que deben ser observadas para que el módulo de habilidad m_i se active. Las listas a_i y d_i representan los efectos esperados al activarse el módulo de habilidad m_i : La lista a_i contiene las condiciones que se espera sean verdaderas después de activarse el módulo y la lista d_i las que se espera dejen de serlo. El valor α_i representa el nivel de activación que ha acumulado el módulo. La figura 3.1 representa un módulo de habilidad. Como puede observar, los módulos de habilidad se asemejan a los operadores empleados en los sistemas de planificación clásicos [Fikes y Nilsson 71].

Cuando todas las precondiciones de un módulo de habilidad se observan verdaderas, decimos que éste es *ejecutable*. Ser ejecutable es una condición necesaria para que un módulo de habilidad pueda activarse.

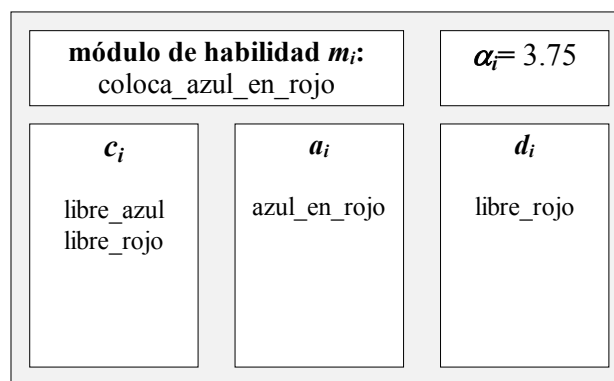


Figura 3.1 Imagine un agente que actúa en un medio ambiente donde puede encontrar cubos de diferentes colores. La idea es que este agente sea capaz de construir torres con esos cubos. Una destreza deseable en este agente es que sea capaz de colocar un cubo sobre otro. En este caso el módulo de habilidad m_i se encargará de colocar un cubo azul sobre uno rojo. Para ello es necesario que el cubo azul y el rojo estén libres. Estas precondiciones se encuentran en la lista c_i . Si el módulo de habilidad m_i se activa, el resultado esperado de sus acciones es que el cubo azul este sobre el rojo (a_i) y que el cubo rojo deje de percibirse como libre (d_i). El nivel de activación acumulado en este módulo es de 3.75 (α_i).

Los módulos de habilidad que conforman un agente se organizan en una red, relacionados por medio de tres tipos de ligas: *sucesor*, *predecesor* y *conflicto*. La composición de cada módulo de habilidad m_i en términos de c_i , a_i , y d_i definen esta red de la siguiente forma:

Hay una *liga de sucesor* (figura 3.2) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in a_i \cap c_j$. Decimos que el módulo de habilidad m_i tiene como sucesor al módulo m_j debido a que las precondiciones p de m_j serían verdaderas de activarse m_i , por lo que existe la posibilidad de que m_j se active después de m_i . Observe que es posible que existan más de una liga del mismo tipo entre dos módulos de habilidad.

Hay una *liga de predecesor* (figura 3.3) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in c_i \cap a_j$. Decimos que el módulo de habilidad m_i tiene como predecesor al módulo m_j , debido a que las precondiciones p de m_i serían verdaderas

de activarse m_j , por lo que existe la probabilidad de que m_j se active antes de m_i . Observe que por cada liga predecesor, existe una de sucesor en sentido contrario.

Hay una *liga de conflicto* (figura 3.4) del módulo de habilidad m_i al módulo de habilidad m_j por cada proposición p tal que $p \in c_i \cap d_j$. Decimos que el módulo m_i tiene como módulo conflictivo al módulo m_j debido a que las precondiciones p de m_i dejarían de observarse verdaderas de activarse m_j , por lo que m_i no podría activarse. Observe que pueden existir conflictos mutuos entre dos módulos.

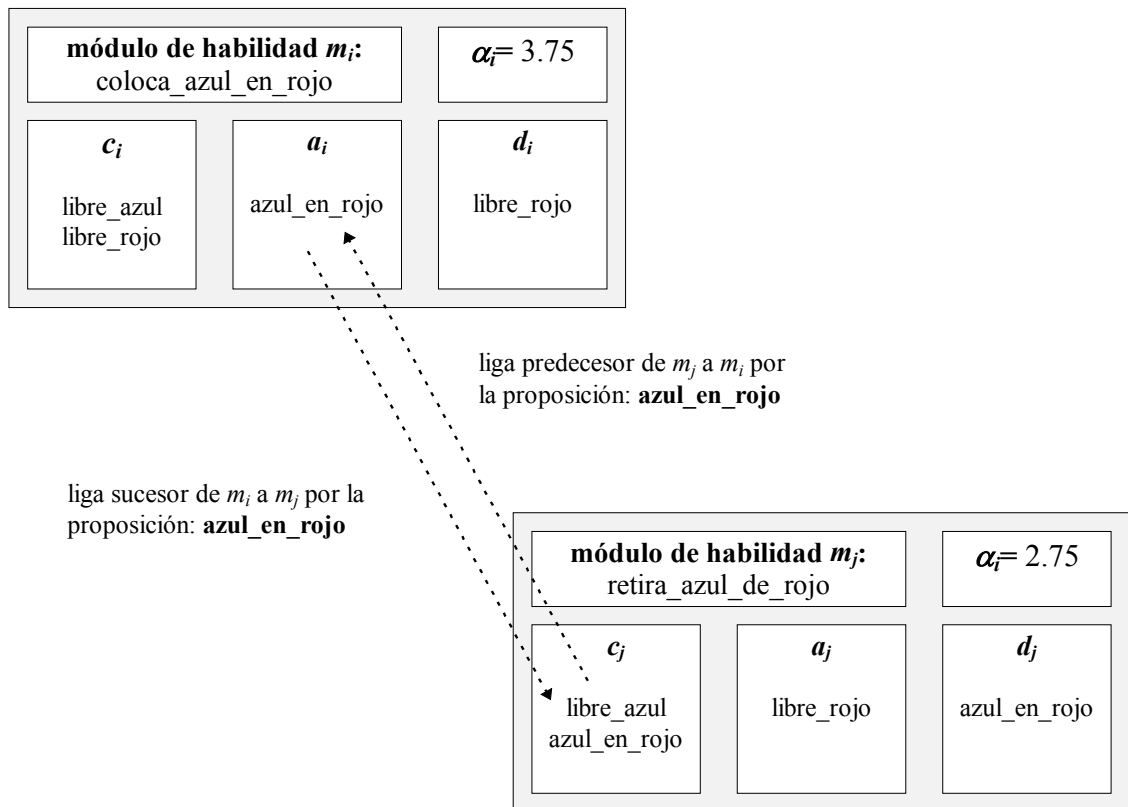


Figura 3.2 La existencia de una liga de sucesor implica la existencia de una liga de predecesor en sentido contrario. En este caso, hay una liga de predecesor del módulo m_j al módulo m_i por la proposición azul_en_rojo. Observe que también la proposición libre_rojo define ligas entre estos dos módulos. Por claridad, únicamente se muestran las ligas originadas por azul_en_rojo.

La idea detrás de estas ligas es que permitan una dinámica de activación/inhibición entre los módulos de habilidad que componen un agente. Esta dinámica tiene efecto sobre el nivel de activación de los módulos, de tal forma que después de un tiempo, aquellos módulos que representan la acciones adecuadas a la situación del agente y las metas que debe llevar a cabo acumulen una mayor activación. Si el nivel de activación de un módulo

ejecutable ha rebasado cierto umbral, entonces se activará desplegando sus habilidades. En esta dinámica de activación/inhibición intervienen los siguientes factores:

Activación proveniente del medio ambiente. Decimos que un módulo de habilidad m_i está en correspondencia parcial con el medio ambiente, si al menos una de sus precondiciones c_i se observa verdadera. Los módulos de habilidad que están en correspondencia parcial con el medio ambiente se estimulan, incrementando su nivel de activación, con la intención de que los módulos que son relevantes a una situación en el medio ambiente sean los que acumulen un mayor nivel de activación. La situación actual del medio ambiente se obtiene utilizando un conjunto de sensores virtuales, los cuales utilizan los datos generados por sensores reales para determinar cuando una proposición es verdadera. La situación actual puede considerarse como la unión de las observaciones de todos los sensores virtuales.

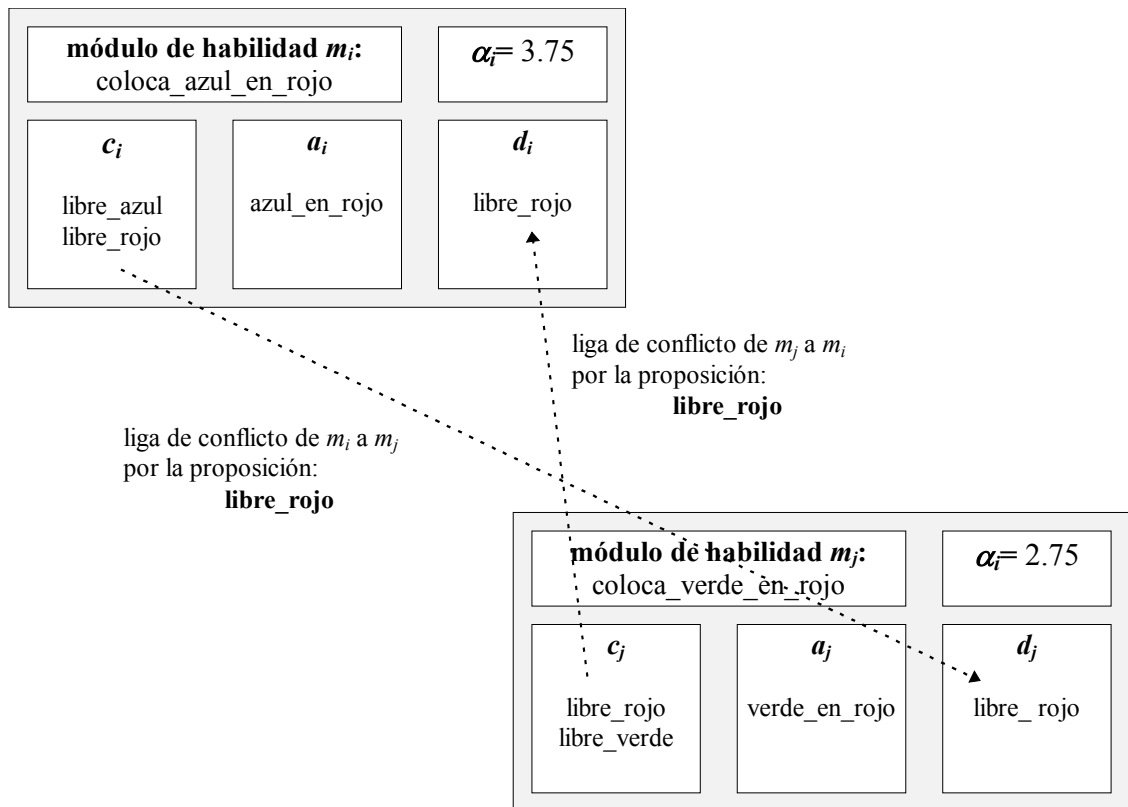


Figura 3.3 Al agregar un nuevo módulo de habilidad para colocar un cubo verde sobre un cubo rojo podemos observar un conflicto mutuo con el módulo coloca_azul_en_rojo: Si m_j se activa, el cubo rojo ya no estará libre y m_i no podrá activarse. De igual forma, si m_i se activa el cubo rojo ya no estaría libre y m_j no podría activarse.

Activación proveniente de las metas globales del agente. Un módulo de habilidad m_i puede llevar a cabo una meta global del agente, si ésta es miembro de su lista a_i . Los módulos de habilidad que pueden realizar alguna de las metas globales del agente se estimulan incrementando su activación. Las metas globales pueden ser de dos tipos:

permanentes, en cuyo caso estimularan al agente permanentemente; y temporales, las cuales una vez realizadas dejan de estimular al agente.

Inhibición proveniente de las metas realizadas. Decimos que un módulo de habilidad m_i puede deshacer una meta global del agente, si ésta es miembro de su lista d_i . Los módulos de habilidad que con sus acciones pueden deshacer una meta previamente realizada se inhiben, por lo que su nivel de activación sufre un decremento.

Los tres factores mencionados se consideran continuamente, de tal forma que la presencia de cambios imprevistos en el medio ambiente y el planteamiento de nuevas metas al agente, se ven reflejados inmediatamente en los niveles de activación de cada módulo de habilidad.

Además de la influencia del medio ambiente y de las metas globales del agente, la red de comportamiento presenta una dinámica entre los módulos de habilidad que obedece a los siguientes factores:

Activación de sucesores. Un módulo de habilidad ejecutable m_i estimula a sus sucesores m_j por cada proposición falsa p que define la liga de sucesión entre ellos, esto es $p \in a_i \cap c_j$. Esta propagación hacia adelante de energía de activación es deseable, pues la activación del módulo m_i contribuirá a que las precondiciones p de sus sucesores sean verdaderas, incrementando la posibilidad de que estos lleguen a ser ejecutables.

Activación de predecesores. Un módulo de habilidad m_i que no es ejecutable estimula a sus predecesores m_j por cada proposición falsa p que define la liga de predecesor entre ellos, esto es $p \in c_i \cap a_j$. Esta propagación hacia atrás de energía de activación es deseable, pues la activación de los sucesores m_j haría que las precondiciones p del módulo m_i cambiasen de falsas a verdaderas.

Inhibición por conflictos. Todo módulo de habilidad m_i , ejecutable o no, inhibe a aquellos módulos $m_j \neq m_i$ con los que tiene conflictos, por cada proposición p que es verdadera y define la liga de conflicto entre ellos, esto es $p \in c_i \cap d_j$. Decimos que $m_j \neq m_i$ si $c_j \neq c_i \vee a_j \neq a_i \vee d_j \neq d_i$. Para estas listas (c, a y d) se cumple que $l_j \neq l_i$ si $(\exists p) [(p \in l_j \wedge p \notin l_i) \vee (p \in l_j \wedge p \notin l_i)]$. Observe que los módulos `coloca_azul_en_rojo` y `coloca_verde_en_rojo` de la figura 3.3 presentarían conflicto con ellos mismos por la proposición `libre_rojo`, es importante señalar que un módulo no puede inhibirse a si mismo por conflicto dado que $m_j \neq m_i$. En el caso de presentarse un conflicto mutuo entre dos módulos de habilidad, el módulo con mayor activación inhibe al de menor activación, lo cual evita que se presente el fenómeno de mutua eliminación entre módulos relevantes. En caso de querer simular el comportamiento irrelevante [Maes 91] que se presenta en algunos animales, la inhibición entre módulos con conflictos mutuos se da en ambos sentidos.

El algoritmo consiste en un ciclo en el que se llevan a cabo los siguientes cálculos para todos los módulos de habilidad que componen al agente:

1. Se calcula el impacto del medio ambiente, las metas y las metas realizadas.
2. Se calcula el impacto de la dinámica de activación/inhibición entre los módulos
3. Se normalizan los niveles de activación para mantener su suma constante
4. El módulo de habilidad que cumple con los siguientes condiciones se activa, su nivel de activación se fija en cero y el umbral regresa a su valor inicial:

- a) es ejecutable
 - b) su nivel de activación sobrepasa cierto umbral
 - c) tiene el nivel de activación más alto entre los módulos que cumplen a y b (en caso de empate se elige uno aleatoriamente)
5. Si no hay un módulo de habilidad que se active, el umbral se decrementa en un 10%

Cuatro parámetros globales pueden ser utilizados para ajustar la dinámica de activación/inhibición entre los módulos de habilidad y, por lo tanto, la selección de acciones por parte del agente:

- θ el umbral que deben superar los módulos de habilidad para activarse. Su valor disminuye un 10% en cada tiempo del ciclo principal en que ningún módulo de habilidad entra en actividad y regresa a su valor original cuando un módulo se activa. Este parámetro está relacionado con π que es el nivel promedio de activación de los módulos que componen un agente.
- σ la cantidad de energía que una proposición que se observa verdadera inyecta a la red
- γ la cantidad de energía que una meta inyecta a la red
- δ la cantidad de energía que las metas realizadas retiran de la red

Algunas de las propiedades globales que resulta interesante observar son: la secuencia de módulos de habilidad que entran en actividad, la optimalidad de esa secuencia (su cálculo es dependiente de cada problema) y la velocidad con que las metas son realizadas (la relación entre el número de tiempos que se ha repetido el ciclo y la cantidad de módulos de habilidad que han sido activados).

3.3 Modelo matemático

3.3.1 Módulo de habilidad

Un módulo de habilidad m_i se define como la tupla: $m_i = (c_i, a_i, d_i, \alpha_i)$ donde:

- c_i Conjunto de proposiciones que necesitan observarse verdaderas para que m_i pueda activarse (precondiciones).
- a_i Conjunto de proposiciones que se observaran verdaderas al activarse m_i .
- d_i Conjunto de proposiciones que dejaran de observarse verdaderas al activarse m_i .
- α_i Nivel de activación del m_i , $\alpha_i \in \mathfrak{R}^+$.

3.3.2 Agente

Un agente A , capaz de exhibir n habilidades ó acciones, se define como: $A = \{m_1 \dots m_n\}$

3.3.3 Proposiciones

Sea $p_i = c_i \cup a_i \cup d_i$ el conjunto de proposiciones utilizados para definir el módulo de habilidad m_i .

Decimos que $P_A = p_1 \cup p_2 \cup \dots \cup p_n$ es el conjunto de proposiciones utilizado para definir al agente A .

Algunas proposiciones $p \in P_A$ están relacionadas con un sensor virtual que determina su valor de verdad con base en la información proveniente de sensores reales.

Dado un agente A , definimos:

$M(p) = \{m_i: m_i \in A \wedge p \in c_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de c_i .

$A(p) = \{m_i: m_i \in A \wedge p \in a_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de a_i .

$D(p) = \{m_i: m_i \in A \wedge p \in d_i, 1 \leq i \leq n\}$ como el conjunto de todos los módulos de habilidad m_i en A que tienen a la proposición p como miembro de d_i .

3.3.4 Percepción

La función $S(t) = \{p: p \in P_A \wedge p \text{ es verdadera al tiempo } t\}$ regresa el conjunto de proposiciones que se observan como verdaderas en el medio ambiente al tiempo t .

Esta función es dependiente del dominio del agente y descansa en la existencia de sensores virtuales que, como mencionamos anteriormente, determinan el valor de verdad de su proposición asociada con base en la información proveniente de un sensor real.

Sea $S_p(t)$ el sensor virtual asociado a la proposición $p \in P_A$ y $SR_p(t)$ el sensor real que provee la información necesaria para determinar el valor de verdad de p , entonces:

$$S_p(t) = \begin{cases} \{p\} & \text{Si la información de } SR_p(t) \text{ determina que } p \text{ es verdadera} \\ \{\} & \text{En cualquier otro caso} \end{cases}$$

Desde este punto de vista la función $S(t)$ regresa el conjunto formado por la unión de todos los sensores virtuales que posee el agente A .

3.3.5 Metas globales del agente

La función $G(t) = \{p: p \in P_A \wedge p \text{ es una meta del agente al tiempo } t\}$ regresa el conjunto de proposiciones que son metas globales de la sociedad de módulos de habilidad que conforman al agente A . Esta función es dependiente del dominio.

La función $GR(t) = \{S(t) \cap G(t)\}$ regresa el conjunto de metas que el agente ha realizado al tiempo t .

3.3.6 Módulo de habilidad ejecutable.

La función:

$$E(m_i, t) = \begin{cases} 1 & \text{Si } c_i \supset S(t) \\ 0 & \text{en cualquier otro caso} \end{cases}$$

regresa 1 si todas las precondiciones del módulo m_i se observan verdaderas al tiempo t , en cuyo caso decimos que el módulo m_i es ejecutable.

3.3.7 Parámetros.

π es el nivel de activación promedio. Dado un agente A_n :

$$\sum_{i=1..n} \alpha_i(t) = \pi \cdot n, \forall t$$

θ el umbral de activación. Para que un módulo de habilidad m_i pueda activarse es necesario que $\alpha_i > \theta$ y que $E(m_i, t) = 1$

σ es un valor representando la influencia del medio ambiente en el agente ó la cantidad de energía que una proposición que se observa verdadera inyecta en la red de comportamiento.

γ es un valor representando la influencia de las metas en el agente ó la cantidad de energía que una meta inyecta en la red de comportamiento.

δ es un valor representando la influencia de las metas realizadas y conflictos en el agente ó la cantidad de energía que una meta realizada retira de la red de comportamiento.

Los parámetros $\pi, \theta, \sigma, \gamma, \delta \in \mathbb{R}^+$

3.3.8 Dinámica de Activación / Inhibición

Activación por influencia del medio ambiente. La cantidad de energía que recibe un módulo m_i del medio ambiente, por correspondencia parcial con éste es:

$$entrada_por_estado(m_i, t) = \sum_{\forall p \in P} \sigma \frac{1}{|M(p)|} \frac{1}{|c_i|}$$

donde $P = S(t) \cap c_i$ y $|X|$ indica la cardinalidad del conjunto X .

Activación por influencia de las metas. La cantidad de energía que un módulo m_i recibe debido a su capacidad de realizar alguna meta del agente es:

$$entrada_por_metas(m_i, t) = \sum_{\forall p \in P} \gamma \frac{1}{|A(p)|} \frac{1}{|a_i|}$$

donde $P = G(t) \cap a_i$

Inhibición por metas realizadas. La cantidad de energía que es retirada de un módulo m_i cuya acción puede afectar una meta previamente realizada es:

$$salida_por_metas_realizadas(m_i, t) = \sum_{\forall p \in P} \delta \frac{1}{|D(p)|} \frac{1}{|d_i|}$$

donde $P = GR(t) \cap d_i$

Propagación hacia adelante. La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de predecesor es:

$$factor_fwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{\sigma}{\gamma} \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 1 \\ 0 & \text{Si } E(m_i, t) = 0 \end{cases}$$

donde $P = \{p: p \in c_i \cap a_j \wedge p \notin S(t)\}$

De esta forma, la cantidad de energía que un módulo m_i , recibe por propagación hacia adelante es:

$$fwp(m_i, t) = \sum_{\forall m \in A} factor_fwp(m_i, m, t)$$

Propagación hacia atrás. La cantidad de energía que un módulo m_i recibe de un módulo m_j a través de sus ligas de sucesor es:

$$factor_bwp(m_i, m_j, t) = \begin{cases} \sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|A(p)|} \frac{1}{|a_i|} & \text{Si } E(m_i, t) = 0 \\ 0 & \text{Si } E(m_i, t) = 1 \end{cases}$$

donde $P = \{p: p \in a_i \cap c_j \wedge p \in S(t)\}$

De esta forma, la cantidad de energía que un módulo m_i , recibe por propagación hacia adelante es:

$$bwp(m_i, t) = \sum_{\forall m \in A} factor_bwp(m_i, m, t)$$

Inhibición por conflicto. La cantidad de energía que es retirada de un módulo m_i por presentar conflictos con otro módulo m_j a través de sus ligas de conflicto es:

$$factor_confp(m_i, m_j, t) = \begin{cases} 0 & \text{Si } \alpha_i(t-1) > \alpha_j(t-1) \wedge (\exists p)(p \in S(t) \cap c_i \cap d_j) \\ \max \left[\sum_{\forall p \in P} \alpha_j(t-1) \frac{1}{|D(p)|} \frac{1}{|d_i|}, \alpha_i(t-1) \right] & \text{En cualquier otro caso} \end{cases}$$

donde $P = \{p: p \in d_i \cap c_j \wedge p \in S(t)\}$

De esta forma, la cantidad de energía que es retirada de un módulo m_i , por inhibición debida a conflictos es:

$$confp(m_i, t) = \sum_{\forall m \in A \wedge m \neq m_i} factor_confp(m_i, m, t)$$

Nivel de activación. El nivel de activación del módulo m_i al tiempo t se define como:

$$\alpha(m_i, t) = \begin{cases} 0 & \text{Si } t = 0 \\ normaliza \left[\begin{array}{l} \alpha(m_i, t-1) + \\ entrada_por_estado(m_i, t) + \\ entrada_por_metas(m_i, t) - \\ salida_por_metas_realizadas(m_i, t) + \\ fwp(m_i, t) + bwp(m_i, t) - confp(m_i, t) \end{array} \right] & \text{Si } t > 0 \end{cases}$$

donde *normaliza* es una función que se encarga de que la activación total del agente A_n permanezca constante en π .

3.3.9 Módulo de habilidad activo

El módulo de habilidad m_i que se activa al tiempo t está determinado por:

$$activo(t, m_i) = \begin{cases} 1 & \text{Si } \begin{cases} \alpha(m_i, t) > \theta \wedge \\ E(m_i, t) \wedge \\ \alpha(m_i, t) > \alpha(m_{j \neq i}, t) \end{cases} \\ 0 & \text{En cualquier otro caso} \end{cases}$$

3.4 Propiedades y configuración de parámetros

El algoritmo de selección de acciones presentado exhibe propiedades interesantes que discutiremos en esta sección, entre ellas: capacidad de planificación, comportamiento orientado a metas, comportamiento guiado por el medio ambiente, adaptabilidad, tendencia a favorecer los planes en curso, prevención de conflictos, previsión y velocidad.

De especial importancia es la posibilidad de configurar la red, mediante los parámetros globales, para poner énfasis en aquellas propiedades que deseamos sean exhibidas durante la selección de acción.

3.4.1 Capacidad de planificación

Las redes de comportamiento exhiben una capacidad de planificación al considerar, de cierta forma, el efecto de una secuencia de acciones antes de llevarlas a cabo. Si una secuencia de módulos de habilidad es capaz de transformar la situación actual en la que se encuentra el agente, en aquella donde es posible dar cumplimiento a las metas globales, entonces los módulos de habilidad en esa secuencia resultarán altamente estimulados. Por el contrario, si una secuencia de acciones implica efectos negativos, los módulos de habilidad en esa secuencia serán inhibidos.

La relevancia de las metas se obtiene por la activación por metas globales del agente y la propagación hacia atrás a partir de los módulos de habilidad que pueden realizar las metas globales del agente. La relevancia de la situación actual y el comportamiento oportunista del agente, se obtienen por la activación por el medio ambiente y la propagación hacia adelante entre los módulos de habilidad que tiene correspondencia parcial con el medio ambiente. La prevención de conflictos y la relevancia de metas interactuantes se lleva a cabo por la inhibición proveniente de metas realizadas y la inhibición entre módulos conflictivos. Los máximos locales en el proceso de selección de acción pueden ser evitados si se permite que la dinámica de activación/inhibición se lleve a cabo durante el tiempo adecuado, fijando el umbral de activación θ lo suficientemente alto, de tal forma que la red evolucione hacia mejores patrones de activación. Finalmente, existe un sesgo a favor de la secuencia de módulos de habilidad que está siendo estimulada, ó plan en curso, debido a los niveles de activación acumulados.

Algunas diferencias con los sistemas de planificación construidos previamente en Inteligencia Artificial [Fikes y Nilsson 71], [Sacerdoti 74] son: Una red de comportamiento no construye una representación explícita del plan a ejecutar, pero expresa la “intención” de llevar a cabo ciertas acciones mediante altos niveles de activación en los módulos de habilidad correspondientes. Una red de comportamiento no tiene un módulo central que lleve a cabo un proceso de búsqueda, la dinámica de activación/inhibición entre los

operadores va resultando en la ejecución de los operadores apropiados. El resultado es que algunos de los problemas asociados a los árboles de búsqueda, como información duplicada en varias partes del árbol, crecimiento exponencial en relación con las dimensiones del problema, representación demasiado estricta, no están presentes en las redes de comportamiento. Como contra parte, las decisiones que produce una red de comportamiento son menos “racionales” que las producidas por los planificadores deliberativos tradicionales en Inteligencia Artificial.

3.4.2 Comportamiento orientado a metas.

El algoritmo presentado contribuye a que las metas globales del agente sean realizadas. Si g es una meta global del agente, entonces una cantidad γ de energía de activación es inyectada a aquellos módulos de habilidad que pueden realizar esa meta. A su vez, estos módulos llevarán a cabo una propagación de energía hacia aquellos predecesores que pueden hacer que sus precondiciones falsas sean verdaderas. Esta propagación hacia atrás contribuye a que los módulos de habilidad involucrados en el cumplimiento de g sean favorecidos con altos niveles de activación.

La dinámica de activación/inhibición favorece a aquellos módulos que contribuyen con diferentes metas globales del agente. De igual forma, favorece el cumplimiento de metas que tiene asociadas las cadenas más cortas entre los módulos que tienen correspondencia parcial con el medio ambiente y aquellos que realizan las metas. Los módulos que tienen menos competencia también resultan favorecidos, esto es, si un sólo módulo puede satisfacer la meta $g1$ y dos módulos pueden satisfacer la meta $g2$, es muy probable que $g1$ se lleve a cabo primero que $g2$. Esto se debe a que los módulos asociados a $g2$ tienen que compartir la cantidad de energía de activación γ . Observe que las precondiciones falsa de un módulo de habilidad pueden ser consideradas como submetas del agente, y por lo tanto presentan estas mismas propiedades.

La relevancia de las metas en la selección de acción puede ajustarse variando la tasa entre γ y σ . Si $\sigma = 0$ la selección de acciones por parte del agente está completamente guiada por las metas, lo cual produce un mecanismo de encadenamiento hacia atrás tradicional. En ese caso, el agente tomará menos oportunidad de las situaciones que se le presenten, será menos reactivo y la tendencia a favorecer el plan en curso será menor, lo cual repercute en la velocidad con que alcanza sus metas el agente. Idealmente deseamos que el agente tenga una fuerte influencia de las metas y tome ventaja de las oportunidades que se le presentan. Esto se logra eligiendo $\gamma > \sigma > 0$. La tasa óptima es dependiente de cada problema.

3.4.3 Comportamiento guiado por el medio ambiente

El algoritmo incrementa el nivel de activación de los módulos que son relevantes a las condiciones observadas en el medio ambiente. Esto es resultado del estímulo que experimentan aquellos módulos que tiene correspondencia parcial con la percepción del medio por parte del agente y la posterior propagación de energía hacia sus sucesores. La relevancia del medio ambiente en el proceso de selección de acción puede ajustarse mediante el parámetro σ . Al incrementar el valor de σ la influencia del medio ambiente es mayor. La propagación de energía de activación hacia adelante hace que los módulos con

mayor probabilidad de activarse, dadas las condiciones actuales del medio ambiente, sean estimulados incrementando su nivel de activación.

La capacidad de aprovechar oportunidades ha sido reconocida como central en el diseño de agentes autónomos [Maes 95].

3.4.4 Adaptabilidad

El proceso de selección de acciones presentado en este capítulo es completamente abierto en el sentido de que el medio ambiente y las metas pueden cambiar en tiempo de ejecución, repercutiendo rápidamente, en un cambio en los patrones de activación/inhibición para ajustarse a la nueva situación. Debido a un proceso continuo de percepción del medio ambiente y re-evaluación de qué metas han sido realizadas, el agente puede adaptarse fácilmente a situaciones imprevistas. El nivel de activación de los módulos juega un papel muy importante en cuanto a la adaptabilidad del agente. En cierta forma, el nivel de activación, representa la relevancia de un módulo a través del tiempo, lo cual permite que las situaciones imprevistas no impliquen una replanificación a partir de cero, y un ajuste de los niveles de activación acorde a la nueva situación sea suficiente. Es muy probable que esta adecuación lleve a la ejecución de otra cadena de módulos de habilidad, o plan alternativo.

Cuando γ y σ son relativamente pequeños en relación con π el sistema es menos adaptativo y presenta una marcada tendencia a favorecer el plan en curso. Esto se debe a que la dinámica de activación/inhibición entre los módulos de habilidad tiene un mayor impacto que la influencia del medio ambiente y de las metas. La elección de los valores adecuados para los parámetros debe evitar que el agente salte entre planes que contribuyen a diferentes metas sin lograr llegar a realizarlas.

La naturaleza distribuida de este algoritmo ofrece otro tipo de adaptabilidad debido a la tolerancia ante fallas. Si un módulo de habilidad presenta fallas, el agente es capaz de utilizar el resto de los módulos para tratar de realizar sus metas.

3.4.5 Tendencia a favorecer los planes en curso

El algoritmo exhibe una tendencia implícita a favorecer los planes en curso, con excepción de situaciones que requieren urgentemente que el agente realice una meta diferente a la del plan en curso. La presencia de este sesgo, puede explicarse en el hecho de que el nivel de activación de los módulos no es reinicializado cada vez que un módulo entra en activación. Como una consecuencia, la historia de su activación en el pasado juega un papel importante en la selección de acción, en particular cuando el impacto del medio ambiente y de las metas es relativamente pequeño en comparación con el nivel promedio de activación π . Pero aún cuando no es este el caso, el algoritmo exhibe dos tipos de sesgo. El sesgo horizontal se refiere a que el algoritmo presenta una tendencia a trabajar con una de sus metas a la vez hasta que la realiza, y sólo entonces considera otra meta. En general, hay una tendencia en atender primero las metas que tienen asociadas secuencias de módulos cortas. Los módulos de habilidad que contribuyen a que la meta en turno sea realizada tienden a ser estimulados. El sesgo vertical se refiere a que el algoritmo favorece la ejecución de módulos que satisfacen submetas del agente, precondiciones falsas, que contribuyen a la misma meta global.

3.4.6 Prevención de conflictos

Activar módulos de habilidad en un orden que no es el adecuado, puede incrementar dramáticamente la cantidad de acciones necesarias para llegar a la realización de las metas, e inclusive puede llevar a situaciones donde ya no es posible cumplir algunas de ellas. Por lo tanto, cualquier mecanismo de selección de acción debe ser capaz de mediar entre acciones conflictivas. Las reglas de inhibición de este algoritmo realizan esta mediación. Los módulos de habilidad que pueden deshacer una meta ya realizada, son inhibidos y su nivel de activación se reduce en un factor de δ . Si este parámetro es lo suficientemente alto, la selección de acciones protegerá metas globales y submetas que hallan sido realizadas. Es necesario establecer un equilibrio entre una selección de acciones que no toma en cuenta las metas que se han realizado y una que es incapaz de deshacer metas realizadas y por lo tanto puede llegar a generar situaciones donde ningún módulo de habilidad se activará, pues la única posibilidad es deshacer una meta ya realizada.

3.4.7 Previsión y velocidad

La selección de acción puede hacerse más ó menos previsor, variando el valor del parámetro θ . Si el umbral de activación θ es lo suficientemente alto como para permitir que la dinámica de activación/inhibición se de por más tiempo, la selección de acciones es más elaborada, pues se evitan máximos locales. Por otra parte, si el umbral es muy alto, la selección de acciones tomará demasiado tiempo, lo cual no es deseable, sobre todo en medios ambientes que cambian rápidamente. En contra parte, si el umbral de activación es lo suficientemente bajo, la velocidad en que el sistema realiza sus metas se incrementa, pero las acciones elegidas son menos orientadas a las metas, menos oportunistas y están más expuestas a los conflictos.

3.5 Consideraciones

Un factor importante a considerar en nuestra presentación del algoritmo es el relacionado con la complejidad del cómputo que éste lleva a cabo. Podemos observar similitudes entre la selección de acciones en una red de comportamiento y los procesos de búsqueda empleados tradicionalmente en IA, por lo cual podríamos suponer que el rendimiento del algoritmo se ve reducido en la medida que el número de módulos de habilidad de un agente aumentan. A continuación expondremos algunos argumentos [Maes 90] que nos hacen creer que las redes de comportamiento no presentan este problema.

El máximo número de pasos que el sistema necesita para seleccionar una acción está acotado por el ancho de la red de comportamiento multiplicado por alguna función lineal del umbral de activación. El ancho de la red es la distancia, en términos de módulos de habilidad, entre los módulos ejecutables y los módulos que realizan metas. Además, el algoritmo evalúa diferentes cursos de acción en paralelo, de tal forma que no es necesario reiniciar el proceso cuando un curso de acción no llega a realizarse, simplemente se da un cambio a otro de los cursos de acción que están siendo evaluados. Lo anterior, aunado al hecho de que el sistema no construye un árbol de búsqueda, ni mantiene representaciones de planes parciales, nos hace pensar que el cómputo requerido por las redes de comportamiento es menor al empleado en técnicas de búsqueda tradicionales.

La experiencia en el diseño de agente autónomos parece indicar que una red de comportamiento crece en profundidad y no en amplitud. Típicamente un agente autónomo debe enfrentar muchas metas que requiere cursos de acción de poca amplitud, en lugar de enfrentar metas que requieren de un número considerable de acciones para ser realizadas y por lo tanto, requieren más “planificación”. Esto nos hace pensar que el impacto del número de módulos de habilidad que componen un agente en su eficiencia no es tan grave. Por otra parte, aún cuando la red de comportamiento tuviera cursos de acción muy amplios, es posible que el algoritmo seleccione cursos de acción alternativos ya que no espera convergencia en los niveles de activación y el umbral de activación disminuye con el paso del tiempo. Debemos señalar que en este último caso, el comportamiento del agente no tendería al óptimo.

El hecho de que las mismas reglas de activación/inhibición se apliquen por igual a todos los módulos de habilidad, que además, están conectados por ligas fijas representadas localmente, abre interesantes oportunidades de paralelizar el algoritmo, lo cual implicaría una mejora considerable en la velocidad de éste.

Un segundo factor a considerar en esta sección es el relacionado con la ausencia de variables. El algoritmo de redes de comportamiento no incorpora variables. En su lugar emplea una *representación directa* como la utilizada en Pengi [Agre y Chapman 90] presentada en el capítulo anterior. Una consecuencia de la ausencia de variables en el algoritmo es que todos los módulos de habilidad ó operadores deben estar instanciados. Así mismo, no es posible plantear metas del *estilo muevete-a(x,y)*. Aunque esto pudiera parecer una limitación sería, en el capítulo anterior presentamos como el robot Toto [Mataric 89] hace uso de mapas de navegación utilizando *representación directa* que posibilitan metas de tipo *muevete-a-la-puerta*. Es posible incorporar variables en el algoritmo creando copias de un módulo de habilidad cada vez que este es instanciado con valores de variables distintos, sin embargo esta opción requeriría una cantidad mucho mayor de cómputo. La tendencia es utilizar *representación directa* y replantearnos la forma en que un agente puede ser “instruido” sobre las metas a realizar.

Debemos mencionar que el algoritmo presenta algunas limitaciones. Esporádicamente, la selección de acción de un agente puede caer en ciclos. Esto se debe a que el sistema no mantiene una historia de las acciones que ha realizado en el pasado. El hecho de que los ciclos se presenten en contadas ocasiones se debe a la interacción del agente con su medio ambiente. En un medio ambiente *dinámico* es muy probable que las condiciones del medio cambien constantemente, lo cual tiene un impacto en la activación propagada a través de la red de comportamiento y en consecuencia la selección de acción podría salir del ciclo en que se encuentra. Una posible solución es incorporar un mecanismo de *habitación* con el cual cada módulo de habilidad que se activa es menos propenso a activarse en el futuro, esto pueden lograrse usando umbrales de activación locales que varían en el tiempo.

Un segundo problema está relacionado con la selección de parámetros. No es claro cómo podemos seleccionar un conjunto de parámetros dado una aplicación específica. Esta selección depende de las propiedades que deseamos exhiba la red de comportamiento, pero además depende de factores como el tamaño y la estructura de la red de comportamiento. Este problema y sus consecuencias en el estudio y enseñanza de las redes de

comportamiento nos sugirieron la necesidad de implementar una herramienta como ABC descrita en capítulo 5.

Capítulo 4

Implementación de un Algoritmo de Redes de Comportamiento

4.1 Presentación

En este capítulo presentaremos algunos aspectos relativos a la implementación del algoritmo de redes de comportamiento revisado en el capítulo anterior. Es posible obtener un listado completo de esta implementación en las fuentes citadas en la introducción de este trabajo. El objetivo de esta presentación es ofrecer al lector los elementos necesarios para operar nuestra implementación del algoritmo de redes de comportamiento. Con esta idea hemos incluido un primer ejemplo de agente basado en comportamiento que ilustra las propiedades del algoritmo que hemos discutido y la forma en que éste se configura a través de los parámetros globales.

Una primera implementación en LISP de este mecanismo fue realizada con la idea de verificar que el comportamiento del algoritmo fuera el reportado en la literatura. Ante los resultados promisorios, se decidió que la implementación final sería programada en tool command language (tcl) [Ousterhout 94] [Welch 95] [Johnson 96]. Esta decisión obedeció a varios factores:

- La extensión tk del tcl, permitiría el rápido desarrollo de una herramienta gráfica para el diseño y simulación de agentes que hicieran uso de la red de comportamiento.
- El lenguaje tcl permitiría la portabilidad de la herramienta a diversas plataformas computacionales (Sun OS, Solaris, Windows95, Windows 3.11 y MacOS) con muy poco esfuerzo.
- Portar la implementación del algoritmo de LISP a tcl no parecía una tarea difícil de realizar, dada las características de ambos lenguajes de programación y la naturaleza del algoritmo.

Se empleó la versión 7.5 de tcl y 4.1 de tk para la segunda implementación de este mecanismo de selección de acción. La programación se llevó a cabo en una máquina Sun SPARCstation 5 bajo el sistema operativo Sun OS y una vez que la implementación estaba

terminada, se realizaron pruebas en otras plataformas, más información al respecto puede encontrarse en la sección 4.5.2.

4.2 Definiendo un agente

Un módulo de habilidad se define sobre un arreglo asociativo de tcl, cuyo identificador es el nombre mismo del módulo. Este arreglo tiene cinco elementos: un *nombre* que identifica al módulo de habilidad, una lista *precond* que contiene las proposiciones que son precondiciones del agente (c_i en el modelo matemático), una lista *agrega* que contiene las proposiciones que se espera sean verdaderas después de la acción del módulo (a_i), una lista *borra* con aquellas proposiciones que se espera dejen de ser verdaderas después de activado el módulo (d_i) y un *alfa* que representa el nivel de activación del módulo (α_i).

El procedimiento `define_mc` (figura 4.1) recibe como parámetros los valores correspondientes a los elementos mencionados, construye el arreglo que representa el módulo de habilidad e inserta su nombre en una lista de todos los módulos de habilidad que conforman al agente que se está definiendo (A). Posteriormente actualiza el conjunto de proposiciones que definen al agente (P_A).

```
proc define_mc {nombre c a d} {
    global MC
    global $nombre
    global P

    set [join [list $nombre (nombre)] {}] $nombre
    set [join [list $nombre (precond)] {}] $c
    set [join [list $nombre (agrega)] {}] $a
    set [join [list $nombre (borra)] {}] $d
    set [join [list $nombre (alfa)] {}] 0

    set MC [linsert $MC 0 $nombre]
    set P [set_union $P [set_union $c [set_union $a $d]]]
}
```

Figura 4.1 El procedimiento `define_mc` permite definir un módulo de habilidad y agregarlo a la lista de módulos `MC` que componen al agente, además actualiza el conjunto de proposiciones `P` que se utiliza para definir al agente

Para ejemplificar el uso de este procedimiento recordemos el ejemplo del mundo de los cubos mencionado en la figura 3.1. La definición del módulo de habilidad `coloca_azul_en_rojo` sería la siguiente:

```
define_mc coloca_azul_on_rojo \
    {libre_rojo libre_azul} \
    {azul_en_rojo} \
    {libre_rojo}
```

La diagonal invertida al final de cada línea de la definición le indican a tcl que el procedimiento `define_mc` aún no ha terminado.

4.2.1 Creando la red de comportamiento

Una vez que se ha definido el conjunto de módulos de habilidad que definen al agente, es necesario establecer las ligas entre estos para conformar la red de comportamiento. El procedimiento `compila_red` (figura 4.2) se encarga de establecer las ligas de cada módulo que pertenece a la lista `MC`, extendiendo el arreglo asociativo de cada módulo de habilidad con tres nuevos componentes: `succ`, `pred` y `conf`.

```

proc compila_red {} {
  global MC
  foreach i $MC {
    global $i
    set [join [list $i (succ)] {}] [sucesor $i]
    set [join [list $i (pred)] {}] [predecesor $i]
    set [join [list $i (conf)] {}] [conflicto_con $i]
  }
}

```

Figura 4.2 El procedimiento `compila_red` establece la red de comportamiento al crear las ligas correspondientes entre los módulos de habilidad que conforman al agente.

Estas listas contienen listas que representan, respectivamente, los sucesores, predecesores y módulos conflictivos de cada módulo. La forma en que las ligas son representadas puede verse en la figura 4.3.

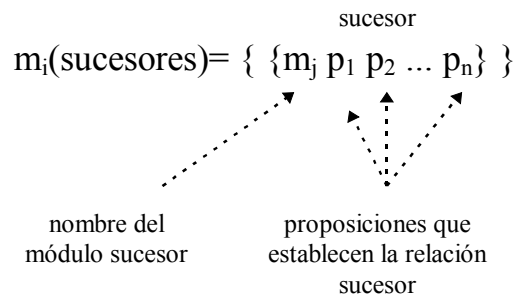


Figura 4.3 Las ligas entre módulos se definen como una lista de listas. En esta figura mostramos la estructura de estas listas. Cada lista miembro de $m_i(\text{sucesores})$ es un sucesor del módulo m_i . La lista de cada sucesor tiene como cabeza el nombre del módulo sucesor y el resto de la lista contiene las proposiciones que establecen esta relación.

Ilustraremos la forma en que estas listas se obtienen tomando como ejemplo el procedimiento `sucesor` (figura 4.4). Este procedimiento recibe como parámetro, el nombre de un módulo de habilidad x . Para todo módulo de habilidad i que pertenece a `MC`, se calcula la intersección $a_x \cap c_i$. Si la intersección es diferente del conjunto vacío, se crea

una lista cuya cabeza es el nombre del módulo i y su resto está constituido por los miembros de la intersección $a_x \cap c_i$ y se inserta en la lista de sucesores.

```

proc sucesor {x} {
  global MC
  global $x

  set succs {}
  foreach i $MC {
    global $i

    set aux [intersection [valor_campo $x agrega] \
                        [valor_campo $i precondition] ]
    if {[length $aux] > 0} {
      set succs [linsert $succs 0 \
                    [linsert $aux 0 \
                          [valor_campo $i nombre]]]
    }
  }
  return $succs
}

```

Figura 4.4 El procedimiento `sucesor` obtiene la lista de aquellos módulos que son sucesores del módulo x y las proposiciones que establecen esa relación.

4.3 Medio ambiente, metas y parámetros globales

Una vez que la red de comportamiento de un agente ha sido compilada, el agente puede ser puesto en funcionamiento. Para ello es necesario establecer un estado inicial del mundo y las metas que el agente debe cumplir. La variable `PV` representa las proposiciones que se observan verdaderas en el mundo. La variable `M` representa las metas del agente y la variable `ML` contiene las metas del agente que han sido realizadas.

Una de las limitaciones de esta primera implementación es el hecho de que el medio ambiente es modificado únicamente por las acciones del agente, de tal forma que no existe la posibilidad de que se presenten situaciones imprevistas. Esto podría solucionarse incorporando un procedimiento que provocara ruido en el ambiente, dentro del ciclo de ejecución del agente. De cualquier forma, el simulador que se describe en el siguiente capítulo permite que el usuario modifique el ambiente donde el agente actúa en tiempo de ejecución.

Los parámetros globales están representados por las variables: `PI`, `TETA`, `GAMMA`, `SIGMA` y `DELTA`. Como veremos más adelante, resulta conveniente crear archivos para la definición de agentes, que contengan el estado inicial del mundo, las metas del agente, los parámetros globales del sistema y, por supuesto, los módulos de habilidad que componen al agente.

4.4 El ciclo principal del agente

El ciclo de operación del agente (figura 4.5). es muy sencillo. Se calculan los niveles de activación de cada módulo de habilidad en ese tiempo y se normalizan esos valores. Se determina si hay un módulo ganador y de haberlo éste es activado, lo cual tiene efectos en el medio ambiente que son registrados inmediatamente. Se actualiza la información referente a las metas realizadas por el agente y las metas que aún debe realizar y finalmente se incrementa el tiempo.

```
while {1} {
  foreach i $MC {
    global $i
    set [join [list $i (alfa)] {}] \
      [nivel_activacion $i $tiempo]
  }
  normalizacion
  activa [ganon $tiempo] $tiempo
  set ML [set_union $ML [set_intersection $M $PV] ]
  set M [set_difference $M $ML]
  incr tiempo 1
}
```

Figura 4.5 Este es el ciclo principal del agente. Para ganar claridad, hemos omitido aquellos procedimientos dentro del ciclo que reportan información relevante para el usuario.

El procedimiento nivel_activacion (figura 4.6). calcula el nivel de activación de los módulos a cada tiempo del ciclo.

```
proc nivel_activacion {x tiempo} {
  global $x

  if { $tiempo == 0 } {return 0} \
  else {expr [alfa_t_1 $x] + \
    [entrada_por_estado $x $tiempo] + \
    [entrada_por_metas $x $tiempo] - \
    [salida_por_metas_protegidas $x $tiempo]+ \
    [fwp $x $tiempo] + [bwp $x $tiempo] - \
    [confp $x $tiempo]\
  }
}
```

Figura 4.6 El nivel de activación de los módulos al tiempo cero es cero, en cualquier otro caso se consideran los efectos de la dinámica de activación/inhibición para calcularlo.

Este procedimiento recibe como parámetros el nombre de un módulo de habilidad x y el número de veces que se ha repetido el ciclo de ejecución a partir de cero ($tiempo$). Si el tiempo es cero, el nivel de activación del módulo x es cero. En cualquier otro caso es necesario calcular su valor agregando al nivel de activación anterior la activación

proveniente del medio ambiente y de las metas, así como la propagación hacia adelante y hacia atrás; y restando la inhibición por metas protegidas y por conflictos.

Para ejemplificar como se obtienen los factores que intervienen en la determinación del nivel de activación de los módulos de habilidad, revisaremos el caso de la activación proveniente del medio ambiente. Este factor se obtiene mediante el procedimiento `entrada_por_estado` (figura 4.7)

```

proc entrada_por_estado {x tiempo} {
  global $x
  global SIGMA

  set props [set_intersection [sensar $tiempo]\
                             [valor_campo $x precondition]]

  set aux {}
  foreach i $props {
    set aux \
      [linsert $aux 0 \
               [expr $SIGMA * \
                    (1. / [llength [M $i]]) * \
                    (1. / [llength [valor_campo $x precondition]])]
      ]
  }
  sumatoria $aux
}

```

Figura 4. 7 Para calcular la activación que un módulo de habilidad x recibe por su correspondencia parcial con el medio ambiente, se obtienen las precondiciones del módulo x que son verdaderas y se realiza la sumatoria expresada en 3.3.8

Este procedimiento recibe como parámetros el nombre de un módulo de habilidad x y el tiempo en el ciclo de ejecución. El primer paso es obtener la lista de precondiciones de x que son verdaderas en ese tiempo. Ya que ha sido obtenida la lista `props`, sobre sus elementos se lleva a cabo la siguiente sumatoria:

$$\sum_{\forall p \in props} \sigma \frac{1}{|M(p)|} \frac{1}{|c_x|}$$

El resto de los factores que intervienen en la propagación de activación en la red pueden ser consultados en los listados completos del programa, disponibles en las fuentes citadas en la introducción.

4.5 Ejemplo 1. El robot de Charniak.

Para ilustrar el funcionamiento de este algoritmo de selección de acción, elegimos como un primer ejemplo la propuesta de Maes [Maes 90] de retomar un problema planteado

originalmente en el capítulo sobre planificación del libro de Introducción a la Inteligencia Artificial de Charniak [Charniak 85]. Este problema plantea la existencia de un robot de dos manos cuyas metas son lijar una tabla y pintarse a sí mismo. En su medio ambiente hay una mesa de trabajo, un pintura en spray, una lija y una tabla. Existe una restricción importante, en el momento en que el robot se pinta a si mismo deja de ser operacional y ya no puede realizar más acciones. Los parámetros para la selección de acción, el estado inicial del mundo y las metas del agente se muestran en la figura 4.8. La definición de los módulos de habilidad del agente se encuentran en la figura 4.9.

```

### Parámetros
set PI 20.0
set TETA 45.0
set SIGMA 20.0
set GAMMA 70.0
set DELTA 45.0

### Estado inicial del mundo
set PV {sprayersomewhere sandersomewhere boardsomewhere
handisempty handisempty operational}

### Metas del agente
set M {boardsanded selfpainted}

```

Figura 4.8 En un principio el robot se encuentra con una pintura en spray, una lija y una tabla a su disposición. Su percepción le indica además que tiene dos manos libres y que aún es operacional. Sus metas son lijar la tabla y pintarse a si mismo.

Como se puede observar, este agente está compuesto por diez módulos de habilidad, los cuales le permiten tomar los objetos de su medio ambiente, así como soltarlos, lijar la tabla, pintarse a sí mismo y ayudarse de una mesa de trabajo para lijar la tabla. Aunque este problema es muy sencillo, posee algunas propiedades interesantes como la presencia de metas y conductas conflictivas.

```

### Módulos de conductuales

define_mc pickupsprayer \
    {sprayersomewhere handisempty} \
    {sprayerinhand} \
    {sprayersomewhere handisempty}

define_mc pickupsander \
    {sandersomewhere handisempty} \
    {sanderinhand} \
    {sandersomewhere handisempty}

define_mc pickupboard \
    {boardsomewhere handisempty} \
    {boardinhand} \
    {boardsomewhere handisempty}

```

```

define_mc putdownsprayer \
    {sprayerinhand} \
    {sprayersomewhere handisempty} \
    {sprayerinhand}

define_mc putdownsander \
    {sanderinhand} \
    {sandersomewhere handisempty} \
    {sanderinhand}

define_mc putdownboard \
    {boardinhand} \
    {boardsomewhere handisempty} \
    {boardinhand}

define_mc putboardinvis \
    {boardinhand} \
    {boardinvis handisempty} \
    {boardinhand}

define_mc sandboardinhand \
    {operational boardinhand sanderinhand} \
    {boardsanded} \
    {}

define_mc sandboardinvis \
    {operational boardinvis sanderinhand} \
    {boardsanded} \
    {}

define_mc spraypaintself \
    {operational sprayerinhand} \
    {selfpainted} \
    {operational}

```

Figura 4.9 Módulos de habilidad para el robot de Charniak.

4.5.1 Un agente trabajando

A continuación analizaremos la selección de acciones llevada a cabo por el agente definido en esta sección. Debemos mencionar que los parámetros elegidos para esta simulación fueron seleccionados después de una serie de experimentos con esta versión del algoritmo. Utilizando el simulador descrito en el capítulo siguiente fué posible aumentar la velocidad con que el agente alcanza sus metas. Este y otros resultados al respecto se reportan en la sección de discusión y trabajo futuro.

Esta primer implementación del algoritmo incluye procedimientos para desplegar información relativa al estado del agente. Cada tiempo del ciclo de ejecución del agente produce una desplegado en la consola (la de tcl en Windows y MacOS o la del sistema en UNIX), con la siguiente información: tiempo actual, estado del mundo, metas pendientes, metas realizadas, umbral de activación y una tabla donde para todos los módulos de habilidad del agente se presentan tres elementos: Un “1” si el módulo es ejecutable ó un “0” si no lo es, el nombre del módulo y su nivel de activación. Cada vez que un módulo entra en actividad, el algoritmo lo reporta al final de esta tabla. Veamos el comportamiento de este agente:

```

Tiempo...: 0
Estado del mundo...: sprayersomewhere sandersomewhere
boardsomewhere handisempty handisempty operational
Metas del agente...: boardsanded selfpainted
Metas logradas...:
Umbral de activacion...: 45.0

0      spraypaintself  0.000000
0      sandboardinvis  0.000000
0      sandboardinhand 0.000000
0      putboardinvis  0.000000
0      putdownboard   0.000000
0      putdownsander  0.000000
0      putdownsprayer 0.000000
1      pickupboard    0.000000
1      pickupsander   0.000000
1      pickupsprayer  0.000000

```

Al tiempo 0, el estado del mundo y los parámetros son los especificado en la figura 4.8. Todos los módulos de habilidad tienen su nivel de activación en 0, aún no se considera el impacto del medio ambiente, ni el de las metas, tampoco hay dinámica de activación / inhibición entre los módulos. Los módulos de habilidad encargados de tomar objetos son ejecutables: pickupboard, pickupsander y pickupsprayer. No se han presentado módulos de habilidad en actividad.

```

Tiempo...: 1
Estado del mundo...: sprayersomewhere sandersomewhere
boardsomewhere handisempty handisempty operational
Metas del agente...: selfpainted boardsanded
Metas logradas...:
Umbral de activacion...: 40.5

0      spraypaintself  78.106400
0      sandboardinvis  39.644900
0      sandboardinhand 39.644900
0      putboardinvis  0.000000
0      putdownboard   0.000000
0      putdownsander  0.000000
0      putdownsprayer 0.000000
1      pickupboard    14.201100
1      pickupsander   14.201100
1      pickupsprayer  14.201100

```

El tiempo 1 es muy interesante. Observe que el umbral de activación se ha decrementado un 10% dado que ningún módulo de habilidad entró en actividad en el tiempo anterior. El medio ambiente y la metas continúan sin cambios. Sin embargo, hay cambios importantes en el nivel de activación de los módulos de habilidad. Los módulos de habilidad capaces de realizar alguna meta del agente han sido estimulados: spraypaintself tiene el nivel de activación más alto pues está relacionado con una sola meta selfpainted, mientras que sandboardinvis y sandboardinhand tienen que compartir la activación proveniente de la meta boardsanded. Los módulos de habilidad que están en correspondencia parcial

con el medio ambiente también son estimulados, aunque en menor proporción a aquellos que realizan metas ya que $\gamma > \sigma$. Los módulos `pickupboard`, `pickupsander` y `pickupsprayer` vieron incrementado su nivel de activación. Es importante señalar que la dinámica de inhibición / activación entre los módulos se da en un factor del nivel de activación al tiempo anterior. Debido a que el nivel de activación de todos los módulos de habilidad en el tiempo anterior era cero, los módulos que no tienen correspondencia con el medio ambiente, ni pueden realizar metas continúan con su nivel de activación en cero. Observe que el nivel de activación de `spraypaintself` es mayor al umbral, sin embargo no es un módulo ejecutable y por lo tanto no puede entrar en actividad.

```
Tiempo...: 2
Estado del mundo...: sprayersomewhere sandersomewhere
boardsomewhere handisempty handisempty operational
Metas del agente...: boardsanded selfpainted
Metas logradas...:
Umbral de activacion...: 36.45

0      spraypaintself  0.000000
0      sandboardinvis  41.683400
0      sandboardinhand 43.773300
0      putboardinvis  11.255300
0      putdownboard   1.044980
0      putdownsander  1.044980
0      putdownsprayer 1.044980
1      pickupboard    19.973700
1      pickupsander   40.394400
1      pickupsprayer  39.784800

ACTIVADO: pickupsander
```

Al tiempo 2 el mundo y las metas continúan sin cambios, el umbral de activación se ha reducido otro 10% y hay cambios muy interesantes en los módulos de habilidad. Las reglas de inhibición por conflicto han comenzado a trabajar: `spraypaintself` es inhibido por `sandboardinvis` y `sandboardinhand` reduciendo su nivel de activación a cero. Al mismo tiempo, comienza un proceso de diferenciación entre los dos módulos de habilidad que realizan la meta `boardsanded`: el módulo `sandboardinhand` pertenece a la cadena de módulos que con menos acciones puede llevar a cabo la meta `boardsanded` y por lo tanto es estimulado un poco más que `sandboardinvis`. El módulo `putboardinvis` es predecesor de `sandboardinvis` y por ello recibe mayor estimulación que los demás módulos que sueltan objetos (`putdown...`). El módulo `spraypaintself` tiene como predecesor a `pickupsprayer` por lo que este último recibe una cantidad considerable de estimulación ya que `spraypaintself` tenía un nivel de activación muy alto en el tiempo anterior. El módulo `pickupsander` es predecesor de `sandboardinvis` y `sandboardinhand` por lo que resulta ser el más beneficiado, rebasando el umbral y entrando en activación.

```
Tiempo...: 3
Estado del mundo...: boardsomewhere sanderinhand
sprayersomewhere handisempty operational
```

```
Metas del agente...: selfpainted boardsanded
Metas logradas...:
Umbral de activacion...: 45.0
```

```
0      spraypaintself  16.133800
0      sandboardinvis  43.976100
0      sandboardinhand 48.202400
0      putboardinvis  18.945200
0      putdownboard   2.113150
1      putdownsander  0.000000
0      putdownsprayer  3.647260
1      pickupboard    37.622700
0      pickupsander   0.000000
1      pickupsprayer  29.359600
```

La activación de un módulo de habilidad trae cambios en el medio ambiente y en las metas por lo que impacta fuertemente el flujo de activación en la red. En el tiempo 3 el estado del mundo reflejan los cambios ocurridos en el tiempo anterior: ha sido retirada la proposición `sandersomewhere` y se ha agregado `sanderinhand`. El nivel de activación ha vuelto a su nivel original. El nivel de activación del módulo de habilidad `spraypaintself` crece un poco, pues está relacionado con una meta que aún se debe realizar. La diferencia entre `sandboardinvis` y `sandboardinhand` se hace mayor, marcando una tendencia a que sea `sandboardinvis` y su cadena de predecesores quienes realicen la meta `boardsanded`. Puede observarse cierta urgencia por tomar la pintura de spray (`pickupsprayer`), pero el módulo que más estimulación recibe es `pickupboard` ya que de tomar la tabla el agente estará en condiciones de lijarla. Observe que el módulo encargado de soltar la lija (`putdownsander`) es ejecutable ahora, pero ha sido fuertemente inhibido.

```
Tiempo...: 4
Estado del mundo...: boardsomewhere sanderinhand
sprayersomewhere handisempty operational
Metas del agente...: boardsanded selfpainted
Metas logradas...:
Umbral de activacion...: 40.5
0      spraypaintself  17.730500
0      sandboardinvis  38.320500
0      sandboardinhand 45.199800
0      putboardinvis  21.272300
0      putdownboard   3.439650
1      putdownsander  0.000000
0      putdownsprayer  3.602110
1      pickupboard    55.223600
0      pickupsander   0.000000
1      pickupsprayer  15.211600
```

ACTIVADO: `pickupboard`

En el tiempo 4 el módulo de habilidad `pickupboard` entra en actividad gracias a la tendencia en los patrones de activación mencionada en el tiempo anterior. El módulo `sandboardinhand` se perfila como el próximo módulo en activarse. Los módulos `spraypaintself` mantiene un nivel de activación semejante al del tiempo anterior.

```

Tiempo...: 5
Estado del mundo...: sprayersomewhere sanderinhand boardinhand
operational
Metas del agente...: selfpainted boardsanded
Metas logradas...:
Umbral de activacion...: 45.0

```

```

0      spraypaintself  25.411500
0      sandboardinvis  57.009500
1      sandboardinhand 59.065500
1      putboardinvis   18.854100
1      putdownboard    0.000000
1      putdownsander   0.000000
0      putdownsprayer  3.742190
0      pickupboard     1.868460
0      pickupsander    1.200220
0      pickupsprayer   32.848300

```

```

ACTIVADO: sandboardinhand

```

En el tiempo 5 los cambios en el medio ambiente han sido registrados, el agente tiene la lija y la tabla en sus manos (sanderinhand y boardinhand). El módulo sandboardinhand es el de mayor nivel de activación y resulta activado. Al mismo tiempo pickupsprayer registra un incremento considerable en su nivel de activación. Observe que el nivel de activación del módulo sandboardinvis también se ha incrementado considerablemente (bastaría que el agente colocara la tabla en la mesa de trabajo para hacerlo ejecutable), lo cual ocasionará algunos problemas.

```

Tiempo...: 6
Estado del mundo...: boardsanded sprayersomewhere sanderinhand
boardinhand operational
Metas del agente...: selfpainted
Metas logradas...: boardsanded
Umbral de activacion...: 45.0

```

```

0      spraypaintself  43.198400
0      sandboardinvis  46.499400
1      sandboardinhand 4.637820
1      putboardinvis   40.707500
1      putdownboard    4.195110
1      putdownsander   0.000000
0      putdownsprayer  5.726660
0      pickupboard     2.236720
0      pickupsander    1.771840
0      pickupsprayer   51.026800

```

En el tiempo 6 se registra como una de las metas ha sido realizada (boardsanded), por lo que es retirada de la lista de metas pendientes del agente. El módulo de habilidad spraypaintself se incrementa considerablemente, lo mismo que pickupsprayer que no puede activarse pues el agente tiene ambas manos ocupadas. Un factor importante es que sandboardinhand sigue siendo más fuerte que spraypaintself y por lo tanto lo inhibirá en el próximo tiempo.

Como puede observarse, la velocidad del algoritmo ha sido aceptable (60%). Esto se ha logrado enfatizando la influencia de las metas en el comportamiento del agente. Sin embargo, lijar la tabla y pintarse a sí mismo constituyen metas conflictivas y era necesario considerar seriamente la inhibición por conflictos. Esto llevará a una reducción en la velocidad con que se realiza la segunda meta. En general, lo que se observa a partir de aquí es una competencia entre `spraypaintself` y `sandboardinvis` que aún después de realizada la meta `boardsanded`, mantiene un nivel de activación muy alto.

En la elección de parámetros para este ejemplo quisimos mantener cierta optimalidad en las acciones del agente. En especial, no deseábamos que el agente lijara dos veces la tabla (una vez tomándola con su mano y una segunda colocándola en la mesa de trabajo). Lo que observamos es que el agente parece considerar seriamente que debe soltar la tabla en la mesa una vez que el módulo `sandboardinvis` no pueda activarse. Una solución a este problema que encontramos utilizando el simulador ABC, descrito en el siguiente capítulo, fue bajar el valor de σ de tal forma que el medio ambiente tuviera menos impacto. Resumiremos los tiempos por motivos de espacio.

```
Tiempo...: 10
Estado del mundo...: boardsanded sprayersomewhere sanderinhand
boardinhand operational
Metas del agente...: selfpainted
Metas logradas...: boardsanded
Umbral de activacion...: 29.5245

0      spraypaintself  27.760500
0      sandboardinvis 24.665400
1      sandboardinhand 10.246100
1      putboardinvis  20.076500
1      putdownboard   8.269410
1      putdownsander  0.000000
0      putdownsprayer 15.822500
0      pickupboard    5.786160
0      pickupsander   3.462910
0      pickupsprayer  83.910300
```

Para el tiempo 10 el módulo de habilidad `spraypaintself` ha logrado un nivel de activación mayor a `sandboardinvis`. De los módulos de habilidad ejecutables el de mayor nivel de activación es `putboardinvis` que contribuye con una submeta necesaria para que `sandboardinvis` se active. Es por ello que su nivel de activación ha venido disminuyendo paulatinamente. Al mismo tiempo el módulo encargado de tomar el `sprayer` (`picksprayer`) tiene un nivel de activación muy alto y podría activarse en cuanto llegará a ser ejecutable.

```
Tiempo...: 16
Estado del mundo...: boardsanded sprayersomewhere sanderinhand
boardinhand operational
Metas del agente...: selfpainted
Metas logradas...: boardsanded
Umbral de activacion...: 15.6906

0      spraypaintself  27.506000
0      sandboardinvis 14.894300
```

```

1      sandboardinhand 10.238800
1      putboardinvis  16.184600
1      putdownboard   8.441600
1      putdownsander  4.467680
0      putdownsprayer 18.799700
0      pickupboard    6.658640
0      pickupsander   3.863630
0      pickupsprayer  88.945000

```

ACTIVADO: putboardinvis

Finalmente, después de una consideración muy seria, en el tiempo 16 el agente decide colocar la tabla en la mesa de trabajo.

Tiempo...: 17

```

Estado del mundo...: sanderinhand sprayersomewhere boardsanded
handisempty boardinvis operational
Metas del agente...: selfpainted
Metas logradas...: boardsanded
Umbral de activacion...: 45.0

```

```

0      spraypaintself 44.120900
1      sandboardinvis 16.105800
0      sandboardinhand 9.093680
0      putboardinvis  0.000000
0      putdownboard   7.290020
1      putdownsander  0.000000
0      putdownsprayer 19.512500
0      pickupboard    11.855400
0      pickupsander   0.000000
1      pickupsprayer  92.021800

```

ACTIVADO: pickupsprayer

Tiempo...: 18

```

Estado del mundo...: boardsanded sprayerinhand sanderinhand
boardinvis operational
Metas del agente...: selfpainted
Metas logradas...: boardsanded
Umbral de activacion...: 45.0

```

```

1      spraypaintself 51.925500
1      sandboardinvis 41.308100
0      sandboardinhand 20.547300
0      putboardinvis  2.249170
0      putdownboard   22.310100
1      putdownsander  0.000000
1      putdownsprayer 3.993200
0      pickupboard    44.975000
0      pickupsander   2.115350
0      pickupsprayer  10.576700

```

ACTIVADO: spraypaintself

Es importante señalar que mientras el agente decidía cuando soltar la tabla, estaba trabajando también en el plan destinado a pintarse así mismo. De hecho, una vez que el agente tiene una mano libre (tiempo 17), su acción inmediata es tomar el sprayer y pintarse (tiempo 18).

```
Tiempo...: 19
Estado del mundo...: boardinvisе sanderinhand sprayerinhand
boardsanded selfpainted

Metas del agente...:
Metas logradas...: selfpainted boardsanded...
```

Al tiempo 19 el agente ha cumplido sus metas y continua en su ciclo de operación ante la posibilidad de llevar a cabo nuevas metas. La velocidad del agente se redujo al 30%, pero su optimalidad es del 100%. Por su puesto que optimalidad y velocidad pueden variar eligiendo valores diferentes para los parámetros globales.

4.5.2 Portabilidad

Este ejemplo fue ejecutado en diferentes plataformas con la misma implementación del algoritmo. No fue necesario ningún cambio en el código y los resultados obtenidos fueron los mismos. La única variante se registró en el tiempo real en alcanzar la solución, siendo la plataforma más rápida una máquina Pentium a 75 Mhz con sistema operativo Windows 95. La información de las diferentes plataformas y el tiempo en alcanzar la solución pueden verse en la tabla 4.1

Modelo	Procesador	Vel. (MHz)	RAM (Mb)	S.O.	tiempo en realizar metas (seg)
Power Macintosh 7100/80	Power PC	80	8	Mac OS 7.0	38.61
Sun SPARC 5	RISC	70	16	Sun OS 4.1.3 u1	23.29
ACER P75	Pentium	75	16	Windows 95	20.93

Tabla 4.1. Información sobre las diferentes plataformas en que se realizó este experimento y el tiempo empleado por el agente en alcanzar todas sus metas.

Se utilizó la función *time* de tcl para obtener el tiempo en microsegundos, que la implementación del simulador corría el ciclo principal del agente (procedimiento run). Para todas las plataformas empleadas, el procedimiento fue el siguiente:

1. Cargar el intérprete tcl
2. Cargar el simulador de agentes: maes4.tcl
3. Cargar la descripción del agente a simular: ej5.tcl
4. Compilar la descripción del agente

5. Ejecutar la función *time* sobre la simulación del agente (función *run*)

Capítulo 5

ABC

5.1 Presentación

ABC es una herramienta para el diseño y simulación de agentes basados en redes de comportamiento. Un ambiente gráfico implementado en tcl/tk provee al usuario de un conjunto de herramientas para utilizar el algoritmo que hemos descrito e implementado en los dos capítulos anteriores.

Hemos mencionado que uno de los problemas al trabajar con redes de comportamiento está relacionado con la elección adecuada del conjunto de valores para los parámetros del sistema, que nos permitan obtener el comportamiento deseado del agente. ABC ofrece al usuario un conjunto de herramientas que facilitan la definición de agentes en los términos requeridos por las redes de comportamiento y observar su comportamiento para obtener información que pueda ser de utilidad en la elección del conjunto de valores mencionados.

Un usuario de ABC puede observar la competencia entre los módulos de habilidad durante la actividad del agente y acceder una gráfica de activación sobre el tiempo que ha resultado de gran utilidad en la configuración de la red de comportamiento. También tiene acceso a aspectos interesantes del comportamiento del agente como la secuencia de acciones que se han activado y la velocidad de decisión del agente.

ABC permite que el usuario modifique la percepción que el agente tiene del medio ambiente en tiempo de ejecución, de tal forma que podemos experimentar con ruido en la percepción y en las acciones del agente. Así mismo, podemos cambiar las metas que el agente debe realizar. De esta manera buscamos que el agente enfrente un medio ambiente dinámico en el que podemos incorporar los cambios que nos resulten de interés.

Las modificaciones necesarias en el diseño del agente y los parámetros del algoritmo pueden realizarse fácilmente desde ABC gracias a un módulo de diseño y el acceso a los parámetros desde la ventana de simulación.

Debemos mencionar que este capítulo no pretende ser un manual del usuario de ABC. Para acceder la documentación completa de este sistema debe remitirse a los lugares indicados en la introducción de este trabajo.

5.2 Ventana principal

La figura 5.1 muestra la primer ventana que un usuario de ABC accesa cuando utiliza nuestro sistema. Como puede observar, esta ventana se encuentra dividida en varias secciones. Las describiremos en orden descendente:

Medio ambiente. Podrá identificar esta división por su encabezado y un icono del globo terráqueo en ella. Se trata de una lista que contiene los elementos de percepción del agente, el conjunto de proposiciones que se observan verdaderas en el medio a un tiempo dado. Es posible realizar una pausa en la simulación del agente para modificar este conjunto de proposiciones. La forma de hacerlo es dando un *clic* en la lista para colocarnos en el lugar adecuado y editar el contenido de ésta.



Figura 5.1 Ventana principal de ABC.

Metas del agente. La lista a la derecha de un “tache” contiene las metas que el agente debe cumplir. La lista encabezada por una “palomita” contiene las metas que el agente ya realizó. Ambas listas pueden modificarse de igual forma que el medio ambiente. En esta implementación no consideramos metas permanentes, así que cuando una meta es realizada

desaparece de la lista de metas pendientes y pasa a la de metas realizadas. La razón de no considerar metas permanentes, se debe a que los agentes que nos habíamos planteado como ejemplo no harían uso de motivaciones. Modificar el algoritmo para tener metas permanentes y pesos asociados a ellas es muy sencillo, las dificultades aparecen al decidir, dado un agente, como es que esos pesos varían en el tiempo y ante las acciones que lleva a cabo el agente.

Area de simulación. Por debajo de la sección de metas del agente, aparece un área de texto con un encabezado de tres elementos: ejecutable, módulo y alfa. En esta área el usuario podrá observar el listado de los módulos de habilidad que componen al agente durante la simulación. La columna ejecutable indica cuando un módulo “Si” es ejecutable y cuando “No” lo es. La columna módulo corresponde a la lista de los nombres de los módulos de habilidad. La columna alfa contiene los niveles de activación de los módulos. El renglón correspondiente a un módulo ejecutable aparece en rojo, el de uno no ejecutable aparece en azul. Una barra de *scroll* permite movimiento vertical para consultar la información de los módulos que queden fuera del área de texto. Por debajo del área de texto mencionada, encontramos tres elementos de izquierda a derecha: un reloj con una entrada de texto que despliega el ciclo en que se encuentra la simulación, una entrada de texto precedida de una θ que contiene el umbral de activación, y una entrada de texto que despliega el nombre del módulo que esta activo.

Controles de la simulación. Por debajo del área de simulación y al lado izquierdo, encontramos los controles de la simulación representados por una serie de botones muy parecidos a lo de un reproductor de CDs. Sus funciones son, de izquierda a derecha: retroceder un paso en la simulación, ejecutar la simulación continuamente, avanzar un paso en la simulación, realizar una pausa en la simulación y detener completamente la simulación.

Parámetros. A la derecha de los controles de simulación encontramos la sección de parámetros globales de la red de comportamiento. En esta sección el usuario podrá establecer valores para los siguientes parámetros, de izquierda a derecha: π , θ , γ , σ , y δ .

Desde la ventana principal es posible acceder los demás componentes de ABC, para lo cual encontramos los botones: *archivos*, *activación/tiempo*, *diseño del agente*, *ayuda* y *salir*. Finalmente, por debajo de estos botones tenemos una barra de estado que nos despliega información sobre la actividad en la que se encuentra ABC, así como el nombre del archivo que estamos trabajando.

5.3 Ventana de diseño del agente

En esta ventana (figura 5.2), el usuario puede definir los módulos de habilidad que componen a un agente y crear las ligas entre estos, para formar la red de comportamiento. El lado izquierdo de esta ventana presenta una lista de las conductas o módulos de habilidad que han sido definidos previamente. Del lado derecho encontramos una serie de botones con las siguiente funciones, en orden descendente:

Nuevo. Permite definir módulos de habilidad. Su activación abre una ventana como la mostrada en la figura 5.3. Esta ventana de diseño de conductas me permite especificar el

nombre del módulo de habilidad que estamos definiendo, sus precondiciones y sus listas de efectos agregar y borrar. El lado izquierdo de la ventana esta ocupado por la lista de proposiciones que se han usado en la definición del agente hasta ese momento, es posible incluir cualquiera de estas proposiciones en una lista de las mencionadas, haciendo *doble clic* sobre ella. El botón *aceptar* agrega el módulo de habilidad definido a la composición del agente. El botón *borrar* todo limpia las todas las listas mencionadas. El botón *cancelar* cierra la ventada sin agregar el módulo a la composición del agente.



Figura 5.2 Ventana de Diseño del agente



Figura 5.3 Ventana de Diseño de los módulos de habilidad

Editar. Permite realizar cambios a un módulo de habilidad previamente definido. Para ello, se selecciona el módulo de habilidad en la lista de módulos y se hace un *clic* en el botón *Editar*. Esto abrirá una ventana idéntica a la creada por el botón *Nuevo*, pero las listas de ésta contendrán la información del módulo que nos interesa modificar. El botón *Aceptar*

abre una ventana de confirmación, ya que la nueva definición del módulo de habilidad destruiría a la anterior.

Borrar. Seleccionando un módulo de habilidad y haciendo *clic* en el botón *Borrar*, eliminamos ese módulo de habilidad de la definición del agente.

Ver. Seleccionando un módulo de habilidad y haciendo *clic* en el botón *Ver*, abre una ventana como la mostrada en la figura 5.4. Esta ventana despliega la información del módulo de habilidad seleccionado, incluyendo sus ligas con otros módulos y su nivel de activación actual. Las ligas se representan por listas, donde la cabeza de la lista representa el módulo de habilidad con quien se establece la relación y el resto las proposiciones que hacen posible que exista cada liga. Si el usuario no ha creado las ligas entre los módulos de habilidad, las listas en la sección ligas aparecen vacías.

The screenshot shows a window titled 'Conducta' with a dropdown menu on the left. The window is divided into two main sections: 'Datos generales' and 'Ligas'.
Under 'Datos generales':
- 'Nombre' is 'pickupsprayer' and 'Activacion' is '0'.
- 'Precond' is 'sprayersomewhere handisempty'.
- 'Agregar' is 'sprayerinhand'.
- 'Borrar' is 'sprayersomewhere handisempty'.
Under 'Ligas':
- 'Conflictos' is '{pickupboard handisempty} {pickupsander handisempty}'.
- 'Sucesores' is '{spraypaintself sprayerinhand} {putdownsprayer sprayerinhand}'.
- 'Predecesores' is '{putboardinvis handisempty} {putdownboard handisempty} {putdownsai...}'.
A 'Cerrar' button is located at the bottom right.

Figura 5. 4 Ventana para visualización de módulos de habilidad

Crear ligas. Este botón establece las ligas entre los módulos de habilidad que han sido definidos, creando así la red de comportamiento del agente. De cualquier forma, si las ligadas no se han establecido, los controles de avanzar un paso y simulación continua en la ventana principal, crean las ligas.

Cerrar. Este botón cierra la ventana de diseño del agente.

5.4 Ventana de activación/tiempo

La ventana de activación en el tiempo (figura 5.5) provee información sobre el comportamiento del agente que puede ser utilizada para la configuración de los parámetros en busca del comportamiento que deseamos sea exhibido por el agente. La parte superior de esta ventana es ocupada por una gráfica de activación en el tiempo. En el lado izquierdo de la gráfica encontramos la lista de los módulos de habilidad que componen al agente y al

final de ésta, la etiqueta *tiempo*. El nivel de activación de los módulos de habilidad se representa por un cuadro de color, que puede ir gradualmente del blanco al rojo. El color blanco representa a niveles de activación de cero y el rojo niveles de activación que están un 100% arriba del umbral de activación. Entre más alto sea el nivel de activación de un módulo, el cuadro desplegado tenderá al rojo. Un cuadro con una línea amarilla arriba y abajo de él indica que el módulo de habilidad es ejecutable en ese tiempo. La línea verde sobre los cuadros desplegados marca la secuencia de conductas que se han activado, las cuales están representadas por cuadros de color amarillo. El tiempo aparece desplegado en color verde cada 10 ciclos de simulación.

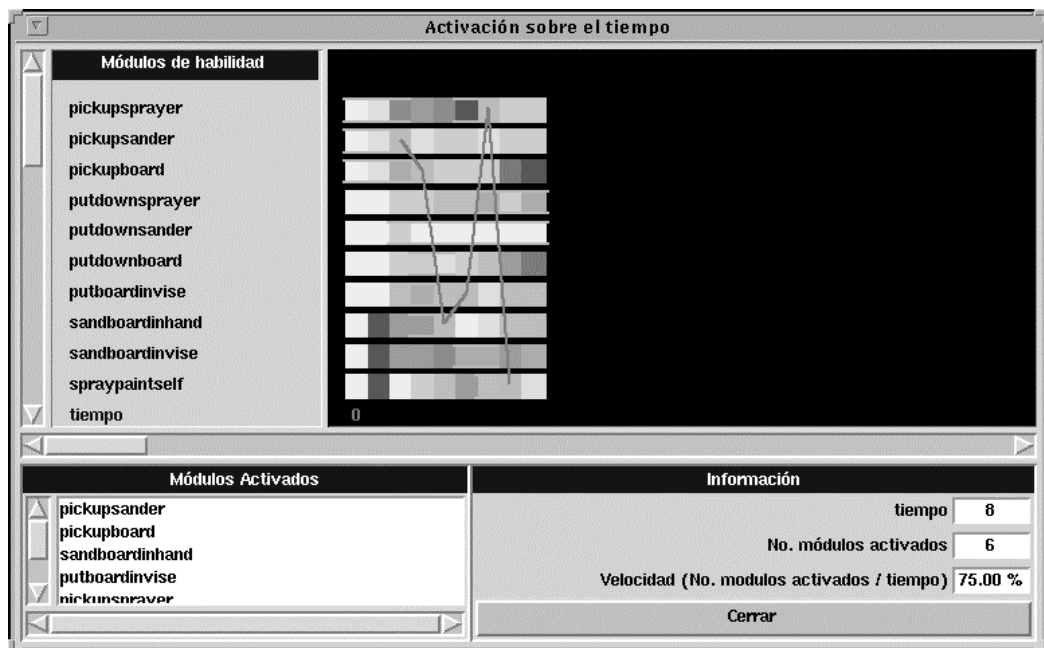


Figura 5.5 Gráfica de actividad en el tiempo

Por debajo de la gráfica de activación encontramos dos secciones. A la derecha aparece la lista de los módulos que se han activado durante la simulación. A la izquierda aparecen tres entradas de texto reportando el ciclo en que la simulación fue graficada, el número de módulos de habilidad que se han activado y la velocidad de decisión del agente (número de módulos activados entre el número de ciclos de la simulación).

5.5 Administración de archivos.

El botón de *archivos* en la ventana principal despliega una ventana con un conjunto de botones que me permite administrar los archivos con la definición de los agentes diseñados y simulados con ABC. La funcionalidad de estos botones es, en orden descendente:

Abrir. La definición de un agente realizada en la ventana de diseño del agente, así como las condiciones iniciales de percepción del medio ambiente, las metas y los parámetros con

que se efectúa una simulación pueden ser guardados en un archivo. Este botón abre una ventana como la mostrada en la figura 5.6, la cual permite explorar los directorios de una computadora para localizar el archivo de interés. Una vez localizado el archivo, este puede ser cargado para su utilización haciendo *doble clic* en él o un *clic* en el botón *Aceptar*.



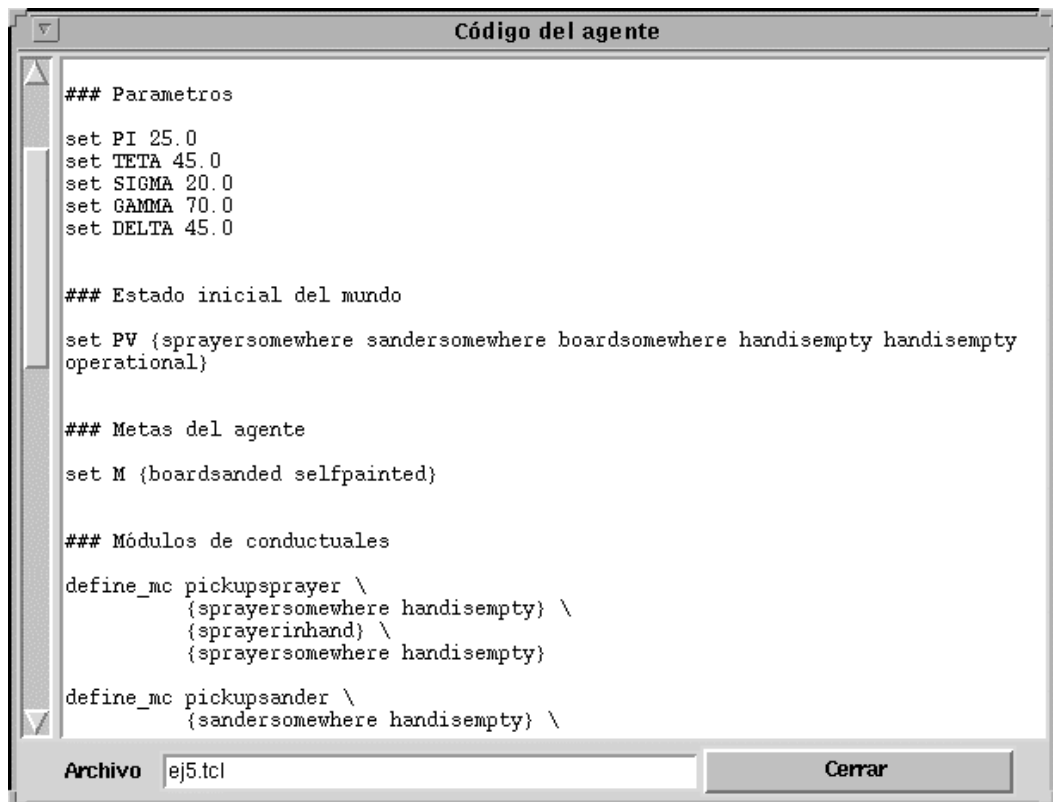
Figura 5. 6 Ventana de Selección de Archivos

Guardar. Este botón guarda en un archivo la definición del agente y la información relevante que aparece en la ventana principal. Si es la primera vez que la información del agente será archivada, este botón abre una ventana como la mostrada en la figura 5.6.

Guardar como... Este botón abre una ventana como la mostrada en la figura 5.6 para especificar el nombre con el que queremos un archivo sea guardado.

Ver. Este botón abre la ventana mostrada en la figura 5.7. El texto desplegado en esta ventana es el código que define al agente, sus metas y su medio ambiente, así como a los parámetros de la red de comportamiento. Esta es la información que será almacenada en un archivo con los botones *Guardar* y *Guardar como...*

Cerrar. Este botón *cierra* la ventada de Administración de archivos.



```
### Parametros
set PI 25.0
set TETA 45.0
set SIGMA 20.0
set GAMMA 70.0
set DELTA 45.0

### Estado inicial del mundo
set PV {sprayersomewhere sandersomewhere boardsomewhere handisempty handisempty
operational}

### Metas del agente
set M {boardsanded selfpainted}

### Módulos de conductuales
define_mc pickupsprayer \
    {sprayersomewhere handisempty} \
    {sprayerinhand} \
    {sprayersomewhere handisempty}
define_mc pickupsander \
    {sandersomewhere handisempty} \
```

Archivo

Figura 5.7 Ventana de código del agente

Capítulo 6

Incorporación de Agentes al medio ambiente.

6.1 Agentes y medio ambiente

Los agentes que hemos descrito en los capítulos anteriores, actúan en un medio ambiente simulado por las herramientas propuestas. Aunque es posible experimentar con situaciones propias de la interacción con un medio ambiente *dinámico*, como la incertidumbre en percepción y acción, y cambios imprevistos en el medio, no hemos abordado el problema relacionado con el diseño e implementación de actuadores y sensores. A partir del ejemplo de un agente situado en el mundo de los cubos, hemos implementado un agente con ABC. En este capítulo realizaremos la incorporación de este agente a un medio ambiente gráfico. Esta labor implica tres tareas que guardan una estrecha relación: el diseño de la red de comportamiento del agente, el diseño del medio ambiente y el diseño de sensores y actuadores del agente. La definición generada se muestra en la figura 6.1.

```
### ABC Agentes Basados en el Comportamiento
### Maestría en Inteligencia Artificial
### Universidad Veracruzana LANIA A.C.
### (c) 1996, Alejandro Guerra Hernandez
### Archivo...: /tmp_mnt/home/aguerra/agentes/Version2/ ej7b.tcl

### Parametros

set PI 10.0
set TETA 60.0
set SIGMA 20.0
set GAMMA 60.0
set DELTA 50.0

### Estado inicial del mundo
set PV {clear_a clear_b a_on_c}

### Metas del agente
set M {a_on_b b_on_c}
```

```

### Módulos de conductuales
define_mc stack_b_on_c \
  {clear_b clear_c} \
  {b_on_c} \
  {clear_c}
define_mc stack_b_on_a \
  {clear_a clear_b} \
  {b_on_a} \
  {clear_a}
define_mc stack_a_on_b \
  {clear_a clear_b} \
  {a_on_b} \
  {clear_b}
define_mc take_a_from_b \
  {clear_a a_on_b} \
  {clear_b} \
  {a_on_b}
define_mc take_a_from_c \
  {clear_a a_on_c} \
  {clear_c} \
  {a_on_c}
define_mc take_b_from_a \
  {clear_b b_on_a} \
  {clear_a} \
  {b_on_a}
define_mc take_b_from_c \
  {clear_b b_on_c} \
  {clear_c} \
  {b_on_c}
define_mc take_c_from_a \
  {clear_c c_on_a} \
  {clear_a} \
  {c_on_a}
define_mc take_c_from_b \
  {clear_c c_on_b} \
  {clear_b} \
  {c_on_b}
define_mc stack_a_on_c \
  {clear_a clear_c} \
  {a_on_c} \
  {clear_c}
define_mc stack_c_on_b \
  {clear_b clear_c} \
  {c_on_b} \
  {clear_b}
define_mc stack_c_on_a \
  {clear_a clear_c} \
  {c_on_a} \
  {clear_a}

```

Figura 6.1 Archivo de definición para el agente del mundo de cubos generado con la herramienta ABC.

Así que las tareas a realizar son la implementación de un medio ambiente para situar al agente y el diseño de sensores y actuadores de éste.

6.2 Implementación del medio ambiente

Decidimos construir un ambiente gráfico del mundo de los cubos que facilitará la manipulación del medio ambiente y la experimentación con diversos valores para los parámetros globales que ajustan la selección de acción. Después de una primera versión, decidimos que el ambiente gráfico debía incorporar facilidades para experimentar la forma en que la red de comportamiento enfrenta situaciones de imprecisión en su percepción y en sus acciones, así como cambios imprevistos en las metas que debe realizar y en el medio ambiente. La implementación de este ambiente se realizó en tcl/tk en una Sun SPARCstation 5. El resultado puede verse en la figura 6.2.

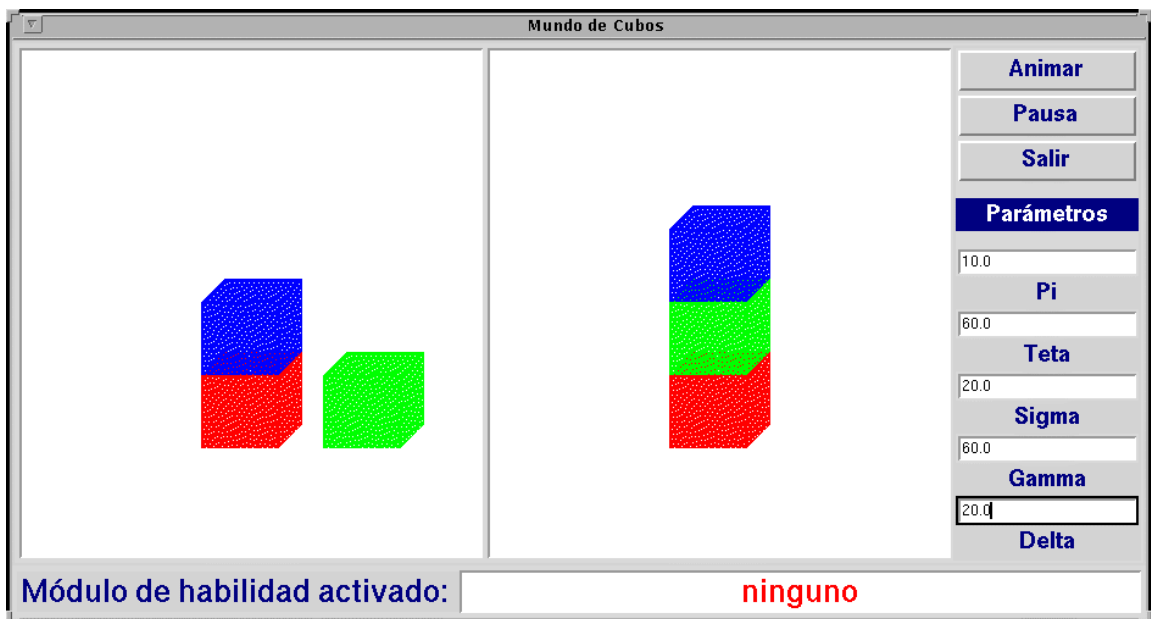


Figura 6.2 El mundo de los cubos, un ambiente gráfico para la experimentación sobre aspectos de percepción y actuación de los agentes basados en redes de comportamiento.

La sección de bloques a la izquierda representa el estado actual del medio ambiente donde actúa el agente. La sección derecha representa la configuración que deseamos construya el agente. Del lado superior derecho tenemos tres botones: animar, inicia los trabajos del agente; pausa, interrumpe temporalmente los trabajos del agente para modificar sus metas, medio ambiente e inclusive los parámetros con que se lleva a cabo la selección de acción; y salir, termina la ejecución de este simulador. Por debajo de estos botones, se encuentra la sección de parámetros que permite configurar el comportamiento de selección de acción exhibido por el agente. En la base de la ventana se despliega el nombre del módulo de habilidad que ha entrado en actividad. El código completo en tcl/tk correspondiente a la implementación de esta interface gráfica puede encontrarse en las fuentes citadas en la introducción de este trabajo.

6.3 Percepción

La percepción del agente le permite obtener información sobre el estado del mundo y las metas que debe realizar. Recordemos que la función de percepción del agente se compone de una serie de sensores virtuales que, con base en sensores reales, determinan si una proposición asociada a ellos es verdadera o no. Los sensores virtuales que utilizaremos, permiten que el agente establezca la relación existente entre los cubos del medio ambiente y son seis: tres de ellos determinan si los cubos están libres y los otros tres sobre quién está colocado cada cubo. El sensor real puede consultarse el código completo del agente. La función de percepción se define como la unión de estos sensores virtuales (figura 6.3).

```
proc percepcion {} {
    global configuration
    set aux {}
    foreach i \
        [list [clear $configuration rojo] \
              [clear $configuration verde] \
              [clear $configuration azul] \
              [on $configuration rojo] \
              [on $configuration azul] \
              [on $configuration verde]
        ] { if [expr [string compare $i {}] != 0] \
            {set aux [linsert $aux 0 $y]} }

    return $aux
}
```

Figura 6.3 Función de percepción del agente. Los sensores virtuales son los procedimientos: `clear $configuration cubo` y `on $configuration cubo`.

```
proc percepcion_meta {} {
    global configurationfinal
    set aux {}
    foreach i \
        [list [clear $configurationfinal rojo] \
              [clear $configurationfinal verde] \
              [clear $configurationfinal azul] \
              [on $configurationfinal rojo] \
              [on $configurationfinal azul] \
              [on $configurationfinal verde]] \
        {if [expr [string compare $i {}] != 0] \
            {set aux [linsert $aux 0 $i]} }

    return $aux
}
```

Figura 6.4 Esta función de percepción actúa sobre la sección de cubos donde se establecen las metas. Observe que emplea los mismos sensores virtuales que la función percepción.

Estos mismos sensores virtuales son utilizados para que el agente perciba la sección de cubos donde se le plantean las metas. Esto es deseable porque el usuario puede cambiar las

metas del agente en tiempo de ejecución. Esta función de percepción es muy semejante a la anterior.

6.3.1 Sensor virtual clear

Este sensor determina si un cubo está libre o no. Para ello observa la posición de los cubos en su sección. El hecho de que medio ambiente contenga únicamente tres cubos facilita estas operaciones, aún así ejemplifican el diseño de sensores virtuales y su incorporación al agente. El sensor recibe como parámetro el nombre de un cubo y determina si se encuentra libre o no. De estar libre el sensor reporta su proposición asociada: *clear_color-de-cubo*, en otro caso regresa una cadena vacía.

```
proc clear {configuracion cube} {
    set position [lsearch -exact $configuracion $cube]
    if { $position < 3 } {
        return "clear_$cube"
    } else {
        set up [ lindex $configuracion [ expr $position -3 ] ]
        if { $up == "0" } {
            return "clear_$cube"
        } else {
            return ""
        }
    }
}
```

Figura 6.5 Sensor Virtual clear. El parámetro configuración determina en que sección de cubos actúa el sensor.

6.3.2 Sensor virtual on

Este sensor recibe como parámetro un cubo y determina sobre quién está colocado. Si el cubo está sobre la mesa, el sensor regresa la cadena vacía. En caso de estar colocado sobre otro cubo, regresa su proposición asociada *color-de-cubo_on_cubo-abajo*.

```
proc on {configuracion cube} {
    set position [lsearch -exact $configuracion $cube]
    if { $position > 5 } { return "" } else {
        set below [ lindex $configuracion [ expr $position + 3 ] ]
        if { $below == "0" } {
            return "" } else {
            return [format "%s_on_%s" $cube $below]
        }
    }
}
```

Figura 6.6 Sensor virtual on.

6.4 Efectores

Los efectores son permiten al agente modificar su medio ambiente a través de sus acciones. En agentes robóticos están constituidos por partes físicas como brazos, ruedas, patas. En nuestro caso, los efectores pueden verse como procedimientos computacionales que realizan modificaciones en el medio ambiente donde está situado el agente. Hemos implementado dos actuadores para nuestro agente: uno de ellos se encarga de quitar un cubo que está sobre otro y colocarlo sobre la mesa; el otro se encarga de colocar un cubo sobre otro.

6.4.1 Efector take

El efector take recibe como parámetro el nombre de un cubo y su función es quitarlo de encima de algún cubo y ponerlo sobre la mesa. Observe que el efector no verifica si realmente el cubo está sobre algún colocado sobre algún otro cubo, o si hay espacio en la mesa para colocarlo. La información relativa a estas restricciones es capturada por los sensores y la decisión de cuando es adecuado activar el efector take se da en el algoritmo de selección de acción.

```
proc take { cube } {  
  
  global configuration  
  
  set position [.frame1.frame6.canvas6 coords $cube]  
  set originX [lindex $position 0]  
  set originY [lindex $position 1]  
  set movX [ expr (150 - $originX ) / 10]  
  set movY [ expr (80 - $originY ) / 10]  
  set i 0  
  
  while { $i < 10 } {  
    .frame1.frame6.canvas6 move $cube $movX $movY  
    set y 1  
    while { $y < 100 } {  
      incr y  
    }  
    incr i  
    update  
  }  
  
  set i 6  
  while { $i < 9 } {  
    set destination [lindex $configuration $i]  
    set final 0  
  
    if { $destination == "0" } {  
      set final $i  
      set i 9  
    } else {  
      incr i  
    }  
  }  
  
  if { $final == 6 } {
```

```

        set destinationX 50
        set destinationY 250
    } elseif { $final == 7 } {
        set destinationX 150
        set destinationY 250
    } else {
        set destinationX 250
        set destinationY 250
    }

    set movX [ expr ( $destinationX - 150 ) / 10 ]
    set movY [ expr ( $destinationY - 80 ) / 10 ]
    set i 0

    while { $i < 10 } {
        .frame1.frame6.canvas6 move $cube $movX $movY
        set y 1
        while { $y < 100 } {
            incr y
        }
        incr i
        update
    }
    PositionChange $cube $final

    puts $configuration
}

```

Figura 6.7 El efector take se encarga de quitar un cubo de encima de otro y colocarlo sobre la mesa.

6.4.2 Efecto stack

El efector stack coloca un cubo sobre otro. Al igual que el efector anterior, stack no verifica que el cubo destino esté libre, ni que el cubo que deseamos mover lo este. Su tarea es implementar el movimiento indicado.

```

proc stack { cube cube_in } {
    global configuration

    set final [expr [ lsearch -exact $configuration $cube_in ] -
3 ]

    set position [ .frame1.frame6.canvas6 coords $cube ]
    set originX [lindex $position 0]
    set originY [lindex $position 1]

    set movX [ expr ( 150 - $originX ) / 10 ]
    set movY [ expr ( 80 - $originY ) / 10 ]
    set i 0

    while { $i < 10 } {
        .frame1.frame6.canvas6 move $cube $movX $movY
        set y 1
        while { $y < 100 } {
            incr y
        }
    }
}

```

```

    }
    incr i
    update
  }

  if { $final == 0 } {
    set destinationX 50
    set destinationY 130
  } elseif { $final == 1 } {
    set destinationX 150
    set destinationY 130
  } elseif { $final == 2 } {
    set destinationX 250
    set destinationY 130
  } elseif { $final == 3 } {
    set destinationX 50
    set destinationY 190
  } elseif { $final == 4 } {
    set destinationX 150
    set destinationY 190
  } elseif { $final == 5 } {
    set destinationX 250
    set destinationY 190
  } elseif { $final == 6 } {
    set destinationX 50
    set destinationY 250
  } elseif { $final == 7 } {
    set destinationX 150
    set destinationY 250
  } else {
    set destinationX 250
    set destinationY 250
  }

  set movX [ expr ( $destinationX - 150 ) / 10 ]
  set movY [ expr ( $destinationY - 80 ) / 10 ]
  set i 0

  while { $i < 10 } {
    .frame1.frame6.canvas6 move $cube $movX $movY
    set y 1
    while { $y < 100 } {
      incr y
    }
    incr i
    update
  }

  PositionChange $cube $final
}

```

Figura 6.8 El efector stack coloca un cubo sobre otro.

6.5 Modificaciones en el algoritmo de selección de acción

La implementación original del algoritmo debió ser modificada en orden de incorporar los efectores y sensores que emplea el agente. Los cambios realizados son muy sencillos y se encuentran localizados principalmente en el ciclo principal de ejecución del agente. Estos cambios obedecen a la incorporación de los sensores del agente. Las variables PV y M asociadas a la percepción del medio ambiente y de las metas respectivamente, están ahora instanciadas con la unión de los sensores virtuales del agente mediante los procedimientos percepción y percepción meta. Decidimos detener la simulación cuando el agente alcanza sus metas, por ello el ciclo se repite mientras existan diferencias entre la percepción del medio ambiente y las metas logradas. La nueva definición del ciclo principal se muestra en la figura 6.9.

```
while {[llength [set_difference $PV $ML]] != 0} {
    update
    if {$Pausa} {set accion Pausa; update; continue}
    set PV [percepcion]
    set M [percepcion_meta]
    foreach i $MC {
        global $i
        set [join [list $i (alfa)] {}] [nivel_activacion $i
$tiempo]
    }
    normalizacion
    activa [ganon $tiempo] $tiempo
    set ML [set_intersection $M $PV]
    incr tiempo 1
    actualiza_historia
}
```

Figura 6.9 Una nueva definición para el ciclo principal del agente incorpora la percepción del medio ambiente.

El procedimiento activa ha sido reformado para que evalúe el script almacenado en la lista efector de los módulos de habilidad. En la versión original del algoritmo, activa enviaba un mensaje a la interfase gráfica ABC para indicar que módulo de actividad había entrado en actividad. La figura 6.10 muestra la nueva definición de la función activa. La última línea incluye un eval del efector del módulo de habilidad que entre en actividad.

```
proc activa {modulo tiempo} {
    global TETA
    global PV
    global umbral_aux accion
    global $modulo

    if { [llength $modulo] == 0 } {
        set TETA [expr $TETA - ($TETA * .1)]
    } else {
        set TETA $umbral_aux
        set accion [valor_campo $modulo nombre]; update
        set [join [list $modulo (alfa)] {}] 0
    }
}
```

```
    eval [valor_campo $modulo efector]  
  } }
```

Figura 6.10 Una nueva definición de activa para ejecutar el script efector de los módulos de habilidad.

6.6 Comportamiento del mundo de cubos

El agente situado en el mundo de cubos es capaz de resolver diversas configuraciones a petición del usuario. Problemas complejos como que se muestra en la figura 6.2 y pasar el bloque en la base de una torre a su cima han sido resueltos satisfactoriamente.

El usuario puede usar el botón de pausa para modificar la posición de los cubos, tanto en la sección de estado actual, como en la de metas. El agente resuelto satisfactoriamente los casos en que se le presentan estos cambios imprevistos en su medio ambiente.

Observe que la definición del agente sólo considera sobre quien está colocado cada cubo. En consecuencia los cubos pueden aparecer en desorden con respecto al eje horizontal. Este problema puede resolverse incrementado el número de efectores y sensores, de tal forma que consideren esta información.

Consideramos que este mundo de cubos puede ser utilizado para enseñar a una persona los problemas relacionados con el ajuste de parámetros. Por ejemplo, un valor alto en el parámetro δ lleva al agente a situaciones donde no puede seleccionar otra acción pues una meta parcial ha sido alcanzada y la configuración de parámetros no le permite deshacer metas realizadas; un valor alto de π ó uno bajo en δ lo lleva a realizar las metas parciales inmediatas sin considerar el total de las metas a realizar.

Discusión y Trabajo Futuro

1 *Discusión.*

A lo largo de este trabajo hemos presentado un mecanismo de selección de acción descentralizado y dinámicamente reconfigurable, con el cual un agente autónomo que opera en medios ambientes dinámicos puede elegir las acciones necesarias para llevar a cabo múltiples metas que pueden variar en el tiempo, conteniendo con cambios imprevistos en el medio ambiente y en las metas que debe realizar, así como eventuales fallas en sus sensores y efectores.

Hemos señalado que las redes de comportamiento poseen propiedades interesantes como comportamiento orientado a metas, relevancia del medio ambiente, adaptabilidad, persistencia en cursos de acción, prevención de conflictos, previsión y rapidez; y cómo estas propiedades pueden configurarse variando el conjunto de valores de los parámetros globales. Dentro de las propiedades señaladas en este trabajo, nos parece de central importancia el hecho de que las redes de comportamiento exhiban cierta capacidad de planificación como un resultado emergente de la dinámica de activación / inhibición entre los módulos de habilidad que conforman al agente y por lo tanto, constituyan una posible solución a los problemas que enfrentan los métodos de planificación tradicional al ser utilizados en ambientes dinámicos.

Creemos que las dificultades presentes en la elección de parámetros constituyen uno de los mayores problemas para la aplicación de las redes de comportamiento en donde queremos garantizar que la conducta del agente sea la esperada. En este contexto quisiéramos presentar brevemente, algunos ejemplos del uso del simulador ABC para encontrar un mejor conjunto de valores para los parámetros de la red de comportamiento, que resulten en comportamientos más adecuados a nuestras preferencias. Presentaremos dos experimentos para el problema del robot de Charniak descrito en la sección 4.5, en donde se logró incrementar la velocidad con que el agente realizaba sus metas y se ganó tolerancia a fallas en el agente.

Recordemos que el robot de Charniak tiene dos metas: lijar una tabla y pintarse a si mismo. En la simulación de la sección 4.5 encontramos que la primer meta se realiza con una velocidad aceptable (66%) en el tiempo seis de la simulación, pero que una vez resuelta esta meta el agente no realiza otra acción hasta diez tiempos más tarde y de forma continua

produce las acciones necesarias para realizar la segunda meta. Con la primera versión de nuestro algoritmo, se intentaron algunas modificaciones sin resultados satisfactorios, uno de ellos consistió en elevar el valor de γ y de δ buscando que las metas tuvieran un mayor peso en las decisiones del agente y que los cursos de acción que estaban siendo beneficiados en la propagación de activación inhibieran otros cursos posibles. El resultado de estos cambios fue un incremento en la velocidad a costa de pérdida de la optimalidad, esto es, el agente lijaba dos veces la tabla, lo cual queríamos evitar.

Empleando ABC y en especial, la gráfica de activación sobre el tiempo fue posible observar que los niveles de activación de los módulos de habilidad `sandboardinvis` y `sandboardinhand` eran muy parecidos, lo cual ocasionaba que aún cuando la meta `sandboard` hubiera sido realizada por el módulo de habilidad `sandboardinhand`, `sandboardinvis` mantuviera niveles de activación muy altos. En la definición original del agente este problema se resolvía estableciendo el umbral de activación lo suficientemente alto para que `spraypaintself` inhibiera a `sandboardinvis`, con la consecuente pérdida de velocidad. La gráfica de activación nos indicó que `sandboardinvis` incrementaba considerablemente su nivel de activación cuando la tabla era colocada en la mesa de carpintería por `putboardinvis`, debido a la influencia del medio ambiente. Esto nos llevó a reducir σ obteniendo la misma secuencia de módulos de habilidad activados con una velocidad del 85% (seis módulos activados en siete pasos de simulación).

En un segundo experimento con el robot de Charniak aprovechamos las facilidades de ABC para realizar modificaciones en la percepción del agente, para introducir ruido en las acciones de éste. Este experimento busca simular la falla en efectores de un agente, por ejemplo, el hecho de que la lija no halla sido tomada correctamente por la mano del agente. El primer módulo de habilidad que este agente activa es `pickupsander`, lo cual debe resultar en que el agente tiene la lija en una de sus manos. Si realizamos una pausa en la simulación inmediatamente después que el agente ha tomado la lija y cambiamos las condiciones perceptuales de éste de tal forma que la lija no aparezca en su mano y tenga dos manos libres, observaremos que en el siguiente tiempo el agente vuelve a tomar la lija. Otro ejemplo de tolerancia a fallas observado en este experimento es la robustez del agente ante la falla en una de sus manos. Desde un principio modificamos la percepción del agente para que únicamente una de sus manos aparezca como libre y bajo esta limitante el agente es capaz de llevar a cabo sus dos metas, con mayor lentitud claro está.

En los experimentos con ABC para ajustar los parámetros de la red de comportamiento hemos encontrado dificultades para incrementar la velocidad del agente que opera en el mundo de los cubos descrito en la sección 6.1. Los intentos realizados hasta ahora han resultado en un comportamiento menos coherente que el resultado original en donde la coherencia del agente se obtiene manteniendo niveles de activación muy altos. Parece ser que mantener las metas del agente como permanente nos ayudaría a mantener coherencia y obtener cierto grado de libertad para buscar mayor velocidad.

Los resultados obtenidos son muy prometedores y nos hacen pensar que ABC es una herramienta de utilidad en la enseñanza de sistemas basados en el comportamiento, en donde la experimentación es central para un buen diseño de los agentes que queremos construir. Así mismo, el capítulo 6 ejemplifica las bondades de ABC en el diseño de

agentes que actuaran en medios ambientes reales, a partir de su simulación. La seguridad de que el comportamiento del agente ha sido el esperado en la simulación y el hecho de conocer las condiciones perceptuales que necesitamos para implementar estos agentes, sin duda alguna es de gran ayuda en el proceso de implementación de agentes basados en redes de comportamiento.

2 Trabajo futuro

2.1 Proyecto monots

El proyecto Monots estudia la conducta de forrajeo del mono aullador (*alloutta palliata*) desde la perspectiva de generación de modelos que ofrezcan un marco teórico posible para explicar las observaciones experimentales. Se han propuesto diversas soluciones a este problema entre las que encontramos modelos de control óptimo (*optimal foraging theory*). El Dr. Manuel Martínez ha sugerido que estos modelos hacen excesivas suposiciones para describir conductas simples e introducen explicaciones teleológicas innecesarias, por lo que la aplicación de las redes de comportamiento en este problema podrían ser de gran utilidad. El ha propuesto un modelo de forrajeo basado en una simplificación de las redes de comportamiento [García et.al. 96], pretendemos que nuestro simulador ABC ayude a enriquecer este modelo y a su aplicación en la conducta de cortejo del mismo mono.

Una de las razones por las que la aplicación de las redes de comportamiento a este problema parece promisoria es que este mecanismo de selección de acción tienen una clara influencia etológica [Hendriks-Jansen 96]. Las redes de comportamiento requieren módulos de habilidad bien definidos, al igual que los patrones de acción fijos (FAP) propuestos por Lorenz. También requieren de condiciones perceptuales bien definidas que recuerdan la noción de estímulo señal (*sign stimulus*) usado en etología para activar las FAPs. A diferencia de los modelos etológicos, las redes de comportamiento proveen más de una condición perceptual por módulo de habilidad y todas estas deben ser verdaderas para que el módulo pueda activarse. Al igual que en las redes de comportamiento, en los modelos propuestos por Lorenz las conexiones entre conductas consumatorias y sus motivaciones son fijas.

Una primera etapa de este proyecto sería realizar las extensiones necesarias al simulador ABC para que pueda contender con el problema del mono aullador. Estas extensiones comprenden:

1. Modificar el esquema de inhibición por conflictos, de tal manera que sea posible exhibir comportamientos desplazados.
2. Modificar el esquema de metas, de tal forma que sea posible manejar motivaciones y los pesos asociados a estas.
3. Definir la red de comportamiento que genere la conducta de forrajeo en el mono aullador. Este paso incluye establecer la relación entre las motivaciones y la conducta de forrajeo en el mono.

Esta primera etapa es propuesta a partir de nuestra experiencia con una primer red de comportamiento cuyo objetivo es exhibir la conducta de forrajeo (figura 1). Los resultados

obtenidos con esta red nos indican que la omisión de motivaciones y el hecho de mantener factores constante en el medio ambiente sobre simplifican el problema y restan valor a la competencia entre los módulos de habilidad. El comportamiento de este agente exhibe un patrón interesante en donde el mono se alimenta, descansa y luego se mueve de forma coherente, sin embargo la simulación entra en un ciclo ante la ausencia de factores que obliguen al mono a contender con cambios en su medio ambiente.

```
### ABC Agentes Basados en el Comportamiento
### Maestria en Inteligencia Artificial
### Universidad Veracruzana LANIA A.C.
### (c) 1996, Alejandro Guerra Hernandez

### Archivo...: /home/aguerra/agentes-prog/ej6.tcl

### Parametros
set PI 20.0
set TETA 45.0
set SIGMA 20.0
set GAMMA 70.0
set DELTA 50.0

### Estado inicial del mundo
set PV {descansado hambriento manos_libres
comida_disponible}

### Metas del agente
set M {alimentacion descanso locomocion}

### Módulos de conductuales
define_mc descansar \
    {cansado satisfecho manos_libres} \
    {descanso descansado hambriento} \
    {cansado satisfecho}

define_mc moverse \
    {descansado satisfecho manos_libres} \
    {locomocion cansado hambriento} \
    {descansado satisfecho}

define_mc alimentarse \
    {manos_con_comida hambriento} \
    {alimentacion satisfecho} \
    {hambriento}

define_mc soltar_comida \
    {manos_con_comida } \
    {manos_libres} \
    {manos_con_comida}

define_mc levantar_comida \
    {manos_libres comida_disponible} \
    {manos_con_comida} \
```

{manos_libres}

Figura 1 Red de comportamiento para la conducta de forrajeo del mono aullador (allouata palliata).

Si la primer etapa arroja resultados positivos, como un comportamiento sensible a las motivaciones, procederemos a extender ABC para incorporar cambios imprevistos en el medio ambiente e iniciaremos el diseño de una red de comportamiento para la conducta de cortejo en el mono aullador.

Referencias

- [Agre y Chapman 90] AGRE P., y David Chapman, "What are plans for" En: Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, P. Maes editor, Ed. MIT Press/Bradford Books, Estados Unidos, 1990
- [Bates 94] BATES, Joseph, "The Role of Emotions in Belivable Agents" En: Communications of the ACM, Doug Riecken editor, Vol. 37, No. 7, Estados Unidos 1994
- [Blumberg 94] BLUMBERG B., "Action Selection in Hamsterdam: Lessons from Ethology", en: Proceedings of the Third International Conference on the Simulation of Adaptative Behavior, Meyer J.A. y Wilson S.W. editores, Ed. MIT Press/Bradford Books, Estados Unidos, 1994
- [Beer 90] BEER, Randall D., et.al. "A Biological Perspective on Autonomous Agent Design" En: Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, P. Maes editor, Ed. MIT Press/Bradford Books, Estados Unidos, 1990
- [Brooks 86] BROOKS R. A., "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, RA-2, April 1986, Estados Unidos, 1986
- [Brooks 91a] BROOKS R. A., "Intelligence without Reason", Computers and Thought Lecture, Proceedings of IJCAI-91, Sidney, Australia, 1991
- [Brooks 91b] BROOKS R. A., Challenges for a Complete Creatures Architectures, En: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptative Behavior, Meyer J.A. y Wilson S.W. editores, Ed. MIT Press/Bradford Books, Estados Unidos, 1991
- [Brustolini 91] BRUSTOLINI, José C., "Autonomous Agents: Characterizations and Requirements" Reporte Técnico CMU-CS-91-204, Pittsburgh, Carnegie Mellon University, Estados Unidos, 1991

- [Chapman 92] CHAPMAN, D., Vision, Instruction and Action, Ed. MIT Press / A Bradford Book, 1990, citado en [Maes 95]
- [Charniak 85] CHARNIAK, E. y D. MACDermott, Introduction to Artificial Intelligence, Ed. Addison-Wesley, Estados Unidos, 1985
- [Chin 91] CHIN, D.N., “Intelligent interfaces as agents”, en: Intelligent user interfaces, Sullivan y Tyler editores, pp. 177-206. Ed. ACM. New York, NY., Estados Unidos, 1991
- [Covrigaru y Linsay 91] COVRIGARU, Arie A. y Robert K. Lindsay, “Deterministic Autonomous Agents” En: AI Magazine, Estados Unidos, otoño, 1991
- [Drescher 92] DRESCHER, G.L., Made-up minds: A Constructivist Approach to Artificial Intelligence, Ed. MIT Press / A Bradford Book, 1991, citado en [Maes 95]
- [Etzioni y Weld 95] ETZIONI, Oren y Daniel S. Weld, “Intelligent Agents on the Internet: Fact, Fiction, and Forecast”, en IEEE Expert, Agosto, Estados Unidos, 1995
- [Fikes y Nilsson 71] FIKES, Richard E. y Nils J. Nilsson, “STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving” En: Artificial Intelligence 2(1971), Estados Unidos, 1971
- [Firby 87] FIRBY, R. James, “An Investigation into Reactive Planning in Complex Domains”, En: Proceedings of the AAAI-87, Estados Unidos, 1987
- [Foner 93] FONER, Leonard, “What’s An Agent, Anyway? A Sociological Case Study” Agents Memo 93-01, MIT Media Lab, Estados Unidos, 1993
- [Franklin 95] FRANKLIN, Stan, Artificial Minds, Ed. A Bradford Book / MIT Press, Estados Unidos, 1995
- [Franklin y Graesser 96] FRANKLIN, Stan y Art Graesser, “Is It an Agent, or Just a Program? A Taxonomy for Autonomous Agents”, borrador, Institute for Intelligent Systems, University of Memphis, Estados Unidos, 1995
- [Genesereth y Nilsson 88] Genesereth, Michael y Nils J. Nilsson, Logical Foundations for Artificial Intelligence, Ed. Morgan Kaufmann Publishers, Inc., Palo Alto, CA., Estados Unidos, 1988
- [Hendriks-Jansen 96] HENDRIKS-JANSEN, Horst, Catching Ourselves in the Act: Situated Activity, Interactive Emergence, Evolution, and Human Thought, Ed. MIT Press / A Bradford Book, Estados Unidos, 1996
- [Jennings y Wooldrige 96] JENNINGS, Nick y Michael Wooldrige. “Software Agents” en IEEE Review, January, Estados Unidos, 1996
- [Johnson 96] JOHNSON, Eric F., Graphical Applications with Tcl&Tk, Ed. M&T Books, New York, NY., Estados Unidos, 1996

- [Kaelbling y Rosenchein 90] KAELBLING, Leslie Pack y Rosenchein S., "Action and Planning in Embidded Agents", En: Designing Autonomous Agents: Theory and Practice form Biology to Engineering and Back, P. Maes editor, Ed. MIT Press / A Bradford Books, Estados Unidos, 1990
- [Kaelbling 93] KAELBLING, Leslie Pack, Learning in embedded systems, Ed. A Bradford Book / MIT Press, Cambridge, MA., Estados Unidos, 1993
- [Laurel 90] LAUREL, Brenda, "Interface Agents: metaphors with character", en The Art of Human-Computer Interface Design, editor Brenda Laurel, Ed. Addison-Wesley, Estados Unidos, 1990
- [Maes 89] MAES, Pattie, "The dynamics of action selection", en: Proceedings of the IJCAI-89 Conference, Detroit, Estados Unidos, 1989
- [Maes 90a] MAES, Pattie y Rodney Brooks, "Learning to Coordinate Behaviors", reimpresión de AAAI 90, American Association for Artificial Intelligence, Ed. Morgan Kauffman, Estados Unidos, 1990
- [Maes 90b] MAES, Pattie, "How to do the right thing", en: Connection Science J., special issue on Hybrid Systems, 1(3) (February 1990), también MIT- AILAB memo 1180, Estados Unidos, 1990
- [Maes 90c] MAES, Pattie, "Situated Agents can have Goals", En: Designing Autonomous Agents: Theory and Practice form Biology to Engineering and Back, P. Maes editor, Ed. MIT Press/Bradford Books, 1990
- [Maes 91] MAES, Pattie, "A Bottom-up Mechanism for Behavior-Selection in an Artificial Creature", En: From Animals to Animats. Proceedings of the First International Conference on the Simulation of Adaptative Behavior, Meyer J.A. y Wilson S.W. editores, Ed. MIT Press/Bradford Books, 1991
- [Maes 92] MAES, Pattie, "Learning Behavior Networks from Experience" En: Toward a Practice of Autonomous Agents Systems. Proceedings of the First European Conference on Artificial Life, F. J. Varela y P. Bourguine editores, Ed. MIT Press / Bradford Books, 1992
- [Maes y Kozierok 93] MAES, Pattie y Robyn Kozierok, "Learning Interface Agents", en: Proceedings of the AAAI 93 Conference, Ed. MIT Press, Cambridge, MA, Estados Unidos, 1993
- [Maes 94] MAES, Pattie, "Agents that reduce work and information overload" en: Communications of the ACM, 37(7):31-40, Estados Unidos, 1994

- [Maes 95] MAES, Pattie, "Modeling Autonomous Agents", en: Artificial Life Journal, editor C. Langton, Volumen 1, No. 1 y 2, Ed. MIT Press, Estados Unidos, 1995
- [Mahadevan y Conell 91] MAHADEVAN, S. y J. Conell, "Automatic Programming of Behavior Based Robots using Reinforcement Learning", en: Proceedings of the 9th National Conference on Artificial Intelligence, Ed. MIT Press, 1993, citado en [Maes 95]
- [Mataric 94] MATARIC, Maja J., Interaction and Intelligent Behavior, Tesis Doctoral, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, Estados Unidos, 1994
- [Mataric 89] MATARIC, Maja J., "Qualitative sonar based environment learning for mobile robots" En: SPIE Mobile Robots, Estados Unidos, 1989
- [Marks et.al. 88] MARKS, Mitchell, Kristian Hammond y Tim Converse, "Planning in an Open World: A Pluralistic Approach", En: Proceedings of the Cased Base Reasoning Workshop, Estados Unidos, mayo, 1988
- [McFarland 95] MCFARLAND, David, "Autonomy and Self-Sufficiency in Robots" En: The Artificial Route to Artificial Intelligence, Luc Steels y Rodney Brooks editores, Ed. Lawrence Erlbaum Associates, Estados Unidos 1995
- [Minsky 86] MINSKY, Marvin, The Society of Mind, Ed. Simon and Schuster, New York, Estados Unidos, 1986
- [Newell 81] NEWELL A., "The Knowledge Level", Reporte Técnico CMU-CS-81-131, Carnegie Mellon Universitu, Estados Unidos, 1991
- [Nilsson 92] NILSSON, N.J., "Toward Agent Programs with Circuit Semantics", reporte número STAN-CS-92-1412, Stanford University, Estados Unidos, 1992
- [Ousterhout 94] OUSTERHOUT, John, Tcl and Tk Toolkit, Ed. Addison-Wesley, Estados Unidos, 1994
- [Rossenblatt y Payton 89] ROSSENBLATT J. y D. Payton, "A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control, en: Proceedings of the International Joint Conference on Neural Networks, IJCNN, Washington DC, Estados Unidos, 1989
- [Russell y Norvig 95] RUSSELL, Stuart J. y Peter Norvig, Artificial Intelligence, A Modern Approach, Ed. Prentice-Hall, Estados Unidos, 1996
- [Sacerdoti 74] SACERDOTI, Earl D., "Planning in a Hierarchy of Abstraction Spaces" en Artificial Intelligence 5(1974), Ed. North Holland, Estados Unidos, 1974
- [Sheth y Maes 93] SHETH, B y Pattie Maes, "Evolving Agents for personalized information filtering", en: Proceedings of the Ninth Conference

on Artificial Intelligence for Applications, Ed. IEEE Computer Society Press., Estados Unidos, 1993

- [Shoham 93] SHOHAM, Yoav, "Agent-Oriented Programming", en: Artificial Intelligence, No. 60, Volumen 1, Estados Unidos, 1993
- [Tyrell 93] TYRELL T., Computational Mechanisms for Action Selection, Tesis Doctoral, Centre for Cognitive Science, University of Edinburgh, 1993
- [Welch 95] WELCH, Brent, Practical Programming in Tcl and Tk, Ed. Prentice-Hall, Estados Unidos, 1995
- [Wilkins 85] WILKINS, David E., "Recovering from Execution Errors in SIPE" Reporte Técnico 346 SRI, Estados Unidos, 1985
- [Wooldridge y Jennings 95] WOOLDRIDGE, Michael y Nicholas R. Jennings. "Intelligent Agents: Theory and Practice", enviado a: Knowledge Engineering Review, revisado junio 1995, Reino Unido, 1995