Programación natural de un robot social mediante diálogos

Javier Fernández de Gorostiza Luengo

Director:
Dr. Miguel Ángel Salichs Sánchez-Caballero

Escuela Politécnica Superior
de la Universidad Carlos III de Madrid

Programación natural de un robot social mediante diálogos

Programación natural de un robot social mediante diálogos

Javier Fernández de Gorostiza Luengo

Director:
Dr. Miguel Ángel Salichs Sánchez-Caballero

Tesis Doctoral

2010

Escuela Politécnica Superior
de la Universidad Carlos III de Madrid

Universidad Carlos III

Publication Data:
Javier Fernández de Gorostiza Luengo
Título
Universidad Carlos III

— A mi madre Mamen, a mi padre Martín. A mi hermana Beo. A mi hija Cricri. A John Coltrane —

AGRADECIMIENTOS

Sean mis primeras palabras de agradecimiento a mi amigo, compañero y padre investigador, Miguel Ángel Salichs, por todo el cariño mostrado desde el primer día que nos conocimos, por toda la confianza que ha depositado en mí, por una constante presencia y disponibilidad ante cualquier consulta, y por su ejemplo de mente crítica, actitud paciente y condescendencia.

A mi compañero Ramón Barber, por sus sabios consejos en cuanto a redacción y presentación de esta tesis, por toda su ayuda en general, y por haberme ayudado a integrarme en el grupo desde el primer día.

Quería dar las gracias a mi ordenador personal DELL Inspiron 6400 con su Intel Centrino Duo. Con él he pasado la mayoría del tiempo en los últimos cuatro años. Él ha ejecutado todos los programas que he ido implementando, sin ninguna queja, ni disfunción, recordándome una y otra vez que los errores en el sistema corrían bajo mi responsabilidad y que él realizaba lo que exactamente yo le había dicho. Él también ha permitido construir este documento, el texto, las figuras, tablas, referencias, etc. Asimismo, será en él donde realizaré la presentación que culminará el presente trabajo. En fin, gracias portátil, espero que algún día tu forma evolucione a un robot personal, y que pueda interactuar contigo como se describe en el presente trabajo, del cual tú eres uno de los principales protagonistas.

No olvidaré nunca la compañía de todos mis compañeros. Vayamos por partes.

Álvaro Castro, Fernando Alonso y, en su día, Rafael Rivas los "amos del calabozo", que no solo han dedicado mucho tiempo y esfuerzo en realizaciones de administración que no aparecerán directamente en sus respectivas tesis, ni en sus publicaciones, pero de los cuales nos hemos beneficiado todo el grupo, sino que siempre han estado dispuestos a echarme una mano en lo que hiciera falta. Por supuesto gracias a David Godoy, "el manitas", que prácticamente ha construido él solito a Maggie, además de mantener los ordenadores del laboratorio y atender cualquier necesidad que tuviéramos. Todo con mucha pulcritud, cuidado y cariño no habituales. Quizás solamente comparables a la amabilidad de Ángela Nombela. Gracias por toda tu ayuda y cariño. Gracias

también a Elena Delgado, qué pena que nos dejaste. Gracias a Ana Corrales. Gracias a Víctor por los aires nuevos que contigo han llegado al grupo. Gracias a María Malfaz por su compañía, sabios y prácticos consejos. Ella fue mi primera compañera. Gracias a Milagro Lafuente, a Dolores Blanco, a Alaan Khamis, a Felipe Zottola y a Santiago Garrido, que me ayudaron a integrarme en el departamento cuando todavía me costaba salir de mi timidez. Especiales gracias a Luis Moreno por su confianza, y por su conversación. Aprecio mucho su sinceridad, su sentido crítico y su acogida para conmigo. Especiales gracias también a Carlos Balaguer, quien me invitó a conocer y unirme al departamento contestando y atendiendo a un correo de un humilde estudiante más.

Por supuesto quería agradecer al resto de compañeros del "Robotics", por todo su cariño, aceptación y amistad. Especiales gracias a Concha por todos sus consejos para la redacción de esta tesis. Gracias a Paolo, Juan, Santi, Alberto, Mohamed, Julio, Pavel, Mario, Diego, Luigi, Raúl, Juan, Miguel, Javi, Ramiros, Fernandos, Carlos,... También al resto de compañeros del departamento Jose María Armingol, Arturo de la Escalera y Paco Pepe, por su confianza en un profe tan despistado.

Gracias a mis compañeros de despacho, con los cuales me he sentido tan acompañado y con los que me he divertido tanto. Espero no haberos molestado demasiado con mi música y mis extravagancias, Carla, Nico, Fer, Álvaro y Ana.

Gracias a Carmen y a Paqui por los más de dos mil cafés servidos...

Fuera de la universidad he de agradecer a mi hermano negro Michel tantas cosas... muchas de ellas solo pueden decirse musicalmente. Gracias a mi hermano mayor Daniotas, es un maestro en todos los sentidos. A Jaime, mi Jaime (cómo te echo de menos...). Gracias a Ikerman, "el genio", fuente de ideas a borbotones, Tsunami Killcore (Sophie), TeSa, Raurúrulo, Troglo, Beatmac, Pedro, Davibotis, Gallar, Varelus, Gonzalo, Sonia y Susana,... o ¿deberé decir Susana y Sonia? (suspiro...). Sonia, gracias por tu aportación sobre lingüística, y por tus palabras de ánimo. Todos habéis aportado mucho para que este trabajo sea posible. Especiales gracias a Tonia Raquejo, una auténtica maestra, por todo lo que me ha enseñado sobre arte, sobre lo monstruoso, sobre el mito de la creación, sobre el ser humano, pero sobretodo, gracias por su cariño y su aprecio.

Por supuesto, y a pesar de que me han destrozado el corazón, Cris, mi "hermana pequeña", gracias por todo tu apoyo en general y con nuestra nena en particular, a ti Ana... ¡qué cerca estuvimos!..., a ti Martinha, la mejor persona que he conocido nunca..., cuánta unión, cuánto apoyo y cuánto cariño...

No os olvidaré nunca, para mi desgracia...

Por supuesto, gracias a mis divinos y amados padres, a mi hermana Begoñurri, y a mi hija Cricri, a los que dedico esta tesis. Gracias a toda mi numerosa familia, en la que inlcuyo a "los bibitos" Tere y Jaime, sin los cuales esta tesis y otras muchas cosas no hubiesen sido posibles. Espero que lloréis un poquito en la presentación, ¿no?

Aún así, he de decir que siempre me ha acompañado y me acompaña un fuerte sentimiento de soledad... bueno no siempre. La única persona que consigue apaciguar ese sentimiento constante y que mejor describe lo que es la Interacción es John Coltrane. A ti, *Mr. J.C.*

RESUMEN

La tesis tiene tres partes conceptuales diferenciadas. Una referida a los **robots sociales**: motivaciones para su desarrollo, aspectos de diseño y funcionamiento. Se presentan las características más importantes de *Maggie*, un robot social desarrollado por el RoboticsLab[1], que ha servido de plataforma para el desarrollo de las otras dos partes: un sistema de gestión de diálogo y un sistema de edición verbal de secuencias.

El **sistema de gestión del diálogo** incluye dos habilidades de voz: habilidad de reconocimiento automático del habla, y habilidad de síntesis de voz con entonación controlada. El gestor del diálogo comunica ambas habilidades entre sí, controlando el texto que se sintetiza según las entradas del reconocedor de voz. Este control se especifica en un fichero escrito en el lenguaje estándar `voiceXML` que se interpreta en tiempo de ejecución.

Integrado con el sistema de diálogo se presenta un sistema de **edición verbal de una secuencia.** Una secuencia es una entidad que representa una red de movimientos del robot en interacción con su entorno, dentro del cual se ubica el usuario. El editor permite que la secuencia pueda ser editada y ejecutada sin necesidad de realizar cambios en el sistema, mediante un acceso al robot más natural y por tanto, más cercano a un usuario no experto.

Las secuencias creadas cumplen con todos los requisitos especificados en el estándar de SFC's, incluyendo ejecución de acciones en serie, esquemas de selección entre varias condiciones y ramas secuenciales que se ejcutan en paralelo. También se añade la posibilidad de creación de bucles. De este modo las funcionalidades de bajo nivel del robot se hacen accesibles a alto nivel para el usuario.

A modo de ejemplo, se presentan algunos resultados experimentales de edición y ejecución de secuencias mediante diálogos, tras los cuales se obtienen ciertas conclusiones.

Finalmente el trabajo concluye con la propuesta de algunas líneas futuras de investigación en interacción humano robot.

[1] http://roboticslab.uc3m.es

ABSTRACT

This dissertation has three different conceptual parts. One of them is related to **social robotics**: motivations, design aspects, and operation details. In this part, the social robot Maggie is described. This robot has been developed at the RoboticsLab and used as a platform for the development of the two remaining main parts of this thesis: a Dialogue Manager System and a Sequence Verbal Edition System.

The **Dialogue Manager System** includes some skills such Automatic Recognition Skill and Text-to-Speech Synthesis with prosodic control. The dialogue manager communicates both skills and parses a `voiceXML` standard language file.

A **Sequence Verbal Edition System** has been developed. It is integrated with the Dialogue System. A sequence can be defined as an entity that represents a net of robot movements in interaction with its environment, where the user is located. The edition system allows the edition and modification of the sequence without the necessity for making changes in the global system. Therefore, this way of operation makes the access to the robot more natural and intuitive for non-expert users.

Some experimental results are presented, such some examples of how the sequence edition and execution processes work. Also, some final conclusons are extracted and drawn.

The presented work gives some clues on how to solve some problems that are related to conversational interaction and natural robot programming, but this work opens more questions. Therefore, at the end of the thesis some human robot interaction issues are proposed as future works.

" [...]
Lalalí
Io ia
iiio
Ai a i a a i i i i o ia. "

Vicente Huidobro, *Altazor* (1931)

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

"El médico que solo medicina sabe no sabe ni medicina siquiera."
(Gregorio Marañon y Posadillo)

Cada vez más, la robótica demanda una estrecha colaboración entre distintas disciplinas como son la mecánica, electrónica, informática, inteligencia artificial, pero también la lingüística, psicología, sociología, neurología, medicina e incluso el arte. Con esta reflexión introducimos la presente tesis, que se enmarca en esta nueva robótica esencialmente multidisciplinar.

1.1. Estructura del documento

El documento está dividido en tres partes conceptuales principales. Una parte introductoria, donde se presentan el robot Maggie, plataforma de desarrollo de la presente tesis, y el estado del arte relacionado con el tema principal que se trata. Una segunda parte donde se realiza una presentación de las herramientas de voz y de edición de secuencias utilizadas. Y una parte final a modo de conclusión, donde se enumeran las aportaciones de esta tesis, y se proponen nuevas vías de trabajo relacionado.

Capítulo 1. Introducción donde se especifica la estructura del documento. Se expone una breve reflexión teórica sobre la relación de los robots personales con el ser humano, que concluye con la idea de *compañero existencial.* También se exponen los motivos principales para realizar esta tesis, cómo se han escogido los objetivos principales y secundarios, y qué aportaciones principales se han obtenido.

Capítulo 2. En este capítulo se realiza un estado del arte relacionado con el trabajo desarrollado en la tesis. Se divide en tres partes principales: en la primera se expone el nuevo paradigma de *programación natural por el usuario*

final, en la segunda se habla de sistemas de manejo de diálogo, y en la tercera de sistemas de diálogo en robots sociales.

Capítulo 3. En este capítulo se habla de *Maggie*, el robot social desarrollado por el RoboticsLab, que ha servido como plataforma para el desarrollo de esta tesis. Se exponen los dominios de *acciones* y *condiciones* utilizados para realizar *secuencias*.

Capítulo 4. En este capítulo se describe la implementación de la habilidad de reconocimiento automático del habla. Se realiza una pequeña introducción teórica sobre las técnicas utilizadas, donde, además se introduce el concepto de *gramática semántica*.

Capítulo 5. Análogamente al capítulo anterior, en éste se describe la implementación de la habilidad de síntesis de voz con emociones. También se realiza una pequeña introducción teórica haciendo hincapié en las técnicas que permiten controlar la entonación de la síntesis, o *prosodia*.

Capítulo 6. En este capítulo se exponen los detalles de una primera versión de un *editor verbal de secuencias* en el robot en lazo abierto, esto es, sin que el robot pueda tomar la iniciativa de la interacción y realizar las indicaciones pertinentes para establecer con el usuario, un conocimiento de base en común.

Capítulo 7. En este capítulo se presenta el sistema de gestión del diálogo desarrollado. Se exponen ciertos conceptos fundamentales de todo diálogo, y cómo, utilizando el estándar `voiceXML`, se llevan a la práctica en el sistema de gestión del diálogo que se ha desarrollado.

Capítulo 8. Una vez expuesto el sistema gestor de diálogo, se presenta una segunda versión del *editor verbal de secuencias*, el cual utiliza este gestor, para permitir editar la secuencia mediante *interacción hablada*.

Capítulo 9. En este capítulo se presentan algunos ejemplos de sistemas de diálogo para la edición de una secuencia.

Capítulo 10. En el último capítulo se establecen las conclusiones y aportaciones de la tesis, y se apuntan algunas líneas de trabajo futuro.

El escrito cuenta con una serie de apéndices cuyo propósito es el de facilitar la comprensión de toda la tesis, y así hacerla más accesible a investigadores de distintos campos.

Apéndice A. En este apéndice se ha realizado un compendio de los conceptos más importantes en relación a la interacción presencial entre humanos. Estos conceptos se consideran necesarios, si bien no suficientes, para la eficacia de cualquier sistema interactivo con el humano.

Apéndice B. En este apéndice se ha incluido un resumen de los fundamentos del diagrama funcional o, simplemente, *secuencia*, a partir de la descripción directa que se realiza del estándar SFC.

Apéndice C. En este apéndice se ha realizado el esfuerzo de tomar las partes más significativas de la teoría de lenguajes formales, que están directamente relacionadas con las herramientas utilizadas en el desarrollo de la tesis. Así por ejemplo, se presenta la *gramática de contexto libre*, que es utilizada por el reconocedor de voz, así como *máquinas de estados finitos*, que también es utilizada por el reconocedor de voz y por el sistema de diálogo, etc.

Apéndice D. Se incluye un resumen de los fundamentos de la descripción del lenguaje estándar `voiceXML`. Es en este lenguaje, en el que se implementa el sistema de diálogo.

Apéndice E. En este apéndice se exponen todas las reglas literales y semánticas de la gramática utilizada por el reconocedor automático del habla, en el caso en el que no hay interacción conversacional.

Apéndice F. Análogamente al apéndice anterior, se exponen todas las reglas literales y semánticas de la gramática utilizada por el reconocedor automático del habla, para el editor que incluye interacción conversacional.

1.2. Reflexiones sobre el porqué de los robots sociales

(a) Copa de cerámica lacónica con imagen de Atlas y Prometeo, atribuida a Archelisias II (s.VI a.C.)

(b) Pigmalión en un manuscrito de *Las Metamorfosis* de Ovidio (s.XV)

Fig. 1.1: *Representaciones mitológicas relacionadas con el mito de la creación*

En la construcción de un robot autónomo social o un robot personal, no solo vemos su aspecto interno, esto es, el robot en sí mismo, como un objeto tecnológico añadido, más o menos avanzado, para cuya construcción tratamos de resolver gran número de problemas técnicos. También vemos un aspecto externo a esta construcción, que no es sino la implicación de un estudio amplio acerca del ser humano y de su existencia. Los modelos desarrollados sobre el ser humano influyen en el diseño e implementación del robot, y viceversa.

En la figura 1.1 se muestran dos representaciones de mitos relacionados con la *creación*. Pigmalión creando a Galatea. Creador de una mujer como prótesis del hombre puesta al servicio de su creador. Creador sin *hibris*, sin querer ir más allá de los dioses. Una mujer creada desde el marfil (naturaleza animal). El amor de Pigmalión es un amor hacia sí mismo, que se lleva a la mujer. Por otro lado Prometeo[1], el creador trágico de los hombres, optimista, símbolo de las posibilidades humanas, titán filántropo que resulta castigado, inventor del sacrificio y del engaño a los dioses, el que da origen con el fuego a la técnica, a la investigación, al futuro, al número, las letras, las casas, las artes, pero que queda eternamente ligado a su propia obra, devorado por un águila que es él mismo. Todo tiene su precio, todo tiene su esfuerzo, nada es gratuito.

[1] "El que ve las cosas de antemano"

Las características del robot personal, este nuevo *objeto tecnológico*, son tales que lo distinguen claramente de un electrodoméstico clásico o incluso de un ordenador personal. No estamos hablando de una herramienta más con un propósito específico, sino de un objeto, o quizás ya habría que decir *sujeto*, que se acerca en su funcionamiento y en su expresión a nosotros mismos.

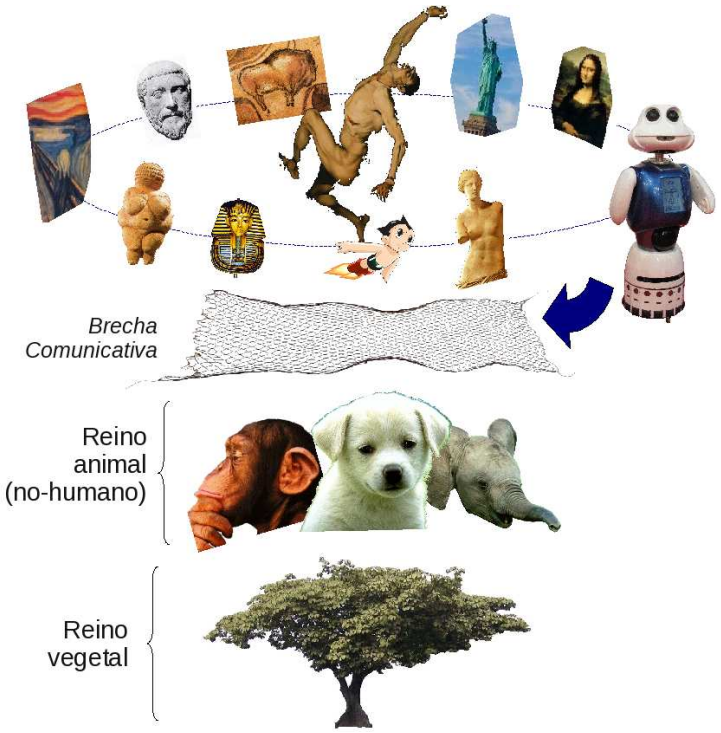


Fig. 1.2: *Ante la brecha comunicativa con el resto de seres, el ser humano, rodeado de múltiples representaciones. La cuestión plantea si los robots sociales cubrirán ese hueco de igualdad.*

¿Responde así un robot social a la necesidad humana de compartir su existencia con otro ser de igual a igual?. En la variedad de seres conocidos, el ser humano experimenta cierta empatía con los animales, con las plantas, pero con una limitación comunicativa obvia. Existe una *brecha comunicativa* evidente entre el ser humano y el resto de seres. Esto provoca una escisión muy clara que, de manera exclusiva, coloca al ser humano en un singular lugar dentro de la existencia.

Por otro lado, desde sus comienzos el ser humano ha mostrado una fuerte voluntad para rodearse de representaciones de la realidad: pinturas rupestres, esculturas con forma humana, muñecas, símbolos religiosos, representaciones

teatrales,... Estas representaciones han existido siempre e independientemente de la cultura que sea y distinguen claramente al ser humano del resto de animales. Cabe entonces plantearse la cuestión de si el origen de tal voluntad de creación pudiese estar relacionado con el hecho antes expuesto de esa limitación comunicativa con el resto de seres.

En la figura 1.2 se ha representado a un ser humano solitario en la existencia, pues ningún otro ser le iguala y por tanto, es incapaz de comunicarse de igual a igual con el resto de seres. Un ser humano rodeado de creaciones propias y representaciones de la realidad. Entre ellas, la de un robot social. La cuestión que quiere aquí plantearse es si el desarrollo de robots personales responde también a ese comentado "aislamiento en la existencia": *¿estaremos desarrollando robots sociales para tener un* ***compañero existencial*** *que sea igual (aunque distinto) a nosotros mismos?*

1.3. Motivaciones para esta tesis

El lanzamiento de robots personales al mercado hasta la fecha, ha pasado de ser un proceso paulatino con propuestas pre-robóticas japonesas como la mascota electrónica "Tamagotxi" de Bandai o el interactivo "Furby" a un proceso fuertemente creciente influenciado por el alto crecimiento en el desarrollo y la comercialización de los denominados productos de "nuevas tecnologías".

Desde la instalación en 1961 del primer robot en una planta de la General Motors de Unimation Inc. (Ternsted, NJ), hasta el año 2001 aproximadamente, 1,25 millones de robots se han instalado a lo largo y ancho del mundo. Según la Federación Internacional de Robótica (IFR), en el año 2008 las ventas de robots de servicio supusieron 11.000 millones de dólares americanos. En ese mismo año se estimaba que unas 20.000 unidades fueran creadas para robots de servicio (no industriales) en defensa, rescate y seguridad, lo que supone más de un 30 % del total de robots de servicio vendidos para uso profesional. A esta cantidad le siguen en 23 % de robots creados para su uso en el campo (como ordeñadores automáticos), un 9 % fueron para robots de limpieza, un 8 % para robots médicos, un 8 % para robots submarinos, 7 % robots para la construcción, 6 % robots móviles de uso general, etc. La expectativa para el período 2009-2012 es de que el número de robots de servicio vendidos crecerá en unas 49.000 unidades más[2]

Esta expectativa de crecimiento en la producción futura de robots no in-

[2]Consulta realizada en 2010 directamente del sitio web http://www.ifr.org

dustriales no solo arrastra a compañías multinacionales como Sony, Omrom, NEC, Honda, Toyota, etc, sino que ha provocado que en múltiples universidades y centros de investigación en los países desarrollados se haya abierto un amplio abanico de nuevos temas de investigación en robótica, algunos de los cuales se alejan y amplían los paradigmas de la robótica más clásica, basada en la mecatrónica.

A comienzos del año 2007 la revista Scientific American dedica a la robótica el número correspondiente al mes de Enero, en el que Bill Gates, presidente de la compañía más valorada del mundo, expone una analogía entre la industria robótica con la industria de PCs treinta años atrás, en la que los ordenadores dejaron de ser máquinas de lujo para grandes industrias y centros de investigación para irse introduciendo poco a poco en las casas y pasar a ser, en la actualidad, máquinas accesibles popularmente [Gates, 2007].

Por tanto, cabe esperar, que el desarrollo, la industrialización y la popularidad de estos nuevos robots sea creciente y relativamente inminente. Sin embargo, los robots no industriales en su acercamiento a un usuario no experto plantean todavía múltiples problemas de difícil y lejana solución. Efectivamente, a medida que estos robots comiencen a formar parte habitual de la vida corriente, será más habitual que sean utilizados por personas con mínimos conocimientos técnicos de robótica, automática o informática. Por tanto, plantean la necesidad de crear sistemas que permitan programar y utilizar los robots sociales de un modo fácil y flexible.

Esto ha venido a abrir un nuevo campo de investigación en interacción humano robot, orientado a programar un robot social de un modo natural, esto es, utilizando lenguaje natural, gestos, etc. Es en este campo donde se enmarca esta tesis.

El trabajo aquí presentado se ha realizado dentro del grupo de investigación RoboticsLab[3] de la Universidad Carlos III de Madrid, que cuenta con una treintena de investigadores de diversas partes del mundo: España, Francia, Rusia, Irak, Argentina, Noruega, India, Argelia, Marruecos, Perú, Venezuela,... La actividad investigadora del grupo abarca diversos proyectos en un amplio abanico que va desde el diseño e implementación mecatrónico de más bajo nivel, al estudio en ingeniería de control, ingeniería software, llegando a la investigación en temas más ligados con la inteligencia artificial. Los robots construidos abarcan robots humanoides como el RH1, robots asistenciales como ASIBOT, robots móviles con manipuladores como MANFRED, y robots sociales como MAGGIE.

[3]http://roboticslab.uc3m.es

La presente tesis se ha realizado gracias a los siguientes proyectos de investigación:

- Asistente Robótico Personal, del MCyT DPI2002-00188, en el período 2003 - 2005.
- Interacción igual a igual entre robots y humanos, del MCyT DPI2005-00309, en el período 2006 - 2008.
- AROS: una nueva aproximación a los robots sociales, del MCeI 2009/00006/001, en el período 2009 - 2011
- CP05-Aplicación de una metodología de diseño avanzada a robots de asistencia personal, de la CAM 2006/03562/001, en el periodo 2006 - 2007.
- Robots de servicios para la mejora de la calidad de vida de los ciudadanos en áreas metropolitanas, de la CAM 2006/03432/001, en el periodo 2006 - 2009.

1.4. Objetivos de la tesis

El objetivo principal de esta tesis es el de acercar un poco más el uso de un robot social a un usuario no experto. Para ello se ha realizado un estudio teórico sobre los modelos de la comunicación presencial humana, que se toma como base para los siguientes dos objetivos más concretos:

- Diseño e implementación de un **sistema gestor de diálogos** para mejorar la interacción humano robot.
- Diseño e implementación de un **sistema de edición verbal de secuencias** mediante el cual el usuario final construye o programa una secuencia de movimientos en el robot.

El sistema de gestión del diálogo debe ser capaz de adaptarse de modo que permita cambiar de diálogo en tiempo real, sin necesidad de realizar cambios en el sistema. La ejecución del diálogo debe atender, por un lado, a aspectos temporales del diálogo como es la gestión del turno, y por otro, a aspectos directamente relacionados con el contenido de la información que se quiere poner en común.

La edición y ejecución de una secuencia debe permitir concurrencia y paralelismo en la ejecución de acciones, así como la posibilidad de que la propia secuencia pueda ser modificada en tiempo de ejecución.

Al crear un editor verbal mediante diálogo de una secuencia se realiza un nuevo avance en el campo de la *programación natural* de un robot social.

2. PROGRAMACIÓN NATURAL MEDIANTE DIÁLOGO EN UN ROBOT SOCIAL

Podemos definir *programación natural* de un robot como una *programación fácil de aprender, efectiva y libre de errores que se realiza mediante una interacción natural entre usuario y robot.* Por tanto, la programación natural es un método de creación y modificación de un programa mediante un usuario no experto. Esto se denomina *programación por el usuario final*[1].

El modo más natural de comunicación entre los seres humanos es el habla. Por tanto, es de esperar que la interacción conversacional entre robots y humanos ofrezca una gran aportación en ese nuevo campo de estudio. Tradicionalmente, los sistemas automáticos de gestión de diálogo han sido desarrollados como un campo independiente, dentro de la Inteligencia Artificial, sin embargo su aplicación a la robótica no ha seguido el mismo grado de avance.

El capítulo consta de tres partes principales. En la primera, se exponen los trabajos más significativos en la actualidad referentes a la programación natural de un robot social por parte del usuario final. En la segunda, se realiza una exposición de los conceptos más importantes relacionados con la gestión del diálogo, en general. En la tercera, se realiza un estado del arte en sistemas de gestión de diálogo concretos implementados en robots sociales, exponiendo las diferencias, ventajas y desventajas entre unos sistemas y otros.

Por último se realiza un resumen a modo de conclusión donde se explican las mejoras que incorpora el sistema de edición verbal de secuencias mediante el sistema gestor de diálogo implementado en esta tesis, respecto a los sistemas descritos en el estado del arte.

[1] Del inglés end-user programming

2.1. Programación natural de un robot por el usuario final

En [Lozano, 1982] se realizan uno de los primeros estudios sobre el estado del arte en sistemas de programación de robots, que se dividen en tres grupos:

- **Sistemas guiados**, en los que el usuario mueve manualmente al robot a la posición deseada y éste recoge los datos obtenidos que luego procesa mediante algún filtro o algoritmo de aprendizaje. En esta categoría entran sistemas de *aprendizaje por demostración* (PbD), como son los trabajos realizados en [Hersch et al., 2008] o [Calinon et al., 2007], en los que se enseña al humanoide HOAP a agarrar objetos.

- **Sistemas de programación a nivel del robot**, en el que el usuario utiliza un lenguaje de programación asociado al robot. Este es el método más habitual de programación de robots, en el que están inmersos todos los desarrolladores.

- **Sistemas de programación a nivel de tarea**, en los que el usuario establece los objetivos de una tarea y el robot se encarga de llevarlos a cabo.

Posteriormente, en [Biggs and Macdonald, 2003] se ha realizado otra clasificación de lo métodos de programación natural en dos grandes grupos, según el usuario o programador tenga o no acceso directo al código de programación: *programación manual* y *programación automática*.

Los sistemas de programación manual pueden ser accesibles mediante texto, en los que se utiliza directamente un lenguaje de programación del robot con más o menos nivel de abstracción. También puede utilizarse una interfaz gráfica, usando diagramas de grafos, diagramas de flujo, etc.

Los sistemas de programación automática, salvo en sistemas simulados, implican que el robot esté en modo ejecución. Se trata de una programación en línea. Estos sistemas pueden seguir tres métodos distintos de programación:

- **Sistemas de aprendizaje:** se construye un programa a partir de una inferencia inductiva de ejemplos que realiza el usuario y las propias pruebas que realiza el robot.

- **Programación por demostración (PbD):** el usuario hace de profesor mostrando al robot los movimientos que puede realizar, o bien mediante tacto y contacto, o bien mediante voz y visión.

- **Programación por instrucción:** en tiempo de ejecución el usuario envía una secuencia de instrucciones al robot. Puede considerarse como el método de más alto nivel, pues el robot suele contar con una serie de primitivas o tareas preprogramadas, a las que las instrucciones dadas hacen referencia. El acceso a estas primitivas suele realizarse mediante reconocimiento de gestos y/o reconocimiento de voz. También se denomina aprendizaje basado en instrucciones (IBL).

El sistema de edición de secuencias que se presenta en este trabajo se encuadra dentro de este último grupo, de sistemas de programación automática mediante instrucciones.

Se sabe que la comunicación natural humana es multimodal. Por tanto involucra tanto mensajes verbales como no verbales. En [Birdwhistell, 1970] y [Davis, 1971] se realizan diversas exposiciones sobre la semántica de ciertos mensajes no verbales, y de cómo información verbal y no verbal se complementan en un mensaje final. Asimismo, en [Paul Watzlawick, 1967] se habla de la comunicación digital (o verbal) y la comunicación analógica (no verbal). Una recopilación de los conceptos más importantes ligados a las diferencias entre una y otra se ha incluido en el apéndice A.1.4. Las características de la comunicación verbal, como es su precisión, su capacidad de permitir compartir información e intercambiar conocimiento hacen que tenga una importancia especial respecto al resto de modos, también muy importantes, como se verá.

De modo natural el ser humano da instrucciones mediante gestos y, principalmente, voz. A la hora de dar instrucciones, los gestos pueden servir para dirigir la atención de modo natural sobre ciertos objetos del entorno, así como para realizar indicaciones deícticas e incluso relacionadas con la intensidad o tamaño de algo. Pero esta información, que es muy eficiente e intuitiva, no resulta tan precisa como la instrucción verbal. Esto justifica porqué en este trabajo se ha dado una importancia capital a la comunicación verbal, aun con conciencia de que este modo resulta insuficiente para una comunicación natural.

2.1.1. Sistemas que incluyen interacción no verbal

En [Strobel et al., 2002] se utiliza un sistema de programación natural que utiliza gestos con la mano y el brazo para controlar un robot manipulador que realiza la limpieza de una superficie. El robot es capaz de detectar hasta seis gestos estáticos mediante un sistema de visión en estéreo y dos tipos de gestos dinámicos mediante un sistema de seguimiento del gesto y cadenas ocultas de Markov. El robot interpreta los gestos de acuerdo a la

percepción del entorno y realiza hasta tres acciones distintas: limpiar la mesa con movimientos circulares, limpiarla con movimientos en líneas paralelas o dejar el utensilio de limpieza en un lugar señalado.

En [Steil et al., 2001] se presenta el sistema GRAVIS (Gestural Recognition Active VIsion System), de interpretación de gestos manuales. El robot detecta cuando se le presentan varios objetos en un entorno y focalizando la atención en el objeto que se le señale. Un manipulador es capaz de agarrar el objeto que tenga el foco de atención.

Estos son algunos ejemplos de sistemas de programación basados en instrucciones gestuales. La ventaja de utilizar gestos en la indicación de instrucciones es clara: es más rápida y natural. Sin embargo, tienen la desventaja de falta de precisión por las características ambiguas del mensaje no verbal (sobre las diferencias entre comunicación verbal y no verbal véase apéndice A.1.4).

Se observa en todos ellos que, en la práctica, los mensajes no verbales se interpretan de igual modo a como se interpretaría la misma información articulada verbalmente. Es decir, la información no verbal no se interpreta del modo tan complejo y cualitativo a como lo hacen los seres humanos, los cuales establecen la jerarquía de sus relaciones principalmente mediante comunicación no verbal. Tampoco se aprovecha la rapidez que en los seres humanos tienen los canales no verbales, dado que en robótica, precisamente los canales que tienen que ver con visión artificial involucran mucha lentitud de cálculo. La interpretación no verbal que se realiza impone una necesaria sintaxis clara en el gesto del usuario, para que pueda ser reconocido, lo cual se aleja de la naturaleza *asintáctica* del mensaje no verbal. Por tanto, la misma información que se le ofrece a estos sistemas de modo no verbal podría realizarse verbalmente sin demasiado coste o inconveniente para el usuario y con considerables ventajas en la eficiencia del sistema.

2.1.2. Sistemas que incluyen interacción verbal

En [Lauria et al., 2002] se propone un sistema que incorpora todos los elementos típicos de un sistema de programación natural mediante instrucciones: sistema de diálogo, dominio de acciones, acciones de edición del programa, etc.

El sistema permite enseñar a un robot móvil una secuencia de rutas por una ciudad en miniatura, mediante diálogo puramente verbal, sin el intercambio de otro tipo de mensajes no verbales. El sistema parte del estudio de un corpus en el que se han analizado hasta 144 rutas distintas, dadas por

24 sujetos, obteniendo un total de unas 72 instrucciones. El léxico obtenido es de 330 vocablos distintos, lo que supone un número demasiado alto como para integrar todo en una sola gramática.

El sistema de gramáticas implementado permite cubrir un 60 % de los enunciados del corpus. El sistema parte de una aceptación total de la existencia de errores de reconocimiento e identificación de los enunciados del usuario, lo cual resulta razonable teniendo en cuenta que estos errores ocurren también en la interacción verbal entre seres humanos. Estos errores tratan de ser cubiertos mediante un sencillo sistema de diálogo basado en la herramienta TRINDI [Lauria et al., 2001], que permite realizar consultas cuando haya un fallo en el reconocedor de habla o inconsistencias en la identificación de los elementos semánticos detectados.

Uno de los problemas que se plantea en este sistema es el de cómo relacionar los elementos del lenguaje percibidos con el dominio de acciones del robot. Para resolverlo se crean dos conjuntos distintos cuyos elementos deben ser relacionados. Los elementos del primero se denominan "símbolos" y están directamente relacionados con las indicaciones que da el usuario, identificadas en el reconocimiento del habla. El segundo conjunto consta de una serie de acciones primitivas de movimiento en el entorno (avanza, para, gira a la derecha, etc). Estas acciones están escritas en `Python`, lenguaje interpretable, para facilitar su combinación en una secuencia.

Al comienzo de la ejecución, el sistema cuenta con una relación uno a uno que se mantiene fija entre símbolos y acciones. Cuando el usuario crea un nuevo procedimiento (una nueva ruta) al robot ello implica la unión de varias primitivas. El sistema crea un nuevo código de ejecución (en `Python`) asociado al procedimiento. Este nuevo código combina las distintas primitivas en serie.

El sistema realiza una comprobación práctica de que la secuencia construida es factible físicamente. Para ello utiliza cadenas de estado-requisitos-acción-nuevo estado, que le permiten detectar si existe alguna inconsistencia, esto es, si la cadena se rompe en algún punto. En este caso, el sistema de diálogo indica la inconsistencia al usuario, si bien no se realiza ninguna corrección.

Este sistema pone de manifiesto el potencial y la concisión que ofrece el lenguaje hablado a la hora de expresar reglas y secuencias de comandos.

En otros trabajos presentados en [Dominey et al., 2007], con la motivación de querer acercar los robots personales al usuario no experto, se presenta un sistema de programación natural a través de lenguaje verbal. Se utiliza el robot humanoide HRP-2, del AIST (National Institute of Advanced In-

dustrial Science and Technology) de Japón. El sistema de diálogo utilizado es RAD (Rapid Application Development)[2], que ofrece el CSLU (Center for Speech and Language Understanding, EEUU). Esta herramienta es una interfaz gráfica a un sistema de diálogos basado en huecos de información.

El sistema cuenta con un dominio de 7 acciones primitivas: abrir y cerrar la mano izquierda o derecha, rotar la cintura y mover los brazos en posición neutra. Con estas primitivas se construyen lo que denominan "macros", como una secuencia seguida de acciones. Para ello el sistema cuenta con un repertorio de cinco comandos de edición: comienza, espera, continúa, OK y ejecuta la macro.

El objetivo de este trabajo es el de acercarse paulatinamente al desarrollo de un sistema general para programar robots a partir de una serie de comandos, que se pueda adaptar a cualquier plataforma robótica. De este modo mejorar trabajos de cooperación entre el humano y el robot.

[2]http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/tutorials/index.html

2.2. Sistemas de gestión de diálogo

Los sistemas de gestión de diálogo vienen siendo desarrollados desde los años sesenta, y hasta la fecha varios paradigmas distintos han dado a luz múltiples plataformas para la gestión del diálogo. Existen así desde sistemas basados en reglas entre patrones hasta sistemas que incluyen técnicas de planificación, pasando por sistemas basados en huecos de información.

2.2.1. Esquema general de un sistema manejador del

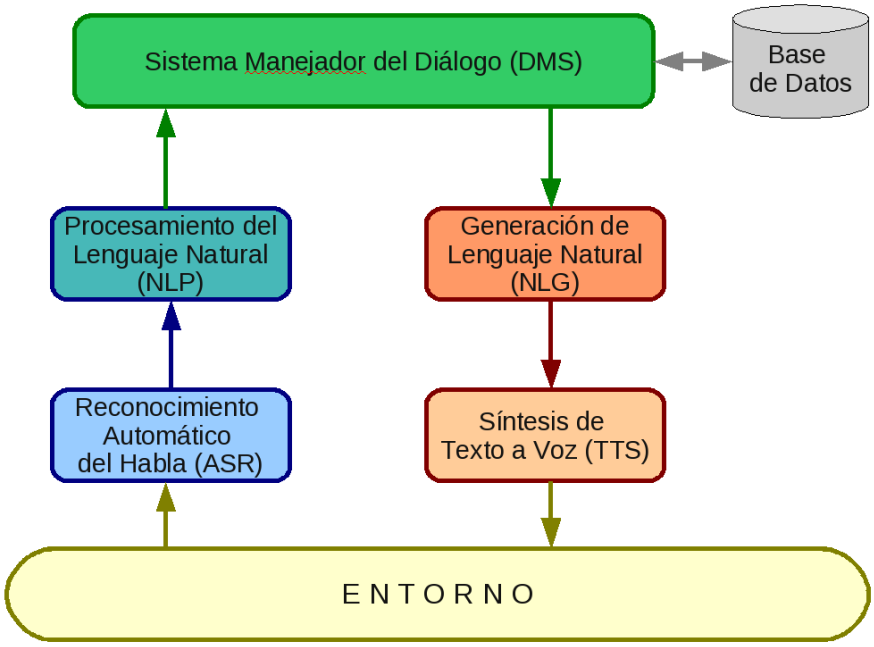


Fig. 2.1: *Esquema general de un sistema de diálogo*

En la figura 2.1 se representa un esquema general de un sistema automático de manejo del diálogo hablado. Éste consta de tres capas diferenciadas.

En una primera capa conectada a los dispositivos de adquisición y expresión de la señal acústica están los módulos encargados de transformar dicha señal en información computable. Es en esta capa donde se ubican el módulo de reconocimiento automático del habla (ASR) y el de síntesis de texto a voz (TTS). Por encima de esta capa está la que se encarga del procesamiento del lenguaje natural. En la parte de percepción, el módulo de procesamiento de lenguaje natural (NLP) se encarga de transformar la información morfosintáctica recibida del reconocedor de habla, en información semántica, que resulte relevante al manejador del diálogo. Éste realizará las operaciones oportunas para escoger qué expresar, cómo y cuándo, e incluso, si ha de tomar información de una fuente externa, representado en la figura como una base de datos. La construcción de la expresión pasa por la generación de lenguaje natural (NLG) y de ahí a la síntesis de texto a voz.

2.2.2. Paradigmas en la gestión del diálogo formal

Considerando el diálogo en su concepción formal[3] hacemos un compendio de sus características estructurales más relevantes que ofrece la literatura hasta la fecha. En [McTear, 2004] se realiza una interesante descripción de hasta dónde la tecnología del habla ha llegado a implementar un diálogo entre el humano y la máquina. En ella se describen los paradigmas más importantes en la gestión del diálogo, haciendo hincapié en los sistemas basados en huecos de información, muy utilizados en plataformas de atención al público, consulta, reserva y compra de billetes de vuelo, etc. Dentro de estos sistemas se destaca el lenguaje estándar `voiceXML`, y cómo desarrollar un gestor de diálogo siguiendo este estándar.

Los primeros sistemas de diálogo pertenecían a aplicaciones muy concretas. Hasta el desarrollo de técnicas de la inteligencia artificial no comienzan a desarrollarse paradigmas de diseño uniformes. De aquellos primeros sistemas cabe destacar BASEBALL [Green et al., 1963] capaz de contestar de modo automático consultas a cerca de los resultados deportivos de la liga de baloncesto. Ya incluía, por tanto, un reconocimiento del concepto que solicitaba el usuario incluyendo campos como fechas, nombre de los equipos, etc. También incluía el acceso a una base de datos externa, y la conversión del resultado de la consulta a un enunciado.

El sistema STUDENT [Bobrow, 1968] era capaz de traducir ecuaciones algebraicas expresadas en lenguaje natural. Incluía un traductor que cons-

[3]Sobre la diferencia entre un diálogo natural y una concepción más formal véase la introducción del capítulo 7

truía una expresión algebraica a partir de una descripción natural. Luego resolvía la expresión y volvía a traducir el resultado a lenguaje natural. Así por ejemplo, era capaz de resolver problemas del tipo *"si maría tiene el doble de edad que Ana tenía cuando María tenía la misma edad que Ana y María tiene 24 años, ¿qué edad tiene Ana?"*, etc.

Quizás, uno de los sistemas de diálogo más conocidos haya sido SHRDLU [Winograd, 1972]. Interacciona con el usuario tomando como dominio temático un pequeño entorno virtual de bloques de distintas formas y colores, con los que se puede realizar ciertas acciones gobernadas por reglas físicas. El sistema utilizaba análisis de los enunciados del usuario, en los tres niveles lingüísticos: sintáctico, semántico y pragmático. En el nivel sintáctico detecta la función de cada palabra en la frase. Semánticamente, relaciona las frases analizadas con el entorno virtual. De este modo podía resolver ambigüedades semánticas. Desde el punto de vista pragmático era capaz de concebir las posibilidades del entorno virtual y actuar sobre él, incorporando además un sistema que le permitiera descartar acciones o efectos en el mundo virtual físicamente imposibles.

Más adelante, a partir de los años 70, con el auge en inteligencia artificial comienzan a aparecer aproximaciones al diálogo con cierta profundidad teórica. Es el comienzo del procesamiento del lenguaje natural. Así, podemos realizar una clasificación de los sistemas de gestión de diálogo según sus mecanismos de funcionamiento:

- **Máquinas de estados finitos**, que modelan el diálogo en distintos estados. El cambio de un estado a otro viene marcado por la detección de una frase o palabra clave. En este grupo podrían también incluirse los sistemas de diálogo basados en patrones, en los que se asocia directamente una plantilla de salida ante cierto tipo de entrada.

- **Basados en huecos de información.** En este modelo existe una representación explícita de los distintos estados de la información. Un algoritmo de control realiza los cambios en estos estados mediante una interpretación de la información relevante del lenguaje natural de entrada. Si existe un modelo del usuario, es un modelo sencillo donde se guardan, por ejemplo, las características y preferencias del usuario.

- **Sistemas multiagente con planificación.** Contienen un modelo de intenciones, objetivos y creencias tanto del sistema como del usuario. El control del diálogo se realiza estableciendo un conocimiento de base común, gracias a que el sistema va guardando un historial del contexto de la conversación.

Estos paradigmas se refieren a diferencias en cuanto al gestor o manejador del diálogo de la figura 2.1. La funcionalidad en los componentes de entrada y salida del resto del sistema suelen ser muy parecidos tanto para unos mecanismos de gestión de diálogo como para otros.

Control del diálogo basado en huecos de información.

Aunque la literatura distingue estos sistemas de los controlados por una máquina de estados finitos, en esta categoría entran intérpretes que por dentro funcionan también como una máquina de estado finitos. Lo que ocurre es que los primeros realizan un control muy sencillo en el que cada estado está asociado a una salida y el cambio de estado ocurre ante una entrada que apenas llega a procesarse a nivel lingüístico. Por tanto son profundamente deterministas.

Los sistemas basados en huecos de información tienen asociado un algoritmo de control que va comprobando en dichos "huecos", qué información relevante no ha sido completada a partir de los enunciados del usuario. El algoritmo reacciona realizando las acciones de consulta pertinentes según los huecos de información que queden. Por tanto, el cambio de estado no está del todo predeterminado.

La información relevante sí que se establece a priori, mediante "huecos" o "ranuras". Así por ejemplo, para una aplicación que quiera tomar los datos personales de un usuario (nombre, apellidos, dirección y teléfono), los huecos de información podrían configurarse del siguiente modo:

```
Frame: usuario
  [nombre]
  [apellidos]
  [dirección]
    [calle]
    [número]
  [teléfono]
    [fijo]
    [móvil]
```

Estos huecos de información comúnmente se denominan *campos*.

A este grupo de gestores de diálogo pertenece la especificación del estándar del `W3C` (World Wide Web Consortium) `voiceXML`[4]. Este lenguaje de etiquetas está concebido para aplicaciones de atención telefónica, aunque su

[4] http://www.w3.org/TR/voicexml20/

uso puede ser extensible a otro tipo de aplicaciones. Es el lenguaje utilizado en el sistema gestor de diálogo que se ha desarrollado en el presente trabajo. En el apéndice D se ha desarrollado un manual sencillo y breve de `voiceXML`. El sistema gestor desarrollado se explica en la sección 7.

Otro sistema de este tipo es SALT (Speech Application Language Tags) concebido para aplicaciones web, que incluye también etiquetas en HTML y XHTML, permitiendo así cierta multimodalidad [Wang, 2002]. El `W3C` está revisando considerar este lenguaje como un estándar en la red, si bien todavía le falta cierta madurez. Por ejemplo, no incorpora un algoritmo de intérprete de las etiquetas propio, sino que el control del diálogo, esto es, la secuencia y coordinación de los eventos del habla deben ser codificados explícitamente según la plataforma en la que se utilice, lo cual supone una fuerte desventaja frente a `voiceXML` concebido como un sistema completo, en este sentido, autónomo.

Control del diálogo basado en planificación.

Son sistemas multiagente que modelan el diálogo como un proceso en colaboración entre el sistema y el usuario, que quiere realizar una tarea y que necesita comunicarse para ello. Por tanto, estos sistemas son más adecuados cuando la aplicación está concebida para la resolución de un problema o tarea mediante negociación.

En [Pallotta, 2003], se describe el desarrollo de un modelo cognitivo del diálogo basado en sistemas multiagentes encargados de reconocer un plan en las "intenciones" del usuario y cooperar con él. Además se introducen conceptos muy relevantes en la definición formal de un diálogo, como es el de *acto del habla* y *acto del diálogo* que serán desarrollados en el capítulo 7, donde se explica el sistema gestor de diálogo desarrollado en esta tesis.

La mayoría de los sistemas basados en planificación, utilizan también la definición de un dominio de huecos de información que puede servir tanto para definir la tarea del diálogo, como de apoyo para resolver una tarea especificada por la intenciones y/o deseos del usuario.

Uno de los primeros sistemas en este sentido es GUS (*Genial Understander System*) [Bobrow et al., 1977]. Servía de agente de viajes expendedor de billetes de avión. Este sistema controla el diálogo utilizando también huecos de información, pero es capaz de inferir un plan por parte del usuario y tenerlo en cuenta a la hora de elaborar un enunciado de salida. Para ello incorpora un procesamiento del lenguaje natural que le permite resolver expresiones relativas a entidades ya nombradas, del tipo *"el siguiente", "por la*

tarde", ... Los huecos de información corresponden con las ciudades de salida, de llegada, horarios, etc.

ATIS (Air Travel Information Service) desarrollado a través de un programa del DARPA o SUNDIAL (Speech UNderstanding in DIALogue) en Europa, involucran numerosos grupos de investigación que intentan construir un sistema de diálogo también basado en un control cooperativo con el usuario.

Otras propuestas se basan más en la dinámica del estado de información o representación interna del diálogo, como es TRINDI basado en MIDIKI [Larsson et al., 2004]. Estos sistemas híbridos están entre los basados en estructuras o huecos de información y los basados en planificación pura. Los estados de información registran la información compartida a lo largo del diálogo. Esta información varía según unas reglas en forma de algoritmos que pueden modificarse en tiempo de ejecución, y que dependen de una estrategia. Los algoritmos de actualización de los estados de información pueden estar escritos en PROLOG.

Así, dada una tarea general del diálogo, por ejemplo realizar una reserva de un billete, se establece una agenda en forma de pila dinámica con un plan a seguir para realizar la tarea. Cada plan consta de un conjunto de estrategias. A su vez, cada estrategia consta de un conjunto de movimientos de diálogo, que es la unidad básica del discurso contenida en un enunciado. Un movimiento del diálogo supone una acción comunicativa por el sistema, por ejemplo, realizar una pregunta, una respuesta, etc.

A través del proyecto DARPA Communicator[5], se ha desarrollado un sistema que plantea una arquitectura como plataforma base que puede ser aplicada a múltiples dominios: planificación de viajes, reserva de hoteles, reserva de vuelos, comunicación de un soldado con un sistema distribuido, etc, muy similar a MIDIKI. El sistema está orientado a resolver una tarea. Esta tarea viene fijada por un marco que se interpreta en tiempo de ejecución, que consta de varios huecos de información. Cada hueco de información está asociado a una acción comunicativa, representada mediante una plantilla donde se describen las cuestiones a realizar al usuario. Los usuarios pueden dar la información en cualquier orden y el sistema realiza las cuestiones convenientes hasta que el dominio de huecos de información esté completo. El marco de huecos de información se establece a priori en un fichero de texto.

Por último, cabe destacar el sistema TRAINS y TRIPS, su extensión posterior (The Rochester Interactive Planning System) [Ferguson and Allen, 1998].

[5]http://www.speech.cs.cmu.edu/Communicator/

Este sistema integra reconocimiento y procesamiento del lenguaje natural, procesamiento del discurso a través de planificación y de reconocimiento del plan del usuario. Está concebido para asistir al usuario en la resolución de un problema relacionado con el dominio de la logística en el sistema de transporte por tren: realización de horarios, planes, etc. Consta de tres componentes principales:

- **Agente manejador del comportamiento** a llevar a cabo una vez escogido el plan a llevar a cabo.
- **Agente manejador de la generación** de la salida verbal del sistema: generación de lenguaje y síntesis.
- **Agente manejador de la interpretación**. Interpreta la entrada del usuario, reconocimiento del habla e interpretación.

El sistema utilizado identifica la entrada del usuario con ciertos *actos del habla*, que están relacionados con un plan. La base del sistema es un estado del diálogo en el que se parametriza el contexto del discurso, incorporando información de los objetivos planteados, las soluciones encontradas, los recursos disponibles y las situaciones según un modelo del mundo. En el estado del diálogo también se incorpora información relacionada con el diálogo en sí. Por ejemplo si hay o no obligación de responder al usuario, etc.

Modelos externos o de correspondencia entre patrones

Hasta aquí se han descrito modelos del diálogo y del procesamiento del lenguaje natural, de alguna manera "internos", es decir, basados en una interpretación del contenido percibido según el objetivo del diálogo. También existen sistemas de gestión del diálogo basados en simplemente hacer corresponder patrones, lo cual podemos considerar como un modelo externo, dado que no llegan a profundizar en una interpretación semántica de la conversación. Sorprendentemente estos sistemas dan muy buenos resultados como conversadores simulados.

ELIZA [Weizenbaum, 1966] simulaba a un psicoterapeuta y tuvo bastante éxito en gran número de pacientes que se afirmaron sentirse comprendidos. El control del diálogo se realiza por reglas de asociación de respuestas a palabras clave del usuario.

Cabe destacar que hasta aquí los sistemas de diálogo descritos no incorporan la posibilidad de que el sistema mismo tome la iniciativa de la

conversación, de modo que el usuario siempre ha de tener la iniciativa, y el sistema se limita a contestar ante las entradas del usuario.

PARRY [Colby, 1975] cuenta con unos 6000 patrones que simulan un paciente paranoico. Es capaz de tomar iniciativas de conversación lo que ha provocado que se le llegue a confundir, por parte de psiquiatras profesionales, con un verdadero paciente paranoico.

Pero quizá el sistema más famoso y elaborado sea A.L.I.C.E. [Wallace, 2000] ganador del premio Loebner al chatbot más parecido a un ser humano. Su potencia reside en primero, representar los diálogos en un formato de lenguaje interpretado muy versátil (Artificial Intelligent Markup Language (AIML)) basado en el paradigma "divide y vencerás" frente al "compón y conquista"; y, segundo, en el gran número de reglas, patrones o categorías en que se basa ($>$150,000) cantidad además en constante ampliación.

2.3. Gestión del diálogo en robótica

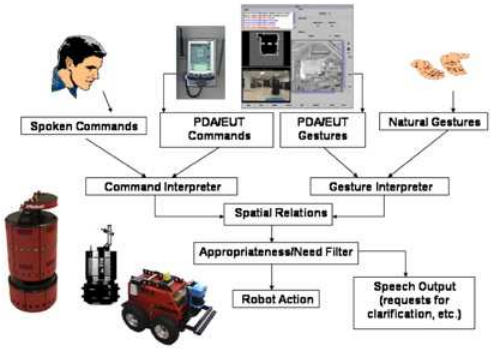
Los sistemas de diálogo descritos y los implementados en robótica no han seguido el mismo ritmo de desarrollo. Los robots comienzan a desarrollarse en paralelo a los mismos sistemas de computación que implementan gestión del diálogo, pero al incorporar aquellos también un cuerpo con sensores y actuadores, el desarrollo de un gestor del diálogo en robótica ha seguido su propio camino: la mayoría son sistemas *ad hoc* que solucionan un problema concreto dentro del propósito del robot y que simplemente, abren la puerta de la comunicación verbal dentro de un sistema mucho más general del que forman parte, y al cual se subordinan.

En esta sección se describen ejemplos de sistemas de gestión del diálogo implementados en robots sociales. La descripción se divide en tres partes correspondientes a los tres tipos de control de gestión del diálogo, que se han comentado: basados en máquinas de estados finitos, en huecos de información y en planificación.

2.3.1. Gestores basados en máquinas de estados finitos

El centro NCARAI (Navy Center for Applied Research in Artificial Intelligence) tiene un departamento dedicado al estudio de la interacción multimodal humano robot. Su propósito es el de implementar robots personales que cooperen con el trabajo del ser humano. Para ello se han centrado en el estudio del lenguaje natural humano y más en concreto, en la resolución de deixis y anáforas, lenguaje espacial, etc. Principalmente han traba-

(a) Robonaut, robot asistente para astronautas en el espacio



(b) Interbot, arquitectura para interacción multimodal

Fig. 2.2: *Interacción multimodal humano robot del centro NCARAI*

jado con dos tipos de robots distintos: **Robonaut** [Schultz, 2004] y **George** [Trafton et al., 2006].

En la figura 2.2(a) se muestra a Robonaut. Está compuesto por un torso de humanoide incorporado a un vehículo espacial. Este robot maneja dos brazos y una cabeza y es capaz de interpretar gestos mediante visión. Incorpora también un interfaz de lenguaje hablado (síntesis y reconocimiento). Está pensado para ayuda y cooperación en el espacio al astronauta: acercarle herramientas, guía de trabajo, etc. Otros modelos parecidos se están desarrollando para asistencia al soldado.

George es un robot modelo Nomad200 que incorpora capacidades automáticas para navegación y reconocimiento de objetos como son: construcción de mapas, localización, planificación de caminos, evasión de obstáculos, detección de obstáculos por color,... En su repertorio de actuaciones a parte de las de navegación básicas de movimiento controlado, incorpora la capacidad de esconderse respecto a un usuario. Esta capacidad implica el que el robot infiera según su posición relativa con el usuario, si este va a ser capaz de verle o no.

En la figura 2.2(b) se muestra la arquitectura utilizada. Se denomina Interbot [Cassimatis et al., 2004] e integra los distintos canales o modos en la interacción natural humana como son lenguaje hablado y gestos así como otros canales exclusivamente electrónicos: comandos y gestos por PDA en un módulo que interpreta toda esta información de acuerdo a relaciones espaciales subjetivas al robot. Estas relaciones son utilizadas por un módulo

que interpreta las necesidades del usuario en el contexto y de ahí el sistema construye una secuencia de acciones, incluido la expresión verbal. En [Skubic et al., 2004] se propone un sistema de diálogo multimodal capaz de fusionar información verbal con información gestual y a través de dispositivos gráficos como mapas en una PDA, etc. Sin embargo, en lo que se refiere a la gestión del diálogo en sí, el sistema está basado en una máquina de estados finitos determinista que maneja un pequeño conjunto de comandos. El sistema es de iniciativa centrada en el usuario, esto es, se limita a responder si ha entendido o no su mensaje multimodal.

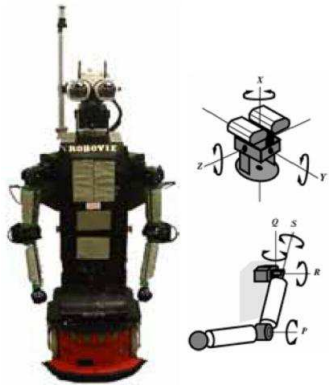
La arquitectura se centra en la interpretación del lenguaje espacial natural humano que incluye: deixis, anáforas y señales espaciales subjetivas, como "mi derecha, detrás tuyo, ahí,..." y otros aspectos espaciales del lenguaje, relativos a la orientación de los objetos del entorno entre sí. Por ejemplo, cómo resolver dependencias del tipo: *"vete a la caja que hay detrás de la columna."* [Skubic et al., 2004].

Esto favorece la capacidad de cooperación de los robots bajo esta arquitectura así como su interacción multimodo. Así, en [Trafton et al., 2006] se mezcla cognición, percepción y acción para implementar el juego del *escondite* entre el robot y un niño. Pero el robot cuenta con un repertorio de acciones reducido, un diseño poco amigable, y un bajo grado de autonomía. Asimismo, la arquitectura propuesta no maneja parámetros fundamentales para la interacción eficaz con el humano que den cuenta de cómo de contento está el humano en cada instante de la interacción, sus propósitos últimos, etc. Tampoco incorpora grandes posibilidades de aprendizaje y adaptación.

Por tanto, la interacción humano robot se ha centrado sobretodo en la fusión multimodal, dejando a un lado los otros aspectos muy profundos del diálogo verbal en sí: percepción y emisión de enunciados naturales, reparto de los turnos y otros aspectos temporales de la interacción, establecimiento de un conocimiento compartido, etc (sobre las características de la interacción conversacional con un robot véase sección 7.1.1).

En [Ishiguro et al., 2001] y [Ishiguro et al., 2002] se presenta a **Robovie**. Este robot personal ha sido desarrollado en el *Intelligent Robotics and Communication Laboratory del Advanced Telecommunications Research Institute (ATR)* en Kyoto (Japón). Robovie está concebido para la investigación en interacción humano robot. Ha sido probado en entornos con niños así como con ancianos. Como dato curioso, un anciano, obligado por deficiencias de movilidad a estar en una silla de ruedas, se levantó ante la simple presencia del robot.

Aunque lo denominan humanoide, la base de Robovie es una base móvil



(a) Algunos GDL de Robovie

(b) Robovie como asistente personal

Fig. 2.3: *Robot Social Robovie por los laboratorios ATR*

construida sobre ruedas. Encima de la base incorpora un torso, dos brazos con 4 GDL cada uno, así como una cabeza donde se han incorporado dos cámaras de vídeo móviles a modo de ojos, que permiten contacto visual. También incorpora una especie de piel a modo de sensor táctil, sensible a la presión. Mide 1,20m y pesa unos 40Kg.

La arquitectura de control propuesta ha sido presentada en diversos trabajos [Kanda et al., 2002] [Kanda et al., 2004]. Es una arquitectura basada en comportamientos, divididos en elementales (contacto visual, movimiento de brazos como "en marcha", señalar objeto,...) y modulares (ir a un punto, esperar cierto texto,...). El control de los distintos comportamientos se hace mediante una gramática de reglas de episodio: secuencia de comportamientos, interrumpir secuencia, transición a un módulo de comportamientos reactivos, ... A pesar de haber llegado a implementar unos 700 episodios sobre unos 100 comportamientos (modulares y elementales) diversos experimentos con niños mostraron que éstos en menos de media hora de jugar con el robot ya se aburrían.

Aunque la arquitectura de control basada en comportamientos ha sido utilizada en diversos trabajos anteriores, en esta se rompe con la idea tradicional de interacción maestro - esclavo y considera al humano como parte del entorno. Sin embargo, El sistema de diálogo es prácticamente inexistente, y la interacción por voz se reduce a un simple dispositivo de entrada/salida.

2.3.2. Gestores basados en huecos de información

En [Haasch et al., 2004] se presenta a **BIRON**. Este robot ha sido diseñado y construido por la universidad de Bielfeld (Alemania) con el propósito de investigar cómo puede un robot móvil asistir al usuario dentro de una casa o en una oficina a través de una interacción natural.

En la figura 2.4 se representa una foto del robot junto con la arquitectura de control desarrollada. Se trata de una arquitectura híbrida dividida en tres niveles: uno reactivo, otro intermedio y otro deliberativo. En el nivel reactivo se sitúan las capacidades sensorimotoras del robot: reconocimiento y síntesis de voz, percepción del usuario, percepción de objetos, navegación, etc. En el nivel deliberativo, un supervisor principal controla al resto de la arquitectura a través de un sistema de visión y localización, otro de seguimiento dinámico de un "tema" y a través de un sistema de diálogo. El sistema de control está orientado a la realización de una tarea, y es esta realización la que va dictando el funcionamiento de cada una de las capacidades del robot.

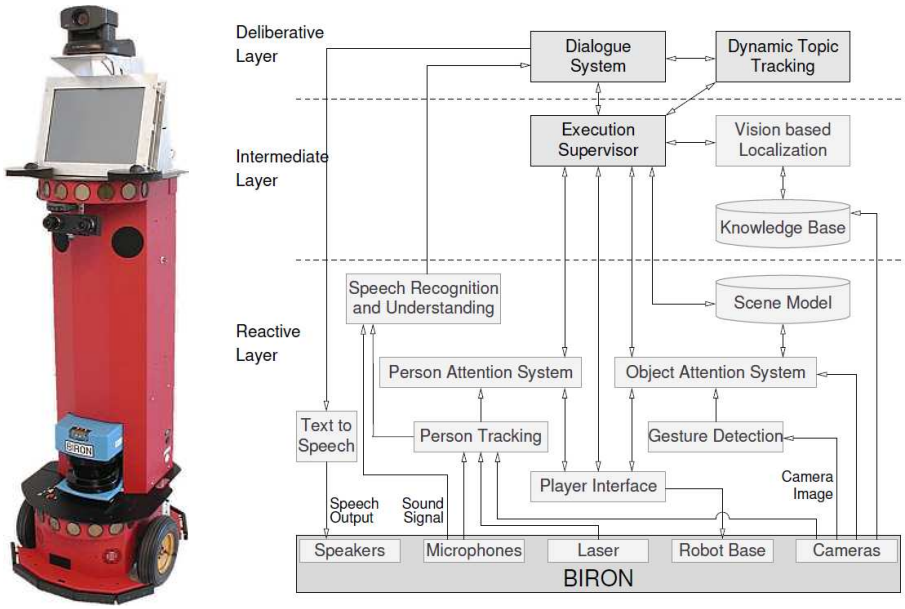


Fig. 2.4: *BIRON (BIelfeld RObot CompanioN)*

En [Toptsis et al., 2004] se explica el sistema de diálogo desarrollado en BIRON. El control se realiza mediante una máquina de estados finitos que se define en `XML`. Está basado en huecos de información que se denominan *"entidades semánticas"*. La tarea del manejador del diálogo es la de completar

los suficientes huecos de información hasta completar un objetivo identificado por un *"estado-objetivo"* en la máquina de estados finitos. Cada estado viene identificado por tres estructuras de datos distintas: una referida a las entradas del usuario, otra al estado del sistema y otra al control de salida del sistema. Así por ejemplo, el usuario puede decir *"voy a enseñarte un objeto"*. Esto provoca el movimiento del diálogo a un estado cuya acción asociada es enviar el comando `show` al sistema. Esto provoca un cambio en el estado del sistema que pasa a activar las capacidades visuales de atención hacia el usuario. Si esta activación no da problemas, el diálogo se mueve a otro estado cuya acción asociada es la de enviar el texto *"OK, ya estoy preparado"*.

La interacción conversacional se realiza siguiendo el objetivo de establecer un conocimiento de base común, mediante patrones tipo presentación-aceptación, así como otros *pares adyacentes* entre actos del habla que ayudan a establecer las unidades de este conocimiento de base común. Por ejemplo, el sistema avisa cuando el reconocedor del habla falla.

El sistema de reconocimiento automático del habla utiliza unas gramáticas que incorporan ciertas reglas sintácticas con las que se completan huecos de información en torno a un verbo principal. El reconocedor incorpora además el intérprete de lenguaje natural que utiliza el sistema de diálogo planificado de ATIS, denominado Phoenix (véase sección 2.2.2). Las entidades semánticas contienen dos tipos de información, una referente al contenido del discurso, y otra referente al propio discurso en sí. Ésta última se sintetiza en una serie segmentada de actos del diálogo que se van memorizando en un historial, siguiendo la propuesta de [Grosz and Sidner, 1986], que divide la estructura del diálogo en tres tipos de historiales distintos: estructura del discurso, entidades a las que prestar atención e intenciones de los interlocutores. Este historial hace más robusto el reconocimiento del diálogo ante las pausas típicas que suele haber en los enunciados del habla natural, ante la repetición de palabras o correcciones dentro del enunciado.

El sistema de diálogo combina el reconocimiento del habla espontánea con información multimodal, lo cual ayuda a robustecer dicho reconocimiento [Hüwel et al., 2006]. Gracias al sistema de atención y de seguimiento de objetos en el entorno, el sistema es capaz de interpretar enunciados con anáforas deícticas, del tipo *"esto es una planta"*. Este es el objetivo final de BIRON, fusionar correctamente percepción verbal con percepción visual.

2.3.3. Gestores basados en planificación

El Laboratorio de Tecnologías del Habla del centro de inteligencia artificial de Alemania (DFKI) ha realizado múltiples aportaciones ante temas de multimodalidad, sistemas de diálogo, navegación e interacción humano robot. Resulta interesante destacar el trabajo realizado en robots móviles que interactúan con el usuario para resolver dependencias espaciales del entorno. En [Kruijff et al., 2007] se describe una arquitectura de interacción humano robot, en el que el usuario juega un papel de tutor o guía ante el robot que aprende fusionando la información de distintos modos: visual, verbal o de sus telémetros, que le llega del entorno. La arquitectura de control está basada en un sistema *BDI (Belief, Desire and Intention)* que sirve como base para las diferentes modalidades donde comparten y fusionan la información sensorial. El diálogo se controla mediante un sistema de mediación que maneja las peticiones o necesidades comunicativas que surgen de la percepción. El sistema de reconocimiento de voz está basado en *Nuance*[6] junto con un intérprete semántico de gramáticas denominado *OpenCCG*[7]. Este intérprete se utiliza tanto para el procesamiento del habla natural como para la generación de lenguaje. El enunciado generado se sintetiza mediante un motor de síntesis denominado *Mary*[8], basado en MBROLA[9], que realiza síntesis por concatenación (sobre los distintos métodos de síntesis de voz véase sección 5.1.2).

Mediante este sistema, el robot es capaz de aprender a distinguir objetos así como a construir un mapa topológico del entorno enlazando las representaciones semánticas de los enunciados del usuario con el mundo interno del robot percibido por sus sensores.

El entorno se representa en diversos niveles: por un lado una representación que permite la navegación del robot mediante SLAM. Por otro lado, una representación ontológica siguiendo el estándar *OWL*[10] que permite realizar *razonamiento ontológico* dinámicamente. A este espacio ontológico tiene acceso tanto los sistemas de percepción como el sistema de diálogo. En él, se representan clases de objetos, instancias de estas clases y relaciones entre individuos, para cubrir así su localización relativa o inclusión. Así por ejemplo, si el robot percibe que está en una habitación y en la habitación hay una máquina de café el robot puede inferir que la habitación es una cocina.

[6] http://www.nuance.com
[7] http://openccg.sf.net
[8] http://mary.dfki.de/
[9] http://tcts.fpms.ac.be/synthesis/
[10] http://www.w3.org/TR/owl-guide

El dominio de enunciados que el sistema es capaz de percibir consta de un pequeño conjunto de comandos tipo *"ven conmigo", "vete al laboratorio", "gira a la izquierda", ...* y de enunciados descriptivos como *"estamos en la oficina", "esto es una caja"* así como cuestiones que involucran el razonamiento ontológico: *"¿dónde está la máquina de café?", "¿dónde está el laboratorio?"*. El dominio de enunciados con los que se expresa el robot se centra en responder con confirmaciones, a responder indicando lo que el usuario le solicita (lugar y objetos) o enunciados del tipo *"no lo se"*.

En [Kruijff et al., 2006] se ha desarrollado un gestor de diálogo basado en planificación que interactúa con el resto de la arquitectura comentada. El planificador de diálogo parte de lo que denominan una *"necesidad comunicativa"* que puede venir del propio flujo del diálogo en cierto instante, o de alguna percepción desde algún modo que involucre tal necesidad, por ejemplo si se encuentra alguna inconsistencia entre lo percibido y el modelo del entorno. Dicho modelo se realiza en una estructura lógica que permite relacionar campos de información de la conversación con actos comunicativos. De este modo, el planificador establece un objetivo comunicativo y construye un plan para llevarlo a cabo. El plan comunicativo se sigue mediante una percepción visual del contexto [Kruijff, 2005] y se representa mediante una estructura lógica que relaciona lo percibido por el reconocimiento de voz con lo percibido mediante visión. La gramática CCG se utiliza para generar el enunciado de solicitud de aclaración a partir de dicha estructura lógica. Por ejemplo, el usuario puede decir *"la bola roja está cerca de la caja azul."* y el sistema se ve obligado a contestar con un enunciado de confirmación. Previamente el sistema intenta verificar mediante la representación ontológica del entorno si el enunciado es cierto. Si encuentra una inconsistencia (por ejemplo, no hay ninguna "bola roja") el sistema realizaría la afirmación "No veo una bola roja".

Por tanto los sistemas de diálogo basados en planificación requieren de "terceras partes" o componentes externos al diálogo que construyan un modelo del entorno y que comparen la percepción hablada con el sistema de creencias, para planificar un enunciado de salida que intente unificar ambas representaciones. Esta planificación incluye el uso de *actos de diálogo* como preguntas, afirmaciones o respuestas.

2.4. Conclusiones

En esta sección se describen las conclusiones a las que se llega tras el estudio precedente. Se explica también qué mejoras aporta el sistema desarrollado en la presente tesis, respeto a los sistemas descritos.

Respecto a los sistemas de programación natural de un robot por el usuario final, se deduce que, por su precisión descriptiva, el lenguaje verbal resulta esencial en la especificación de comandos. Ello no implica que haya que abandonar el manejo de la percepción de gestos, pero al ser esto técnicamente más difícil de tratar e incorporando una información sensorial más vaga que el lenguaje verbal, como punto de partida puede cederse toda la atención a la interpretación del lenguaje natural verbal.

Esta interpretación verbal pasa por el uso de gramáticas, que se consolidan como los mejores elementos lingüísticos capaces de dotar del formalismo necesario para alcanzar el enlace requerido entre lenguaje y significado. En el caso de sistemas de programación natural, este significado queda reflejado en un conjunto de primitivas de comportamiento. Estas primitivas son las que se combinan para construir el programa. En todos los casos analizados en el estado del arte, este conjunto de primitivas resulta muy limitado, conteniendo a lo sumo una decena de acciones o comandos. En el sistema desarrollado, las gramáticas utilizadas son mucho más complejas. Además incorporan la posibilidad de cargarse y modificarse en tiempo de ejecución, como se verá más adelante. El dominio de acciones, condiciones y comandos interpretables por estas gramáticas triplica en tamaño al de los sistemas descritos. Además, este dominio es fácilmente escalable incluso en tiempo de ejecución.

Respecto a los sistemas de diálogo desarrollados en robótica, todavía no queda claro si resulta más flexible o funcional un paradigma de gestión basado en huecos de información o en planificación, y parece que son los objetivos de la plataforma la que hace más adecuado uno u otro. Si a priori se conocen los datos requeridos por el sistema parecería mejor un paradigma de *script* o basado en huecos de información. Si no se conocen, o el robot está orientado a resolver una tarea a través de un plan que involucra comunicación y colaboración con el usuario podría resultar mejor un paradigma basado en planificación.

En cualquier caso, todos los sistemas basados en planificación también consideran la existencia de huecos de información. La planificación simplemente aporta el cómo debe actuar el sistema de diálogo para completar esos huecos de información, para ello utiliza representaciones ontológicas del entorno o reglas de movimiento del diálogo que requieren de terceras partes

que se encarguen de construirlas. Esto, para entornos con pocos objetos (en el estado del arte nunca se nombran más de cinco) o pocos movimientos de diálogo (apenas dos o tres parejas de pares adyacentes: pregunta-respuesta, afirmación-confirmación,...) se ha implementado utilizando ontologías y operaciones con estas estructuras ontológicas, que tampoco llegan a resolver el problema del modelo del mundo de manera más general, y que presentan bastantes limitaciones.

En ningún caso se han encontrado sistemas de diálogo en robots sociales que presten especial atención a los aspectos temporales del diálogo, como es el intercambio de turnos, los ritmos de interacción, las entonaciones utilizadas... de una manera global. Estos aspectos son de suma importancia en la efectividad y empatía de la comunicación entre los interlocutores tal y como se describen en las investigaciones de comunicación no verbal [Davis, 1971], [Birdwhistell, 1970], etc. El sistema que se propone en el presente trabajo, trata de tener muy en cuenta estos aspectos temporales.

Todos los sistemas de programación natural estudiados, tanto los aplicados a robótica como los más generales, parten de un conjunto discreto, de mayor o menor tamaño, de acciones primitivas que se combinan para la construcción de comportamientos compuestos (programas o *macros*). Sin embargo, en todos los sistemas el esquema de estos comportamientos más complejos es el de una secuencia de acciones seguidas. No se incluye la posibilidad de que haya condiciones previas a las acciones que también puedan combinarse con ellas. Tampoco se ha planteado la posibilidad de construir estos comportamientos compuestos utilizando esquemas de composición más complejos que el de una secuencia de acciones seguidas. Por ejemplo, incluyendo una selección entre varias condiciones, concurrencia de varias secuencias o bucles. Estas estructuras son utilizadas en el sistema de programación natural presentado en este trabajo.

3. MAGGIE, ROBOT SOCIAL DEL ROBOTICSLAB.

Maggie ha sido concebida como un robot social que sirve de plataforma de investigación en interacción humano robot. Dentro de este capítulo se exponen los retos que plantean los robots sociales. Maggie se ha diseñado para investigar cómo resolver algunos de estos retos.

En este capítulo se realiza una descripción del robot. Se comienza describiendo cómo se realizó su diseño. Se sigue explicando la arquitectura hardware de los sensores y actuadores que le permiten interaccionar con el entorno. Y se continua profundizando en cómo se han hecho accesibles las funcionalidades de bajo nivel del robot a un nivel más alto, donde el usuario no experto pueda combinarlas y construir secuencias de un modo lo más natural posible.

3.1. Robots sociales

Muchos términos se están manejando para denominar o clasificar a los robots no industriales: robots sociales, robots sociables, robots personales, robots asistentes,... Denominamos a todos estos robots no industriales simplemente como *robots sociales*. Tal y como se define en [Bischoff and Graefe, 2004] un robot asistente personal es *"[...]una máquina versátil, que interactúa completamente de modo autónomo con su entorno y existe principalmente para el beneficio y el bien estar de los seres humanos quienes cómodamente pueden definir y asignar tareas al robot sin necesidad de realizar cambios físicos en él."*.

Ahora bien, los robots personales plantean una serie de requisitos que pueden ser analizados desde tres puntos de vista: *hardware*, problemas electromecánicos e incorporación de nuevos materiales; *software*, problemas de implementación de una arquitectura de control, y de *control*, problemas a la hora de modelar un robot autónomo, capaz de aprender, de colaborar y de interactuar con el humano de igual a igual. En este capítulo, analizamos soluciones planteadas para algunos robots sociales en la actualidad.

3.1.1. Requisitos y retos que plantean los robots sociales

Se ha realizado un esfuerzo de unificar, a partir del estado del arte, cuáles son los principales retos que plantean los robots sociales. Como se describe en [Fonga et al., 2003]), estos retos parten de la idea de tener un robot social que asista personalmente al usuario el cual pueda interactuar con el robot de un modo lo más natural posible.

En [Salichs et al., 2006] se enumeran estos retos que ahora pasamos a explicar con mayor detalle.

Interacción de igual a igual Tradicionalmente, la interacción entre el robot y el humano ha seguido un esquema de maestro-esclavo, en el que el ser humano actuaba como controlador principal del robot, teleoperándolo o enviando órdenes y supervisando que el robot realiza su tarea correctamente. En un sistema de este tipo, la autonomía del robot está limitada. El robot actuaba como una simple herramienta del usuario.

El nuevo paradigma de interacción de igual a igual sitúa al ser humano al mismo nivel que cualquier otro objeto del entorno perceptible por el robot. Y así el robot interactuará con el humano como con cualquier otro objeto del entorno. El comportamiento del robot ahora estará basado en un modelo de sus propias motivaciones e impulsos. De este modo, el usuario sentirá más cercanía hacia el robot, que será percibido como un igual.

Personalidad de un robot social. Atracción y amigabilidad La personalidad de un robot personal se refiere, por un lado, a un modelo interno del mismo que sirva como base de todo su comportamiento y procesamiento de la información, y por otro lado, a un diseño externo que favorezca la creación por parte del usuario de un modelo mental del robot. En este sentido, cabe destacar el efecto "Mago de Oz", que ocurre cuando con solo el contacto visual, el usuario crea todo un modelo mental del robot imaginando comportamientos y creando expectativas influenciadas por los propios deseos del usuario así como por los múltiples registros adquiridos del el cine, la ciencia ficción o los mas-media.

Otro efecto a tener en cuenta en el diseño del robot, o de su "personalidad externa" es el parecido humano que éste ha de tener, para maximizar la *amigabilidad* entre ambos. Existen múltiples estudios sobre el efecto en el usuario de distintas opciones de diseño del robot. Así por ejemplo, en [Kidd and Breazeal, 2004] se propone una métrica de medición de la ami-

gabilidad de un robot según su aspecto externo. Estas métricas aplicadas en la comparación de la amigabilidad de un rostro robótico sintetizado en una pantalla mediante técnicas gráficas con un rostro físico nos dicen que los usuarios prefieren un rostro físico a un rostro en una pantalla.

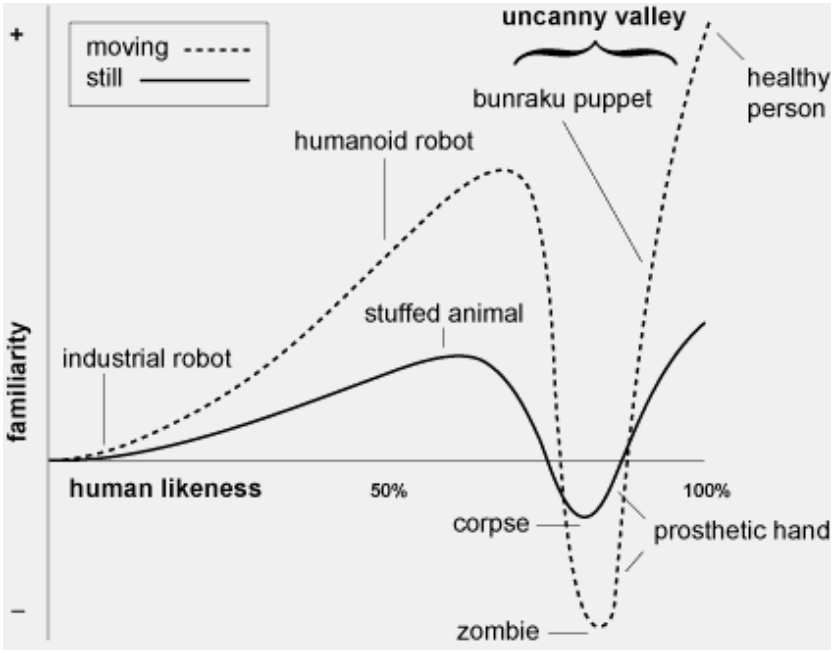


Fig. 3.1: *Representación de la amigabilidad de un humanoide en función de su parecido humano*

En [Kanda et al., 2002] se realiza un estudio de dos aspectos del diseño externo del robot: su comportamiento y su parecido humano. Respecto al comportamiento, se concluye que la amigabilidad de un robot siempre aumenta en la medida en que éste se comporta del modo más parecido posible al ser humano.

Pero respecto al parecido del robot con el ser humano, encontramos un límite en la amigabilidad del robot según su aspecto se asemeja más al de un ser humano. Se representa dicha amigabilidad por una curva donde aparece un valle conocido como el *valle siniestro o misterioso*. Este efecto fue propuesto por primera vez en [Mori, 1970] de cuyo trabajo hemos extraído la figura 3.1. En esta figura se representa la familiaridad o amigabilidad de un "humanoide" según su parecido humano. La curva comienza con los robots industriales. La amigabilidad del robot va aumentando en la medida en que se va pareciendo al ser humano hasta encontrarse con el valle siniestro. En

este valle, el autor sitúa a figuras como la del zombie, o la de un cuerpo muerto. También en el estudio que dedica S. Freud al tema de lo siniestro (1919) se afirma como tal aquello que *siendo animado se presenta como inanimado (un cuerpo muerto) o viceversa, esto es, aquello que siendo inanimado se presenta como animado (un zombie).*

Autonomía Los robots sociales deben tener un alto grado de autonomía. No solo deben ser capaces de ser autónomos energéticamente (dirigiéndose a su estación de carga cuando pertinentemente así lo perciba el propio robot), también el robot debe tomar la iniciativa interactiva para corregir o solicitar información al usuario. Y, en general, el alto grado de autonomía permite al robot personal tomar sus propias decisiones de acuerdo a sus propios objetivos internos.

La autonomía está ligada al paradigma de interacción de igual a igual, ya que para que el robot sea percibido como un "ser vivo" debe contar con un nivel de autonomía equivalente al de los seres vivos (como las mascotas, etc)

Capacidad de adaptación: aprendizaje La capacidad de aprendizaje es una característica crucial para un robot social. El robot debe ser capaz de aprender de la experiencia acumulada en su interacción previa con el entorno. El aprendizaje debe aportar al robot nuevas habilidades.

En [Klingspor et al., 1997] se distingue el aprendizaje de un robot para comunicarse con el ser humano y el aprendizaje de un robot mediante la comunicación con el ser humano, donde se incluye por ejemplo, el aprendizaje por observación. En este sentido, la interacción juega un papel crucial.

En [Breazeal, 2002] y [Breazeal et al., 2004] se realiza una enumeración de algunos retos para que un robot pueda aprender de la interacción con el ser humano, cuando este actúa como guía o "profesor" El robot debe ser capaz de reconocer cuándo su acción va resultar adecuada y cuando resulta errónea de acuerdo a la situación, y así, saber cómo corregir su acción. Además el robot debe tener la habilidad de explorar adecuadamente el entorno para aprender de él.

Cooperación: multimodalidad, iniciativa propia y capacidad de reacción El robot personal social puede cooperar con otros robots o con seres humanos para llevar a cabo un objetivo que puede ser individual o colectivo. Este objetivo puede o no estar descrito explícitamente, dependiendo de la arquitectura de control del robot.

En esta cooperación de nuevo las habilidades de interacción juegan un papel esencial. Para llevar a cabo una interacción eficiente esta debe contar con tres elementos fundamentales: multimodalidad, reactividad y "proactividad."

La multimodalidad está relacionada con la interacción de igual a igual. La interacción natural del ser humano con su entorno es multimodal, es decir, involucra todos los sentidos tanto en la percepción del entorno como en la articulación del acto comunicativo. El robot debe ser capaz de fusionar y fisionar información del tacto, los gestos faciales y corporales, y la comunicación verbal.

La reactividad se refiere a la capacidad de respuesta del robot social en el momento y en la manera adecuada a los cambios del entorno. Así por ejemplo, cuando esté interactuando con el ser humano, el robot debe ser capaz de darse cuenta si se ha perdido la coherencia de la interacción, si se está tratando con información que alguna de las partes carece, etc.

Pero el robot personal no solo debe ser capaz de reaccionar adecuadamente ante cambios en el entorno, sino que, debe ser "proactivo", es decir, mostrar un comportamiento dirigido a un objetivo concreto y tomar la iniciativa en el momento adecuado para o bien corregir al ser humano, ofrecer información o sugerencias que este necesite, etc.

Expresividad Se ha hablado de la necesidad de una comunicación eficiente entre el humano y el robot social. Las capacidades expresivas del robot son fundamentales para esta comunicación eficiente. La multimodalidad no solo está referida a la percepción del acto comunicativo del ser humano, sino también a la articulación del robot mediante gestos acompasados correctamente con la comunicación verbal.

Estos gestos no solo se limitan a la expresión no verbal del cuerpo robótico, sino que en la medida en que el robot es una máquina electrónica también puede incluir nuevos modos de expresión que no contempla el ser humano pero que si son sensibles por éste. El robot puede contar con miembros exclusivos como antenas, cola (como un animal) o bien con una pantalla o proyección que le permita expresarse mediante imágenes en movimiento figurativas o abstractas.

Análogamente, desde el punto de vista sonoro, el robot no sólo puede expresar mediante el control prosódico información no verbal en su comunicación hablada, sino que también puede incluir sonidos "figurativos o abstractos", música, etc.

3.1.2. Algunos robots sociales

Los primeros robots personales que se han popularizado en el mercado están diseñados para entretenimiento y educación. Aunque en Mayo de 2006, Sony anunció la interrupción de su producción[1], **Aibo** [Sony, 2007] cuenta con diversos modelos distintos todos ellos diseñados siguiendo la forma de un perro - robot. Sus características cambian de modelo a modelo pero todos mantienen un cuidado diseño amigable. Aibo viene, además, con diversos elementos adicionales como es, una pelota, un hueso así como con diversos paquetes software para programar al robot.

Aibo cuenta con cámara, sensores táctiles y micrófono. El movimiento preciso de sus cuatro patas desde el punto de vista de control, está muy logrado ya que, junto con los movimientos de su cabeza, le permite grandes posibilidades expresivas, así como de translación. Aibo incorpora además altavoces con los que se expresa mediante sonidos. Todo ello le hace capaz de expresar emociones e incluso, un lenguaje de signos no verbal propio.

Los detalles de su arquitectura no son muy accesibles, algo muy común en todos los robots comerciales. Sin embargo una larga convivencia con el perro-robot nos llevan a sacar sendas conclusiones. Por un lado, Aibo muestra un comportamiento muchas veces aleatorio como son saludos al aire, navegación exploratoria, llamadas de atención, bailes y canturreo, etc. Por otra parte es capaz de atraer la atención mediante la expresión de emociones haciendo lloriqueos de tristeza que se le pasan cuando se le acaricia, o haciendo movimientos de euforia esperando a que aparezca la pelota. La comunicación por voz no es eficiente, pues es rara la vez que el robot hace caso a lo que se le dice. La capacidad de aprendizaje tampoco es muy eficiente, pues aunque el robot viene con un paquete software para enseñarle una secuencia de acciones, en los diversos experimentos que se ha llevado a cabo con el robot, nunca se ha conseguido algo parecido. Tampoco queda claro si la evolución en los comportamientos del perro-robot son adaptaciones al medio y están influenciadas por la interacción con el usuario o más bien, y esto es lo que parece, se trata de una evolución determinada fija e inamovible. Un aspecto que hace muy interesante a Aibo y a muchos otros robots comerciales son los interfaces de programación que incluyen. El sistema operativo que controla al robot (OPEN-R) está muy bien documentado y diversos paquetes de software libre ya han sido desarrollados como "Tekkotsu" en la Carneige Mellon University, creando una gran comunidad científico-pedagógico que utiliza al robot con fines educacionales. Existe también una versión humanoide de Aibo: **Qrio**.

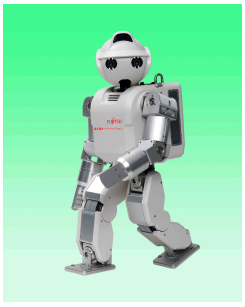
[1] Consulta realizada en 2010 en el sitio http://support.sony-europe.com/aibo/index.asp

(a) AIBO por SONY (Japón)

(b) QRIO de SONY (Japón)

(c) iCat de PHILLIPS (Alemania)

(d) MARON de FUJITSU (Japón)



(e) HOAP de FUJITSU (Japón)

(f) NECORO de OMROM (Japón)

(g) PAPERO de NEC (Japón)

(h) ROOMBA de iROBOT (EEUU)

Fig. 3.2: *Algunos robots sociales comerciales*

Diseñado con forma de gato-robot es **NeCoRo** [Omrom, 2007] (1500 € aprox.) El robot incorpora pelo y una serie de luces que aumentan su capacidad de expresión. No es móvil, pero es capaz de realizar gestos para expresar su estado afectivo.

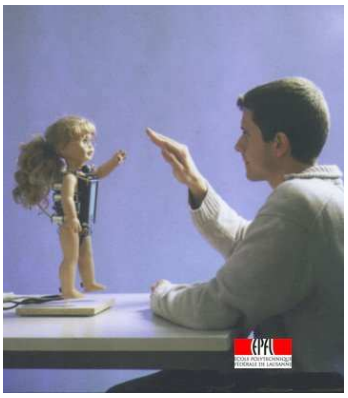
Otro robot del estilo a NeCoRo es **Paro**, un robot con forma de foca de peluche [Shibata, 2007] con fines terapéuticos con personas mayores. Es capaz de mover su cola, su cabeza y sus ojos para expresarse. También puede emitir sonidos parecidos a los de una foca-bebé. Reacciona ante sonidos y cuando se activan sus sensores táctiles. Este robot incorpora un concepto interesante que es el de mantener un ciclo vital: por la noche el robot se duerme.

Otra plataforma interesante muy utilizada para la investigación en interacción humano - robot, es **iCat** [HomeLab, 2007] desarrollado por Phillips en Amsterdam (Holanda). Se trata de un robot personal, torso con forma de gato, no móvil, de 38 cm de alto, cuyo propósito principal, además del de plataforma de investigación, es el de robot compañero, asistente como interfaz al mundo digital al realizar tareas como lectura del correo electrónico, acceso a datos por web, etc. iCat incorpora múltiples GDL (13 servos) de movimientos expresivos de cabeza, ojos, cejas, párpados y labios, con lo que es capaz de generar múltiples expresiones emotivas: alegría, tristeza, enfado, sorpresa,... iCat cuenta con una cámara que le permite el reconocimiento de caras y objetos. Cuenta, también, con dos micrófonos que le permiten reconocimiento de habla identificando, además, la dirección de la fuente acústica. iCat es capaz de expresarse también mediante sonidos y voz. Además, cuenta con una serie de LEDS que le permiten comunicarse de acuerdo a un código de luces.

Robota fue diseñado en al Autonomous System Laboratory del EPFL (Suiza) [Billard et al., 2006] con la intención de estudiar cómo puede aprender un robot imitando al ser humano. Se trata de un robot tremendamente sencillo cuya aplicación es triple: estudio del aprendizaje por imitación, desarrollo de un juguete educacional y desarrollo de una herramienta terapéutica. Se encuentra un modelo de este robot en el museo de Ciencias de Francés en Toulouse.

El robot está construido sobre una muñeca bien conocida, a la que se incorporan diversos servomotores para mover los brazos (1 GDL), las piernas (1 GDL), el cuello (1 GDL), los ojos (2 GDL) y los párpados (1 GDL). Robota es capaz de sintetizar voz. El sistema además incorpora una cámara que es capaz de detectar movimientos de los brazos del usuario.
La arquitectura de control es puramente reactiva: el robot detecta mediante

(a) Robota, el robot muñeco



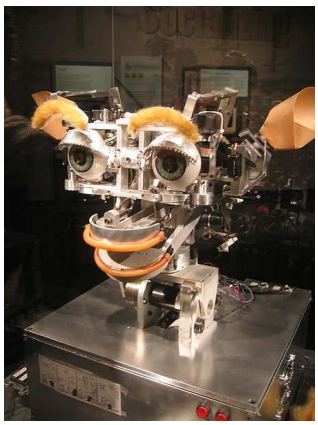
(b) Robota interactuando por imitación

Fig. 3.3: *Robota, robot muñeca desarrollado por EPFL*

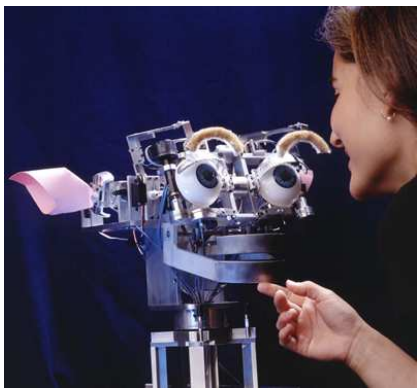
visión o potenciómetros conectados a la articulación del usuario, el movimiento de sus brazos, piernas y cuello, utilizando Modelos Ocultos de Markov adapta dichos movimientos a su espacio de articulación libre y ejecuta los movimientos adecuados [Calinon and Billard, 2004]. A pesar de la sencillez de esta arquitectura, el robot cuenta con dos ventajas fundamentales que le hacen cumplir el propósito de su diseño a la perfección: por un lado el propio diseño, por otro la alta capacidad de reacción y su intensidad activa. Cuando se ha utilizado en terapia con niños autistas éstos han reaccionado muy positivamente ante el robot. Hay que comprender que el autismo es una enfermedad que aisla al individuo de la realidad debido a que es incapaz de procesar la información que le llega debido a su complejidad. Es por ello que al niño autista le suele gustar el movimiento sencillo y repetitivo de las cosas. Quizás esto explique en parte su buena reacción ante Robota. Este sencillo robot ha servido como herramienta para estudiar el autismo. Las investigaciones en este sentido, evalúan el contacto visual entre el niño y el robot, grado de imitación del niño a los movimientos del robot y del grado de acercamiento y presencia del niño ante el robot [Robins et al., 2003]

Kismet[Breazeal and Scasselatti, 2000] y **Leonardo** [Breazeal et al., 2004] basan su arquitectura de control en un modelo de las emociones. La principal aportación a las arquitecturas de control de robots de este tipo es la incorpo-

ración de variables homeostáticas simulando las motivaciones o necesidades de los humanos.
Kismet es una cabeza antropomórfica con 3 GDL para cada ojo, otros 3 GDL para el movimiento del cuello, y 15 GDL en total para mover el resto de la cara: párpados, cejas, labios y orejas. Por tanto se trata de un robot centrado en la expresión por medio de gestos faciales. También incorpora un sintetizador de voz sin control de entonación. Su sistema perceptivo consta de dos cámaras en los ojos y micrófono, de modo que es capaz de detectar ciertos patrones visuales y otros auditivos. No incorpora reconocimiento de voz.



(a) Kismet, el cara robótica para expresión de emociones



(b) Kismet interactuando

Kismet está inspirado en el comportamiento de los niños muy pequeños cuando interactúan con adultos. El sistema visual es capaz de detectar movimiento de caras y "cosas" de colores (como juguetes). El sistema auditivo está diseñado para clasificar la entonación del habla del humano y clasificarla según su intencionalidad o sentido afectivo. Toda esta información entra en una arquitectura actuando en tres variables homeostáticas que representan tres motivaciones humanas: social, estimulante y fatiga. El estado de estas tres variables es comparado con un estado de emoción instantáneo a modo de realimentación, provocando un nuevo estado emotivo al cual va asociado un comportamiento motor determinado. Estos comportamientos no solo fijan la posición de los distintos grados de libertad de la cara robótica, si no que atienden también a su evolución temporal.
Kismet demuestra la importancia de la sincronización y de la dinámica temporal en la interacción y en la comunicación. Incorpora un alto grado de expresividad motora y la arquitectura basada en variables homeostáticas per-

mite un control más natural de esta expresividad. Sin embargo no deja de ser una arquitectura reactiva, en la que para un patrón determinado de entradas acústico-visuales corresponde un patrón de expresión también determinado. De nuevo encontramos la limitación ante el aprendizaje, la adaptación y la evolución en el proceso interactivo.

(c) Leonardo, robot social que interactúa mediante gestos

(d) Leonardo con su piel de peluche

Fig. 3.4: *Principales robots sociales desarrollados por el grupo Robotics Life del MIT*

Leonardo [Breazeal et al., 2004] se presenta como una nueva aplicación de la arquitectura de Kismet basada en variables homeostáticas. Esta vez, el objeto de estudio es la inferencia por parte del robot de un objetivo dentro de la interacción con un humano, a través de la percepción de ciertos patrones no verbales y de la expresión gestual.
Tanto Kismet como Leonardo mueven sus distintos GDL de acuerdo a un algoritmo reactivo que sobrepesa con parámetros homeostáticos internos al robot la posición instantánea del GDL en cuestión. Resulta interesante que sin un nivel deliberativo, sin la representación de las acciones del robot con un plan y sin la representación explícita del entorno y del usuario, el robot de la impresión a éste de que entiende el contexto, al reaccionar adecuadamente a las emociones del usuario y a sus intenciones. Esto muestra la importancia de tener en cuenta tanto la expresión afectiva del usuario en el proceso de interacción como de la intención de sus movimientos. Aquí se hace de modo implícito y eso conlleva la gran limitación ante la capacidad de adaptación, de aprendizaje y de la interesante posibilidad de la invención deliberativa de una nueva acción por parte del robot inesperada para el usuario.

Valerie [Gockley et al., 2005] es el resultado de distintas propuesta de robot social de servicio. El propósito de investigadores como Judi Forlizzi o Adam Schultz era el de construir un robot recepcionista que incorporase toda una personalidad, así como capacidades expresivas emocionales.

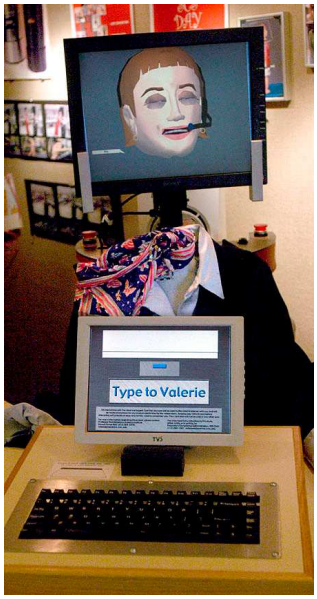


Fig. 3.5: *Valerie, robot recepcionista desarrollada por la CMU*

Se trata de una pantalla situada encima de la clásica base B21 de iRobot. En la pantalla aparece la imagen de una cara humana que habla y gesticula. El robot es capaz de percibir al usuario por medio de un teclado y una pantalla en el mostrador del hotel.

La diferencia en la eficiencia de la interacción humano - robot entre un agente animado y un agente real ha sido estudiada en [Kidd and Breazeal, 2004] cuyos resultados nos hacen concluir que claramente la gente tiene mucha más preferencia y atracción por un agente mecánico que por una animación.

Cabe destacar la clasificación de términos relacionados con las emociones realizada por el grupo de investigación que trabaja con Valerie, distinguiendo entre afecto, emoción y humor y sus consecuencias distintas en la dinámica de las acciones del robot, así como en las acciones mismas.

Hermes ha sido implementado por el Intelligent Robots Laboratory en la Universidad de Bundeswehr (Alemania) en colaboración con el Intelligent Systems Institute del National Institute of Advanced, Industrial Science and Technology (Japón)[Bischoff and Graefe, 2004]. Se trata de un robot huma-

Fig. 3.6: *HERMES desarrollado en la Universidad de Bundeswehr*

noide diseñado para estudiar las principales tecnologías importantes para el desarrollo de robots personales: diseño, percepción, locomoción, localización, navegación, manipulación, comunicación e interacción humano - robot, adaptación y aprendizaje, arquitectura del sistema e integración.
Fue probado y utilizado en un museo, donde los usuarios podían charlar en inglés, alemán o francés, haciéndoles preguntas y donde estuvo realizando servicios a petición del usuario. El robot cuenta con un diseño más mecánico que amigable manteniendo la personalidad de máquina. Físicamente parecido a Robovie [Ishiguro et al., 2002] pero de 1.85m de altura y 250 kg de peso. Cuenta con numerosos GDL tanto en brazos, manos, cabeza, ojos-cámara, así como capacidad de movimiento proxémico mediante una base con ruedas. Cuenta además con múltiples sensores de los que cabe destacar su visión en estéreo y una red de microsensores táctiles en su base, sensibles a la presión, situados en la base, que le permiten captar el terreno mediante tacto. Cuenta además con un interfaz de voz (síntesis y reconocimiento) y manejo de diálogos con resolución de pronombres y huecos de información. Hermes contiene también un interfaz remoto que le permite enviar y recibir e-mails así como recibir órdenes por teclado remoto y expresarse a una pantalla remota.
La arquitectura de Hermes es una arquitectura híbrida cuya unidad principal es la habilidad. La habilidad se define como un módulo de ejecución concreto, que corresponde con la parte más reactiva o automática de la arquitectura. La parte más cognitiva o deliberativa corresponde al manejo de las distintas habilidades del robot, así como análisis de la situación (tanto interna del

robot como externa percibida) para comparar diferentes posibilidades para cambiar la situación en cierta manera. El control final del robot se realiza, por tanto, mediante un módulo de interpretación de la situación, la cual está compuesta por objetivos internos (evitar obstáculo, ...) objetivos externos (orden del usuario,...) y modelo del mundo (mapa topológico con información adicional,...) Este módulo elige qué habilidad llevar acabo para cumplir los objetivos, mediante un autómata de estados finitos fijo. Su arquitectura le permite cierto aprendizaje, al memorizar información del entorno (como serían los atributos en un mapa topológico).

La funcionalidad global de este robot es singular, por el hecho de incorporar tantas y tan difíciles temas principales como son visión, interacción multimodo, manipulación y navegación todo ello en una arquitectura híbrida, esto es, que incluye habilidades de planificación.

No obstante, el diseño del robot resulta poco amigable. El propio autor reconoce el respeto que infundía a la gente en las pruebas que hicieron en el museo. El grado de autonomía no es muy alto, teniendo en cuenta que el robot no es capaz de tomar sus propias decisiones, sino que espera a que un usuario le de órdenes. Tampoco se ha profundizado en el desarrollo de una personalidad para el robot. Su grado de adaptabilidad o aprendizaje está muy limitado por el hecho de estar controlado por una máquina de estados finitos fija. Ello, así mismo, impide una capacidad de aprendizaje de habilidades nuevas siendo obligatorio la programación de las mismas por parte del desarrollador, todo ello a pesar de que el robot es capaz de memorizar y actualizar datos que usan su repertorio de habilidades (lista de nombres de personas conocidas, mapa del entorno, etc). La falta de capacidades de aprendizaje desfavorecen la adaptabilidad y, por tanto, la capacidad de cooperación del robot.

3.2. Maggie: aspectos previos en el diseño

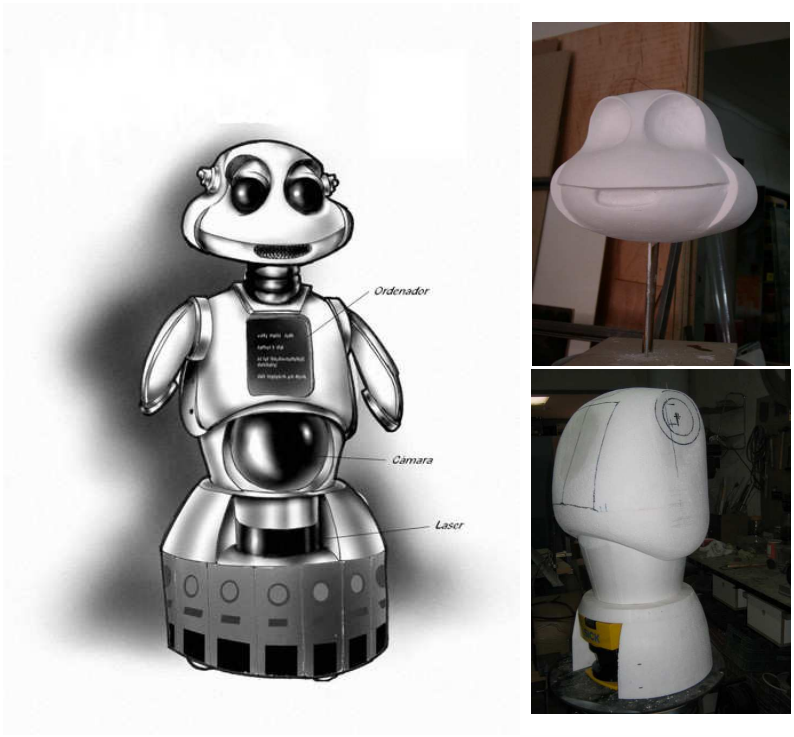


Fig. 3.7: *Detalles del diseño y construcción de la carcasa de Maggie*

El diseño de un robot personal tiene una gran influencia en la relación que se establece entre el usuario y el robot. Ocurre así que mientras el usuario no establece una completa empatía con robots cuyo diseño es todavía demasiado industrial (Hermes, Robovie, BIRON...) si que lo hace con robots como Robota, cuyo aspecto es el de una muñeca. Esto ocurre independientemente de la complejidad de la arquitectura de control del robot, o de sus habilidades de interacción. Esta importancia en el diseño queda bien reflejada al analizar gran parte de los robots personales comerciales, en los que el diseño está tremendamente cuidado.

Para realizar el diseño de Maggie, se realizó un estudio de los distintos diseños de robots sociales en la actualidad. Es la tendencia oriental quien ha perfilado las líneas estéticas de diseño externo del robot. Buscamos un robot que sea como un juguete pero sea complejo, antropomórfico pero amigable, sencillo pero versátil. Además tuvimos que tener en cuenta ciertos elementos *hardware* que queríamos incluir en el robot, como era un telémetro láser, un

tablet PC, una cámara, altavoces, etc.

Uno de los problemas que surgen a la hora de diseñar el aspecto externo de un robot personal es hasta qué punto hacerlo parecido o no al ser humano, es decir, hasta qué punto el diseño debe seguir una tendencia *figurativa* o *abstracta*. Por un lado, ya se ha expuesto el efecto de "valle misterioso" en la figura 3.1, que indica que el robot es amigable cuando es figurativo e igual al ser humano, pero "no demasiado igual". Por otro lado, sabemos por diversos estudios [2] que el ser humano no necesita tener enfrente un objeto figurativo para sentir cierta empatía con él. Además existen ejemplos de otros robots que siguen un diseño no figurativo [3], pero en los que sí hay cierta comunicación emocional eficiente entre el robot y el humano. Todo ello nos llevó a considerar un diseño como el que se muestra en la figura 3.7, prototipo que se escogió de entre otras propuestas menos figurativas. También se muestran los primeros moldes para la construcción de la carcasa del robot.

3.3. Arquitectura hardware

Para tratar los distintos retos a los que se enfrenta un robot personal, éste debe contar con un repertorio de sensores y actuadores suficientes y suficientemente eficientes como para percibir el entorno correctamente y así poder navegar por él, interactuar con los distintos objetos que se encuentren y poder interactuar con el usuario del modo más natural posible.

En la figura 3.8 se ha representado una descripción global de la arquitectura hardware de Maggie. En ella se incluyen los distintos sensores y actuadores incorporados. Todos ellos, son controlados por un ordenador interno que accede a los distintos dispositivos. Además, el robot cuenta con una conexión wireless que le permite tanto acceder a internet, como a ser accesible por otra computadora externa.

El robot cuenta con sensores en los modos visual, táctil, sonoro y telemétrico. Alrededor de la "boca" del robot, hay una **cámara web** que le permite detectar caras y ciertos objetos como son balizas visuales, o un tablero para el juego de tres en raya.

A lo largo de toda la carcasa del robot, se han colocado distintos sensores de tacto capacitivos. Su funcionamiento es por presencia de materiales como la piel humana, pero para su activación no es necesario el contacto directo.

[2] Una defensa de la expresión abstracta como una articulación más profunda, más cercana y más directa con el ser humano puede verse en [Kandinsky, 1912a] y [Kandinsky, 1912b]

[3] Sirva como ejemplo, entre otros muchos, el robot de ficción **R2D2** del famoso largometraje "La Guerra de las galaxias" (Geroge Lucas, 1977)

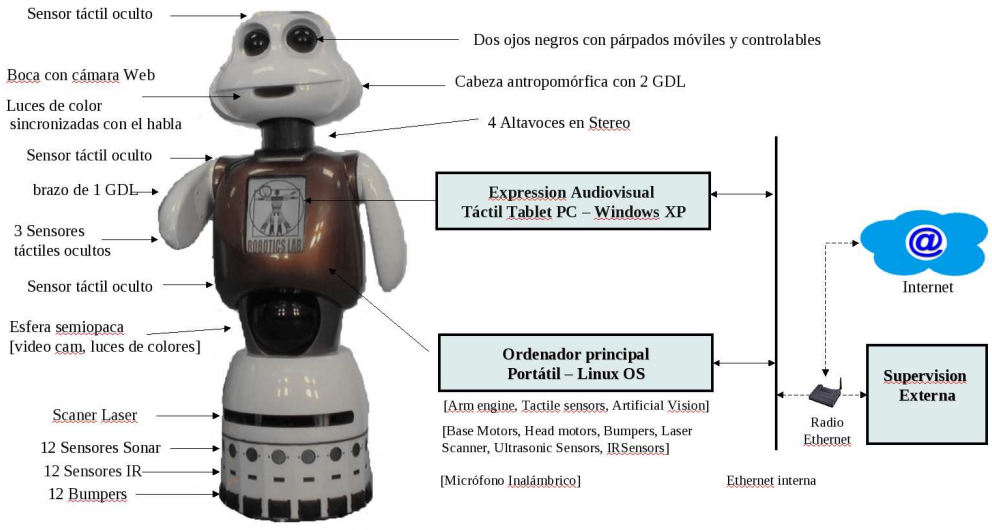


Fig. 3.8: *Arquitectura hardware de Maggie*

Por tanto, se han colocado por dentro de la carcasa, lo que los hace invisibles. Cada sensor capacitivo puede estar en dos estados, activado o desactivado según tenga cerca o no, la piel humana.

Maggie cuenta con un sensor de tacto en la cabeza, en cada hombro, a cada lado de la espalda, en cada costado y tres en cada "mano". Éstas cuentan con tres hendiduras que, al tener cada una su respectivo sensor de tacto, vienen a funcionar como botones.

Un telémetro láser en la base del robot le permite detectar el entorno: obstáculos, puertas, paredes, e incluso obstáculos móviles como las piernas de una persona. Con el mismo propósito cuenta con sensores SONAR que complementan la percepción.

Para hablar con el robot este cuenta con un micrófono inalámbrico. Dado que este micrófono es colocado muy cerca de la boca del usuario, la voz que le llega al robot es más inmune a ruidos del entorno.

Maggie puede navegar por el entorno gracias a dos ruedas independientes que le permiten trasladarse y rotar. Además, Maggie cuenta con dos brazos, de un solo grado de libertad cada uno, que, si bien no están concebidos para realizar tareas complejas como agarrar un objeto, permiten que el robot pueda tener cierta expresividad no verbal, por ejemplo, señalar algún objeto o alguna zona del espacio, saludar con la mano, etc.

El cuello del robot cuenta con dos grados de libertad: uno en movimiento horizontal y otro en vertical. Estos movimientos, por un lado, colaboran a que Maggie tenga mayor expresividad no verbal. Así, puede realizar afirmaciones, negaciones, mirar al usuario cuando este habla, etc. Por otro lado, al tener la cámara web en la boca, los movimientos del cuello también sirven para apuntar hacia dónde quiere enfocar la visión.

Los ojos de Maggie son puramente expresivos. Cuentan con unos párpados, que suben y bajan, permitiendo el parpadeo, o el guiño, y que esconden más o menos los ojos dando así mayor expresividad no verbal al robot.

Maggie es capaz de hablar y emitir sonidos. Los altavoces están colocados en el cuello del robot. En sincronía con el sonido Maggie cuenta con una matriz de LEDs azules en la boca que aumentan su expresividad.

3.4. Arquitectura de control

La arquitectura que controla al robot es una arquitectura híbrida basada en dos niveles de funcionamiento: uno deliberativo y otro automático. Por esta razón, se ha venido a denominar arquitectura automática-deliberativa o *arquitectura-AD*. Su descripción se ha realizado ampliamente en varios trabajos ([Barber and Salichs, 2001], [Barber, 2001], [Malfaz and Salichs, 2004], [Gorostiza et al., 2006] o [Salichs et al., 2006]) Así que en este apartado nos limitamos a realizar un breve resumen o una interpretación de las características más importantes de la arquitectura relacionadas directamente con la implementación realizada en el presente trabajo.

3.4.1. Conceptos de habilidad y de secuencia

Una habilidad es un módulo de ejecución independiente que se encarga de que el robot realice una acción, un comportamiento o una tarea concreta. Cada habilidad puede comunicarse con otras mediante envío y recepción de comandos, mediante envío y recepción de eventos, y mediante lectura y escritura en un sistema de memoria compartida.

En esta tesis se han desarrollado cuatro habilidades principales, que serán explicadas en los siguientes capítulos: una habilidad de reconocimiento automático del habla, otra de síntesis de voz, otra de diálogo y una habilidad de edición verbal de una secuencia.

En la arquitectura existen habilidades muy sencillas encargadas de comunicarse con el hardware del robot. Por ejemplo la habilidad de tacto se encarga de la lectura de los sensores capacitivos del robot, etc. También hay

habilidades más complejas, como por ejemplo, la habilidad "seguir a una persona" que involucra lecturas del telémetro láser y acceso a los motores de la base del robot, a los motores de sus brazos, etc.

Las habilidades pueden combinarse entre sí, de modo que una habilidad contenga a otras y así sucesivamente. Por ejemplo, una habilidad *saludar* podría constar de las habilidades: *hacer gestos* y *sintetizar voz*.

Las habilidades también pueden secuenciarse en una estructura denominada *secuencia*. Existen diversos modos de representación de una secuencia. Por su versatilidad a la vez que sencillez en la representación, en este trabajo se ha escogido el *diagrama funcional*, o *SFC (Sequence Function Chart)*. La especificación de un diagrama funcional se ha estandarizado en [I.E.C., 1993]. De este documento se han extraído las ideas más importantes relacionadas con el presente trabajo, que se han expuesto en la sección B.3.

Mediante un diagrama SFC o simplemente, una secuencia, pueden ejecutarse varias habilidades a la vez, crearse ramas de selección o, simplemente, ejecutar una habilidad seguida de otra.

La secuencia se implementa mediante una estructura en `XML` que incorpora funciones en lenguaje `Python`. Esta implementación será detallada más adelante.

3.4.2. Comunicación en la arquitectura: eventos y memoria compartida

La comunicación entre habilidades se ha dividido de acuerdo a dos aspectos, uno centrado en el aspecto temporal del mensaje que se intercambia, y otro centrado en su propio dato.

Del primero surge el sistema de comunicación mediante eventos. En este caso, lo que más interesa del mensaje es el momento en el que éste se produce. Por ejemplo, si el usuario toca al robot en uno de sus sensores de tacto, interesa que la arquitectura de control comience a responder y a atender a este evento físico en ese mismo momento. El sistema de eventos es, por tanto, discreto y finito. Cada habilidad puede enviar o recibir eventos de distinta índole. La recepción de un evento implica su gestión. Esto se realiza mediante la ejecución de una función específica que maneja la llegada de un evento concreto. Así, el sistema de eventos sigue el patrón *publicador-subscriptor*. Además es un sistema desacoplado, es decir, que las habilidades que reciben y manejan eventos desconocen a la habilidad que generó el evento. Y viceversa: las habilidades que envían un evento desconocen qué habilidades van a recibirlo. Para conseguir este desacoplo evidentemente, se hace necesario un

intermediario que gestione todo el flujo de eventos. Esto se ha implementado mediante un servidor central.

El funcionamiento del sistema de eventos es como sigue:

1. **Subscripción:** una habilidad que se va a encargar de manejar un tipo de evento concreto realiza una subscripción al servidor central. La subscripción implica asociar una función manejadora al tipo de evento en cuestión.

2. **Publicación:** otra habilidad emite un evento de tal tipo.

3. **Repartición** del evento enviado a todos los subscriptores. El servidor central recoge el envío y se encarga de que llegue a todos los subscriptores.

4. **Ejecución** de la función manejadora en cada subscriptor.

Por otro lado, está el sistema de memoria compartida. Este sistema sigue el patrón de pizarra: cualquier habilidad puede escribir en ella un dato, que puede ser leído por otras habilidades. Este sistema también está desacoplado, por tanto, también cuenta con un servidor central.

Cada dato de la memoria compartida viene definido por tres características:

- **Identificador**, que se utiliza en el proceso de lectura o escritura.
- **Formato**, indica la estructura y los tipos de valores del dato.
- **Tamaño**. Según la estructura del dato así es su tamaño en bits. El tamaño siempre debe ser limitado.

Una vez un dato es escrito en la memoria compartida, puede ser leído tantas veces como fuera necesario. El tamaño de la memoria viene limitado por la memoria del sistema en el que se implementa.

3.5. Acciones y condiciones en Maggie

En el presente trabajo se considera la posibilidad de que el robot interactúe con el entorno siguiendo un plan explícito representado por una *secuencia*, que se define del siguiente modo:

Definicion 3.5.1. ***Secuencia:*** *red cuyos nodos los forman acciones y condiciones de manera alternada.*

Definicion 3.5.2. ***Acción:*** *cualquier proceso o efecto de la actividad del robot con su entorno.*

Definicion 3.5.3. ***Condición:*** *una entidad funcional que puede tomar dos valores, verdadero o falso, según ciertas variables del robot o del entorno, donde se incluye al usuario.*

Cabe destacar, por tanto, que el concepto de secuencia aquí manejado, no solamente establece una serie de acciones a ejecutar siguiendo cierto orden esquemático, sino que en la secuencia también se incluyen condiciones que tienen que ver con el entorno, por tanto, la secuencia no establece simplemente un modo de acción en el entorno, sino un modo de *interacción* con el entorno.

En la edición de una secuencia de acciones y condiciones en un robot por parte de un usuario no experto, surge el problema de cómo representar dichas acciones y condiciones para poderlas hacer accesibles de un modo lo más natural posible al usuario. Hoy por hoy, las acciones y las condiciones del robot se implementan a bajo nivel en un software construido en un lenguaje de programación especial creado por desarrolladores. Nuestro propósito es intentar acercar este nivel a un usuario que interacciona con el robot a alto nivel, de modo que la implementación de bajo nivel le resulte transparente.

En esta sección, se presenta una nueva metodología de diseño en este sentido. Es decir, *cómo representar las acciones y las condiciones en un robot para que puedan ser directamente accesibles al usuario mediante interacción natural con el robot.*

El acceso a los sensores y actuadores del robot se ha implementado de dos modos distintos. En un primer modo, cada acción y cada condición está definido por una única función junto con los parámetros pertinentes. Así, el dominio de acciones y condiciones queda implementado *a priori* en dos módulos externos `actions.py` y `conditions.py`. Siguiendo este modelo se ha realizado el editor verbal *no interactivo* de secuencias, explicado en la sección 6.

En una segunda implementación el acceso a los sensores y actuadores del robot, se realiza utilizando objetos, lo cual permite mayor versatilidad al acceso de las propiedades de cada sensor y actuador, y mayor generalidad a la hora de ampliar los dominios de acciones y condiciones posibles. Siguiendo este modelo se ha realizado el editor *interactivo* de secuencias, explicado en el capítulo 8.

3.5.1. Dominio de acciones del robot

Tal y como se define acción dentro de nuestro sistema podemos contar con acciones atómicas indivisibles, como por ejemplo la acción "parpadear" dentro de la habilidad encargada de mover los párpados o acciones que involucren la coordinación de varias habilidades, como por ejemplo la acción "saludar" que involucraría tanto la habilidad de habla como las de movimiento de los brazos, movimiento de la cabeza, etc. Es decir, cada acción tiene asociada una semántica que corresponde a lo que el robot hace al ejecutar dicha acción.

Ahora bien, la incorporación de una acción en forma de etapa dentro de una secuencia implica cuatro requisitos necesarios, que están asociados a cómo debe representarse la acción dentro de nuestra arquitectura. El primer requisito está relacionado con el modo en que la acción es referida mediante voz por el usuario. Así, toda acción debe estar asociada a algún elemento semántico dentro de la gramática del reconocedor automático del habla.

Los otros tres requisitos, están relacionados con el modo en que la acción, que está dentro de una secuencia, va a ser ejecutada en el sistema. Esta ejecución se realiza mediante el intérprete de la secuencia. Cada acción debe contar con un mecanismo de activación, con un mecanismo de finalización y con un mecanismo de información de su estado de ejecución.

1. **Mecanismo de activación.** Se trata de un mecanismo que utiliza el intérprete de la secuencia para comenzar a ejecutar la acción cuando la etapa a la que va ligada pasa a modo activo. Típicamente la acción se activará mediante una llamada a una función. La ejecución de la acción corresponderá así con la ejecución de una función.

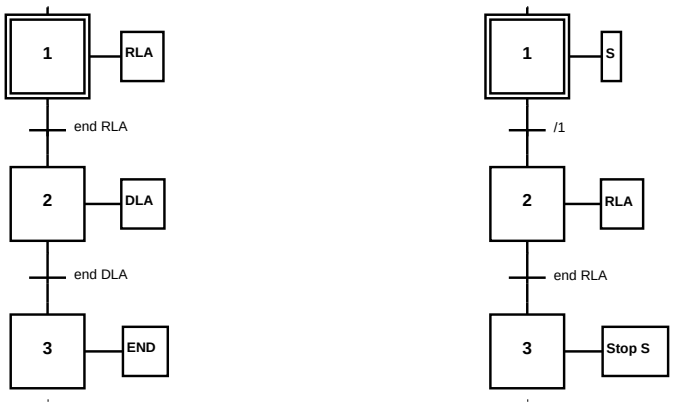
2. **Mecanismo de finalización.** Es el mecanismo que utiliza el intérprete de la secuencia para finalizar la ejecución de una acción que está en marcha. Por ejemplo, la acción "girar" puede no tener un final por sí misma, y el robot se pone a girar sin límite hasta que explícitamente se especifica "parar de girar".

3. **Mecanismo de información del estado de ejecución**. De este modo es posible saber si la acción ha finalizado. Mediante este mecanismo, cualquier condición dentro de una secuencia puede utilizar la información respecto al estado de finalización o no del resto de acciones de la secuencia. De este modo se pueden construir condiciones que dependan de la finalización de una acción, por ejemplo la condición "cuando termina de subir el brazo izquierdo", asociada a la finalización de la acción "subir brazo izquierdo".

Acciones basadas en llamadas a funciones

Cada acción del sistema está representada por una función `Python` y un conjunto de parámetros. Así por ejemplo, tendríamos la funciones `rise( body,side )` y `down( body, side )` para, respectivamente, levantar o bajar una parte del cuerpo, los brazos, la cabeza o los párpados.

Podemos dividir las acciones en dos tipos: las que finalizan y las que no finalizan por sí solas. En el caso de `rise` o `down` se trata de funciones con un final claro.



(a) Dos acciones (*RLA y DLA*) con finalización propia.

(b) La acción *S* finaliza con otra acción posterior.

Fig. 3.9: *Representación de acciones con y sin final dentro de una secuencia simple*

Las acciones con final tienen asociada una función de "comprobación de terminación" que sirve como mecanismo de comprobación del estado de ejecución de la acción. Para cada acción que no finaliza por sí misma, existe otra acción que la hace finalizar. Por ejemplo, la acción "girar sobre si misma" `spin()` no finaliza, pero tiene asociada otra acción "parar de girar" `stopSpin()` que la hace finalizar.

En la figura 3.9 puede verse un ejemplo de dos secuencias simples donde aparecen acciones con final y sin un final. En 3.9(a) la acción "levantar brazo izquierdo" (`RLA`) es sucedida de "bajar brazo izquierdo" (`DLA`) siempre y cuando la primera acción haya finalizado, lo cual se comprueba mediante la condición "fin de levantar brazo izquierdo". Sin embargo en 3.9(b) la acción "comenzar a girar" (`S`) no finaliza por sí misma, sino que es seguida de una condición trivial, esto es, una condición que siempre se cumple, después de la

cual comienza la acción "levantar brazo izquierdo". Por tanto en este punto, el robot estaría girando mientras también levanta el brazo izquierdo. Esta secuencia simple, finaliza cuando, después de haber terminado de levantar el brazo izquierdo se ejecuta la acción "parar de girar" que mandaría finalizar la acción anterior de "comenzar a girar".

Acciones basadas en objetos

En todo robot, no solo en Maggie, la facultad de realizar movimientos o actos, ya sean actos motores, actos verbales, audiovisuales, o de cualquier tipo, pasa por la presencia, en el más bajo nivel, de un conjunto discreto de *actuadores*: motores, sistema de audio, pantallas, etc. Sea

$$\Omega_\alpha = \{\alpha_0(p_0, v_0), \alpha_1(p_1, v_1), ..., \alpha_{n-1}(p_{n-1}, v_{n-1})\} \tag{3.1}$$

el conjunto de todos los comandos de movimiento elementales, en sentido general, de los n actuadores α_i de un robot. En el modelo más sencillo, el acceso a la movilidad de cada uno de estos actuadores, puede ser realizado mediante una función que recibe dos parámetros: uno de aspecto *espacial* y otro de aspecto *temporal*. Así por ejemplo, en un motor el primer parámetro podría ser la *posición* a la que se quiere que llegue y el otro la *velocidad* con la que se quiere que se mueva. En un "actuador" de audio, el primer parámetro podría representar los valores de la amplitud de la onda de audio que se quiere sintetizar y el segundo, el tiempo de duración de dicha emisión, etc.

Una representación funcional tan sencilla nos permite realizar una equivalencia *gramatical* o sintáctica directa, en lo que es el acceso mediante lenguaje verbal a los actuadores del robot. Podemos asociar el comando de movimiento del actuador con la partícula lingüística *verbo*, y asociar los dos parámetros incluidos en el comando, con *complementos* de este verbo: complemento circunstancial de lugar (dónde mover), complemento directo (qué onda sintetizar), complemento circunstancial de modo (cómo moverlo), etc. El actuador en sí, en este sentido, es también un complemento del verbo: su objeto directo o indirecto.

De este modo, el enunciado

"Mueve el actuador tres, cuarenta y cinco grados a veinte grados por segundo"

puede fácilmente hacerse corresponder con la función:

$$\alpha_3(45, 20)$$

Esta equivalencia no es sino la utilizada de modo "pseudonatural" a la hora de escribir un programa para un actuador y escoger el nombre de las funciones que lo mueven y de sus parámetros. Así utilizando programación orientada a objetos el actuador se representa de modo abstracto como una clase y sus métodos representan las acciones que se pueden realizar en el actuador.

Ahora bien, naturalmente desde una perspectiva externa el usuario no percibe por separado cada uno de los actuadores del robot, sino que observa en el robot *articulaciones* aisladas, en sentido general: cabeza, brazos, piernas, capacidad de habla del robot, etc. Cada articulación internamente es un conjunto de actuadores coordinados, pero al usuario tal fisiología le resulta transparente. Sea

$$\Omega_\theta = \{\theta_0, \theta_1, ..., \theta_{m-1}\} \tag{3.2}$$

el conjunto de todas las funciones de movimiento de las m articulaciones (en sentido general, es decir, incluyendo articulaciones no necesariamente mecánicas) del robot. De este modo se cumple que $m \leq n$, dado que cada articulación estará manejada por al menos un actuador. El objetivo ahora persigue cómo acceder desde el lenguaje verbal, de modo general, a dichas funciones de movimiento.

Al aumentar la complejidad de movimiento del robot, el acceso lingüístico a sus capacidades de movimiento ahora no es tan directo como acceder a su posición y velocidad. Esto es debido a que, precisamente, el número posible de movimientos ha aumentado. Así por ejemplo, si pensamos en un manipulador de varios grados de libertad, la cantidad de movimientos y variaciones en cuanto a velocidad, trayectorias, etc es prácticamente ilimitada; lo mismo si pensamos en la cantidad de frases que puede sintetizar un actuador que convierta el texto en habla, y así con cualquier "articulación" del robot.

Sin embargo, nos encontramos con que, en la práctica, los movimientos que realiza un robot no son tan amplios, y que el usuario no necesita acceder directamente a todas las posibilidades que ofrece cada articulación, sino que mediante lenguaje natural el usuario suele hacer mención a un dominio reducido de funcionalidades del robot.

Nuestro objetivo es el de diseñar e implementar un método que realice la unión entre un conjunto de enunciados del usuario con el de las acciones y condiciones posibles en el robot, es decir, *hacer accesible mediante habla las funcionalidades del robot.*

Así por ejemplo, el enunciado:

"Levanta el brazo izquierdo"

correspondería con el siguiente método:

$$Arm.move(p_0, v_0)$$

donde p_0 corresponde con una posición de "brazo levantado" y v_0 correspondería con una velocidad por defecto.

Para realizar el presente trabajo se parte de un dominio inicial de acciones en el robot que sea fácilmente escalable *ad líbitum*. El conjunto de acciones de partida que se ha escogido, en Maggie, incluye movimientos de ambos brazos, de la cabeza, de los párpados y de traslación y rotación:

- Levantar o bajar el párpado derecho o izquierdo.
- Levantar o bajar la cabeza.
- Girar la cabeza a la izquierda, a la derecha o al centro.
- Levantar o bajar el brazo derecho o izquierdo.
- Girar todo el cuerpo a la izquierda o a la derecha.
- Avanzar o retroceder.

Cada sensor y actuador del robot se representa mediante un objeto, y sus funcionalidades, mediante métodos de dicho objeto. Así, en este dominio inicial de acciones en el robot, tendríamos los siguientes métodos:

- `leftEye.move(position)`, para movimientos del párpado izquierdo.
- `rightEye.move(position)`, para movimientos del párpado derecho.
- `Neck.move(position)`, para los movimientos verticales de la cabeza.
- `Neck.spin(position)`, para los movimientos horizontales de la cabeza.
- `leftArm.move(position)`, para los movimientos del brazo izquierdo.
- `rightArm.move(position)`, para los movimientos del brazo derecho.
- `Base.move(position)`, para los movimientos de traslación del robot.
- `Base.spin(position)`, para los movimientos de rotación del robot.

En esta primera versión de este dominio inicial de acciones, los valores de la posición de cada función están prefijados. Así por ejemplo, para los movimientos del brazo, puesto que se consideran dos acciones posibles: subir o bajar el brazo, los valores de posición prefijados corresponden con cada una de estas acciones. Para subir, por ejemplo, se hace a una posición fija que coloca el brazo objetivamente arriba. Naturalmente el valor de este parámetro también podría tomarse de la interacción conversacional con el usuario.

La elección de esta nomenclatura se ha realizado de modo que facilite la construcción de cada uno de estos métodos a partir de los valores semánticos de la gramática del reconocedor automático del habla. Esta construcción se ha detallado en el apartado 6.2, donde se habla del paso que hay entre las gramáticas semánticas y estos métodos asociados a las acciones.

3.5.2. Dominio de condiciones del robot

Las condiciones también se han dividido en dos tipos: explícitas e implícitas. Las primeras se refieren a condiciones asociadas a algún evento de la realidad interna o externa al robot. Por ejemplo, "tocado en el hombro derecho" sería una condición de este tipo. La segunda se refiere a condiciones que indican la finalización o no de alguna acción con final. Por ejemplo en la figura 3.9(a) "fin de subir brazo izquierdo".

Cada condición se representa mediante una función primitiva `Python` con sus parámetros, que está predefinida. Esta función accede a la información del sensor/actuador o lee de la memoria a corto plazo y del sistema de eventos la información necesaria para evaluar en cierto instante si la condición es verdadera o falsa. Así por ejemplo, la condición asociada a los eventos de tacto en la carcasa del robot viene denotada por `touch( body, side )`. Esta función comprueba que el último evento enviado es de tacto, lee el estado de los sensores y evalúa, si se ha tocado la parte del cuerpo correspondiente a sus parámetros `body` y `side`. Otro ejemplo, la función de evaluación que comprueba si se ha subido una parte del cuerpo del robot tiene el prototipo `rise( body, side )`, que es el mismo de la acción "subir una parte del cuerpo". Por dentro, esta función realiza una lectura de los datos de la odometría para comprobar que se ha llegado a una posición considerada como de subida.

3.5.3. Implementación de una secuencia. Secuencia alterable

Una vez expuestos tanto el sistema de representación de las secuencias que se ha utilizado en el presente trabajo (ver B.3), como algunos de los detalles del robot utilizado, Maggie, se va a explicar cómo el robot ejecuta las secuencias.

La secuencia representada mediante un SFC se implementa en un fichero XML y es interpretada mediante un objeto que denominamos *secuenciador*. La ventaja de representar la secuencia en un fichero interpretable es que dicha secuencia puede ser modificada en tiempo de ejecución, esto es, sin tener que modificar ninguna parte del software ya implementado. Además, la secuencia así construida puede ser modificada mientras la propia secuencia es ejecutada. Hemos venido a denominar *alterabilidad* de la secuencia a esta característica.

En las secuencias utilizadas, cada etapa está asociada a una *función de activación* y a una *función de desactivación*. Asimismo, cada transición está asociada a una *función de evaluación* de disparo.

De este modo, en el fichero que representa la secuencia se implementan dos cosas: la estructura de la secuencia, y las funciones de cada nodo. La estructura consta de las distintas relaciones y uniones entre las etapas y transiciones que la componen, y la situación inicial de las marcas. Las funciones de cada nodo son las funciones de activación y desactivación asociadas a cada etapa, así como las funciones de evaluación asociadas a cada transición.

Naturalmente, para permitir que la secuencia sea alterable, dichas funciones también tienen que estar representadas o implementadas en un lenguaje interpretable. Por su versatilidad, facilidad de uso y compatibilidad, el lenguaje escogido es `Python` [Martelli, 2006].

De esta manera, la funcionalidad del robot, las acciones y condiciones que pueden ser incorporadas en una secuencia quedan descritas mediante código en `Python` que, como se explicará más adelante, será incluido en la secuencia alterable mediante interacción por voz.

Cabe destacar que así como la secuencia alterable en sí, esto es, su estructura puede ser modificada en tiempo de ejecución, también el dominio de acciones y condiciones del robot pueden ser modificados sin necesidad de realizar ningún cambio en el software que implementa la arquitectura de control del robot. De este modo nuevas acciones y nuevas condiciones pueden añadirse al dominio del robot dinámicamente. Baste para ello un constructor dinámico adecuado de funciones `Python`, lo cual se propone como trabajo

futuro en el capítulo 10.

La secuencia se representa siguiendo un esquema estructurado en formato `XML`, tal y como se muestra en la figura 3.10. Las etiquetas que definen la secuencia son:

- `<Descripcion>`, donde se añade un texto que sirva para describir qué hace la secuencia.
- `<nEtapas>`, donde se especifica el número de etapas de la secuencia.
- `<nTransiciones>`, donde se especifica el número de transiciones de la secuencia.
- `<mPreCondiciones>`. Aquí se define la matriz de pre-condiciones. Esta matriz indica mediante valores 0 o 1 si existe o no una conexión entre la salida de una etapa y la entrada a una transición. La etapa se identifica mediante el índice de las filas, y la transición mediante el de las columnas. Cada valor de la matriz se introduce mediante etiquetas `<valor>`. La definición de la matriz se realiza por filas mediante etiquetas `<fila>`.
- `<mPostCondiciones>`. Aquí se define la matriz de post-condiciones. Esta matriz indica mediante valores 0 o 1 si existe o no una conexión entre la salida de una transición y la entrada a una etapa. La etapa se identifica mediante el índice de las filas, y la transición mediante el de las columnas. Cada valor de la matriz se introduce mediante etiquetas `<valor>`. La definición de la matriz se realiza por filas mediante etiquetas `<fila>`.
- `<vMarcas>`, es el vector de marcas inicial. Indica qué etapas están activas al comienzo de la ejecución de una secuencia.
- `<FuncionEvaluacion>`, donde se introduce empaquetado el código en `Python` asociado a la función de evaluación de una transición.
- `<id>`. Esta etiqueta identifica con qué transición corresponde una función de evaluación y con qué etapa corresponde una función de activación o de desactivación.
- `<FuncionActivacion>`, donde se introduce empaquetado el código en `Python` asociado a la función de activación de una etapa identificada por el valor en la etiqueta `<id>`.

- `<FuncionDesActivacion>`, donde se introduce empaquetado el código en `Python` asociado a la función de desactivación de una etapa identificada por el valor en la etiqueta `<id>`.

En este formato, por tanto, quedan representados todas las partes de la secuencia: número de etapas, número de transiciones, matrices de pre y post-condiciones[4], vector de las marcas iniciales (etapas activas en el comienzo), funciones de activación y desactivación de cada etapa y función de evaluación asociada a cada transición.

[4] véase B.2.1, donde se define formalmente una red de Petri mediante estas matrices

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Secuencia xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    " xsi:noNamespaceSchemaLocation="">
  <Descripcion>
    Aqui se coloca un texto opcional que sirva para describir
       en que consiste la secuencia.
  </Descripcion>

  <nEtapas>nS</nEtapas>
  <nTransiciones>nT</nTransiciones>

  <mPreCondiciones>
    <fila><valor>1 o 0</valor>...</fila>
    ...
  </mPreCondiciones>

  <mPostCondiciones>
    <fila><valor>1 o 0</valor>...</fila>
      ...
  </mPostCondiciones>

  <vMarcas><fila><valor>1 o 0</valor>...</fila></vMarcas>

  <FuncionEvaluacion>
    <id>0</id>
    <linea>funcion Python que se evalua en la transicion n = 0<
       /linea>
    ...
  </FuncionEvaluacion>
  ...
  <FuncionActivacion>
    <id>0</id>
    <linea>funcion Python que se ejecuta al activarse la etapa
       n = 0</linea>
    ...
  </FuncionActivacion>
  ...
  <FuncionDesActivacion>
    <id>0</id>
    <linea>funcion Python que se ejecuta al desactivarse la
       etapa n = 0</linea>
  </FuncionDesActivacion>
  ...
</Secuencia>
```

Fig. 3.10: *Esquema general de la representación de una secuencia*

4. DESARROLLO DEL RECONOCEDOR AUTOMÁTICO DEL HABLA

El principal objetivo del componente de reconocimiento automático de habla es el de convertir la señal acústica del habla del usuario en una secuencia de palabras. Este proceso suele realizarse en distintas fases que van del análisis de la señal acústica a un análisis lingüístico del habla.

Una completa explicación con detalles técnicos del reconocimiento de voz está fuera de los límites de este trabajo. No obstante una pequeña reseña sobre el tema sí que se realiza para justificar porqué se ha diseñado el resto del sistema del modo en que se ha hecho, esto es, intentando minimizar los errores del reconocimiento y tratándolos del modo más satisfactorio cuando estos errores se han dado. El reconocimiento automático del habla es todavía una tecnología en desarrollo. El habla espontánea propia del lenguaje natural cuenta, además, con elementos que hacen que el reconocimiento de voz sea todavía más tedioso. Por ejemplo, el tratamiento de sonidos que no corresponden con palabras, enunciados con dudas, repeticiones y "autocorrecciones", así como el uso de palabras fuera del diccionario utilizado por el reconocedor, etc.

Siguiendo la literatura más reciente que trata el reconocimiento automático del habla en plataformas de interacción humano máquina y humano robot, como en [Jurafsky and Martin, 2000], [Cole et al., 1997] y [Furui, 2005], identificamos unas fases estándar en el procesamiento de la información contenida en la voz del usuario, si bien, no todas las herramientas de reconocimiento automático del habla funcionan así. Estas fases pueden enumerarse del siguiente modo:

1. Procesamiento de la señal acústica. Donde se crean o "extraen" características espectrales de dicha señal, que llamaremos *observables*.

2. Identificación de fonemas, a partir del conjunto de parámetros obteni-

dos en el paso anterior.

3. Identificación de palabras, a partir del conjunto de fonemas obtenidos en el paso anterior.

4. Identificación de las frases más acertadas, a partir del conjunto de palabras obtenidas en el paso anterior y mediante el uso de gramáticas literales.

5. Extracción de la información relevante, a partir de las frases obtenidas en el paso anterior, y mediante el uso de gramáticas semánticas.

Las *gramáticas literales* se han implementado como gramáticas formales de contexto libre que siguen la formulación propuesta por el estándar `ABNF` (Augmented Backus-Naur Form). Este tipo de gramáticas es explicado en el apartado C.3.2, y en esta sección se da una especificación concreta de cómo se han implementado para el presente trabajo.

Las *gramáticas semánticas* no pertenecen al conjunto de gramáticas tal y como define la teoría de lenguajes formales, o como se representa en la jerarquía de Chomsky (ver figura C.1), sino que son pequeños *scripts* que se encapsulan dentro de las gramáticas literales y que sirven para obtener la información relevante para la aplicación que está utilizando el reconocimiento de habla. También se especificarán con mayor detalle en esta sección.

4.1. Técnicas en el reconocimiento automático del habla

4.1.1. Procesamiento de la señal: de la señal acústica al fonema

Muestreo

La señal acústica obtenida mediante un micrófono se convierte en información digital. Para ello se somete a un proceso de muestreo a una frecuencia que coincide con el doble de la frecuencia crítica o *frecuencia de Nyquist*. El muestreo supone una pérdida del ancho de banda de la señal y, por tanto, de su calidad, de modo que cuanto más pequeña es la frecuencia de muestreo, mayor es dicha pérdida. Por otro lado, cuanto mayor es la frecuencia de muestreo, mayor número de datos se obtienen y, por tanto, más costo va a resultar cualquier cómputo que quiera realizarse posteriormente. Las

frecuencias de muestreo más utilizadas tienen en cuenta el ancho de banda del canal de comunicación y la velocidad de cómputo de la aplicación que va a utilizar dicha señal. Así tenemos 8kHz para señales que viajen por vía telefónica; para señales de voz para reconocimiento del habla 11kHz - 22kHz suele ser más que suficiente; y 44,1Hz o frecuencias de muestreo superiores se utilizan para señales de calidad óptima.

Cuantización

Además de someterse a un muestreo en el tiempo, cada valor obtenido de cada muestra de la señal acústica se cuantiza digitalmente, esto es, se codifica mediante una cadena de bits. La pérdida de información, en este sentido, resulta del redondeo digital que se produce al tener un límite en el número de cifras significativas que la codificación digital permite representar (ruido de cuantización). Por tanto, el número de bits utilizados en la cuantización limita el denominado *margen dinámico* de la señal, así como la precisión en su representación. Para una cuantización de `16 bits`, tendremos la posibilidad de representar 65536 (2^{16}) valores distintos, lo cual nos da unos `96dB` de margen dinámico ($20log_{10}(2^{16})$. Afortunadamente, la voz hablada se mueve en un margen dinámico entorno a los `50dB`, muy inferior al margen de, por ejemplo, una ópera de Wagner (unos `110dB`) cargada de sutilezas dinámicas.

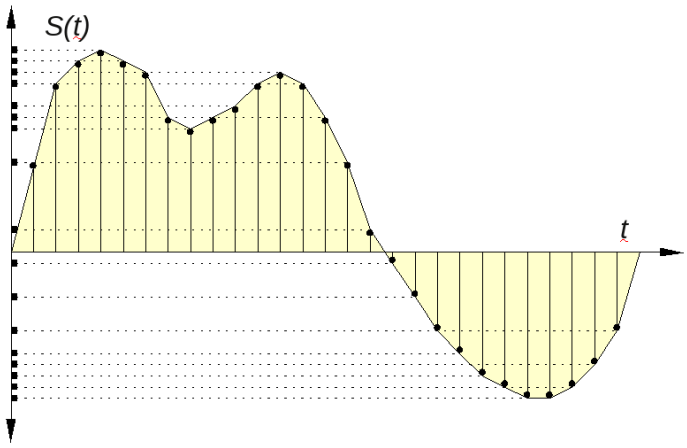


Fig. 4.1: *Ejemplo de muestreo y cuantización de una señal analógica*

Extracción de componentes espectrales

Una vez muestreada y cuantizada, la señal acústica de voz se analiza en "rodajas" de entre 10 y 20 ms. De cada una de estas partes se extraen un conjunto de parámetros espectrales LPC (*Linear Predictive Coding*) como son las *componentes cepstrales de frecuencia de mel* (MFCC) o coeficientes de Fourier del espectro de la señal, los *formantes* o picos espectrales asociados a un fonema; y parámetros estadísticos como coeficientes de regresión entre los parámetros de las distintas rodajas, valores medios, máximos y mínimos, mediana, varianza, primeros cuartiles, etc. Además, algunas herramientas también consideran parámetros espectrales PLP (Perceptual Linear Predictive) que, básicamente, son los mismos parámetros LPC, pero donde previamente se hace pasar la señal acústica por ciertos filtros que simulan la audición humana. La extracción de todos estas características acústicas dejan la señal representada como un conjunto de parámetros que denominamos observables.

Las componentes espectrales no solo se utilizan en el reconocimiento del habla tal y como se va a explicar más adelante, sino que también se vienen usando para la detección de ciertas características emotivas en el habla natural. Por ejemplo, en [Oudeyer, 2003] se explica con detalle el uso de distintos algoritmos para la detección de el grado de presencia de cinco emociones básicas en el habla (calma, tristeza, enfado, alegría y miedo).

También estas componentes espectrales, junto con otras características acústicas del enunciado, se han venido utilizando para la detección de intenciones comunicativas en el habla: si se está realizando una realimentación positiva (alabanza, elogio, ensalzamiento, etc) frente a una realimentación negativa (crítica, censura, vituperio, etc) Así, en [Breazeal and Scasselatti, 2000] se muestra al torso robótico `Kismet` que expresando emociones faciales reacciona ante las distintas características prosódicas de los enunciados del usuario.

Decodificación en fonemas

Los vectores de parámetros observables obtenidos de las características acústicas de la señal de voz se han de computar hacia la estimación de un vector de fonemas probables, que correspondan con dichos observables. Existen dos procesos fundamentales para realizar este cómputo: redes neuronales y funciones gausianas. Ambos establecen una función de densidad de probabilidad (PDF) en un espacio continuo. Las funciones gausianas relacionan un vector de observables con una probabilidad de fonema. Las redes neuronales

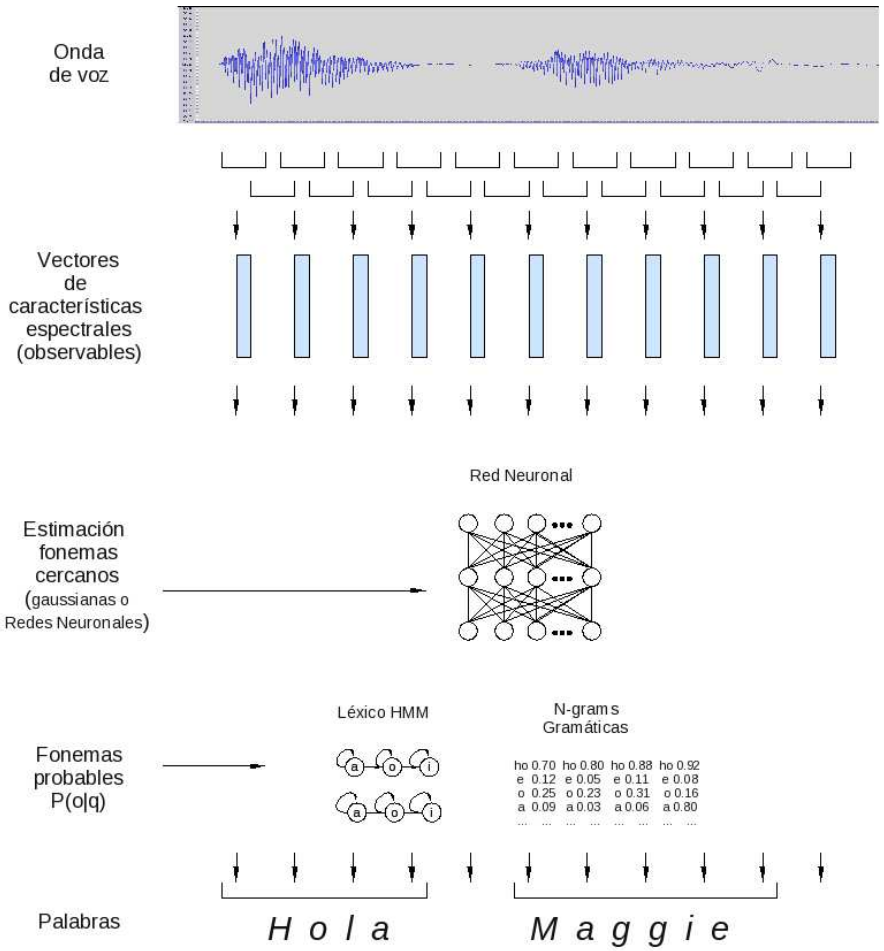


Fig. 4.2: *Arquitectura esquemática simplificada de un reconocedor automático de habla*

reciben como entrada un conjunto de vectores observables y devuelven una probabilidad para cada fonema del lenguaje.

En la figura 4.2 se ha representado de modo muy simplificado todo el proceso que sufre la señal acústica hasta ser interpretada como una cadena de palabras, tal y como se ha comentado.

4.1.2. Modelo acústico: de los fonemas a la palabra

Podría definirse el modelo acústico de un lenguaje hablado como el conjunto de estructuras que representan el inventario de fonemas del lenguaje y su pronunciación. La representación más utilizada en este sentido es mediante cadenas ocultas de Markov (HMM).

Una cadena de Markov es un caso especial de un autómata de estados finitos probabilístico (PFSM). Por tanto, intrínsecamente representa una secuencia no ambigua dado que cada estado está bien determinado. En resumen, los parámetros que definen una HMM son:

- **estados:** un conjunto de estados que se relacionan entre sí mediante probabilidades de transición y que se relacionan con un conjunto de observables mediante funciones probabilísticas.

- **probabilidades de transición:** un conjunto de probabilidades. Cada una representa la probabilidad de transición de un estado a otro.

- **probabilidades de observables:** un conjunto de funciones probabilísticas que relacionan cada estado con un vector de parámetros observables.

- **distribución inicial:** distribución probabilística inicial que establece la probabilidad de cada estado de la cadena en ser un estado inicial.

Como en el reconocimiento automático del habla el alfabeto de salida es bien conocido (el inventario de fonemas del lenguaje) una cadena de Markov es una buena representación de las relaciones entre dichos fonemas: cada fonema queda representado por un estado de la cadena. Por otro lado, el problema que consigue solucionar las cadenas ocultas de Markov es el de relacionar unas entradas que no forman un alfabeto no ambiguo, como son los parámetros observables que representan la señal de voz, con unas salidas que son dicho inventario de fonemas. Esta relaciones son las funciones probabilísticas gaussianas o redes neuronales descritas en el apartado anterior.

Así, dados unos parámetros observables representando una señal de voz, las cadenas ocultas de Markov permiten determinar cuáles son las palabras más probables que corresponden a dichos observables dentro del modelo acústico dado (véase figura 4.2).

4.1.3. Modelo del lenguaje: de las palabras a la frase

El así denominado *modelo del lenguaje* es un modelo estadístico de secuencias de palabras. Puede obtenerse de diversos modos: a partir de un *corpus*, es decir, de una serie de enunciados completos típicos del lenguaje; o a partir de una gramática que incluya expresiones regulares. El modelo de lenguaje también se puede representar con diversas estructuras, aunque la más típica suele ser un árbol de probabilidades, donde en cada nodo se sitúa, o bien, una palabra probable o un identificador de un conjunto de palabras probables, como sería un símbolo no terminal.

Uno de los problemas más frecuentes que encontramos en un reconocedor automático del habla natural reside en el desfase temporal existente entre el procesamiento de un enunciado del usuario y la articulación, por parte del usuario, de dicho enunciado. Es decir, en este punto de análisis acústico del habla, para la herramienta de reconocimiento resulta relevante saber si el enunciado del usuario ha terminado o no y saber si debe dar ya un resultado de reconocimiento o esperar más información acústica.

El modelo del lenguaje suele expresarse mediante alguna de estas dos entidades: `N-grams` o *gramática literal*. Las `N-grams` sirven para predecir una palabra dentro de una frase a partir de sus `N-1` predecesoras. Suelen construirse a partir de un *corpus* de modo que cuanto mayor sea dicho *corpus* mejor será el modelo construido. Estas estructuras suelen ser utilizadas por las herramientas de dictado, que son distintas a las de reconocimiento de voz.

Pero la estructura más utilizada para el reconocimiento del habla y que resulta fundamental en el desarrollo de este trabajo, es la gramática, que aquí denominamos *gramática literal* para distinguirla de la *gramática semántica*. La primera sirve para construir un modelo formal del lenguaje, mientras que la segunda sirve para extraer del resultado del reconocimiento la información relevante para la aplicación que está utilizando dicho reconocimiento.

En esta sección se termina de desarrollar el concepto de gramática introducido en el apéndice C.3.2 y su uso para con el reconocedor automático del habla. Se dan algunos ejemplos de gramáticas utilizadas siguiendo el estándar, ya introducido anteriormente, `ABNF` (Augmented Backus-Naur Form), basado en las gramáticas de contexto libre.

gramática literal en el estándar `ABNF`

El término gramática viene del latín: *grammatîca* y éste del griego *grammatika*. Una de las definiciones que del término ofrece la RAE es *"Ciencia que estudia los elementos de una lengua y sus combinaciones"*. El término

también se viene utilizando para describir las estructuras de cierta área de conocimiento. Existen así trabajos como "Una gramática de la música" (Thomas Busby, 1818) o "Una gramática del color" (George Field, 1888) también se habla de una gramática de los gráficos, de la conducta o, cómo no, una gramática de la interacción [Ochs et al., 1996]. Esto se debe a que la gramática establece a la vez un vocabulario específico y finito de elementos, que pueden ser conceptuales, y un conjunto de relaciones de estos elementos.

En el reconocimiento automático del habla, se entiende por "gramática" como una gramática formal libre de contexto que incluye una expresión regular de símbolos terminales (palabras de un vocabulario) y no terminales.

Al contrario que las herramientas de dictado, en un reconocimiento del habla interesa que la herramienta devuelva un resultado que se pragmático, es decir, que tenga una utilidad práctica en la aplicación que utiliza el reconocedor. Por tanto, de alguna manera, el reconocedor tiene que contar con alguna estructura que le informe de cómo la aplicación quiere el resultado del reconocimiento. Esto se realiza mediante gramáticas.

Por otro lado, cada gramática establece un vocabulario o léxico, y unas reglas de unión de las palabras de ese vocabulario que ayudan al reconocedor a reducir el espacio de búsqueda dentro de las posibles palabras que se obtienen en el proceso de paso de los fonemas a las palabras explicado anteriormente.

Actualmente, existen dos tipos principales de representación de las gramáticas literales para los reconocedores de voz, tal y como distingue la definición del estándar `SRGS` (Speech Recognition Grammar Specification) propuesto por la W3C[1]: una representación basada en un lenguaje de etiquetas tipo `XML`, y otra representación basada en el formato `ABNF` (Augmented Backus-Naur Form) cuya sintaxis se basa en la gramática formal de contexto libre (véase apéndice C). Ambos representan la misma y solamente la misma información, por lo que es fácil pasar de una representación a la otra.

La gramática literal consta de una serie de *reglas* que mantienen la forma de las reglas de una gramática de contexto libre. En casi todas las gramáticas existe una *regla raíz* que engloba al resto de reglas. El término "raíz" se debe a que la regla raíz está en la parte inferior, final o raíz del árbol que el reconocedor de habla genera a partir de la gramática.

Cada regla consta por tanto de una expresión formal en la que se relacionan símbolos terminales y no terminales. Los símbolos no terminales van precedidos del símbolo "`$`". El final de cada regla viene marcado por el símbolo "`;`". De todos los elementos de una relación formal dentro de una gramática

[1] http://www.w3.org/TR/speech-grammar/

ABNF, los más utilizados son los siguientes:

- [] : que encierran símbolos opcionales.

  Por ejemplo: `$grammar=[magui]saluda;` correspondería tanto con el enunciado "magui saluda" como con el enunciado "saluda".

- | : marca una alternativa.

  Por ejemplo: `$grammar=izquierda|derecha;` correspondería con uno de los dos enunciados "izquierda" o "derecha".

- * : permite la repetición de un símbolo varias veces o ninguna.

  Por ejemplo: `$grammar=escucha*magui` correspondería con enunciados como "magui" o "escucha escucha escucha magui".

- + : permite la repetición de un símbolo varias veces y como mínimo una.

  Por ejemplo: `$grammar=escucha+magui` correspondería con enunciados como "escucha magui" o "escucha escucha escucha magui".

- (): que agrupan operaciones con preferencia dentro de la gramática.

  Así por ejemplo mientras que la gramática `$grammar=amor|es;` correspondería con uno de los dos enunciados "amor" o "es", la gramática `$grammar=amo(r|res);` lo hace con "amor" o "amores".

La operación AND que marca una secuencia obligatoria de símbolos viene dada por el carácter "espacio en blanco".

La implementación de una gramática generalmente se realiza mediante uno o varios ficheros de texto. Se suele utilizar la extensión .grxml para gramáticas implementadas siguiendo el estándar XML definido en el estándar SRGS, y la extensión .gram para las que siguen el formato ABNF. Desde un fichero puede apelarse a una regla de otro fichero, lo cual permite estructurar ordenadamente la implementación de todo el sistema de gramáticas.

Ejemplo de gramática literal En el siguiente ejemplo se presentan varias reglas gramaticales en forma de no terminales que se combinan en una regla principal que define la gramática.

```
#ABNF 1.0 ISO-8859-1;
language es-ES;
```

```
tag-format <loq-semantics/1.0>;
public $root = $dialog;

$dialog = ($fp | $fc | $ky | $nn | $ad | $e);

/* conventional-opening */
$fp = hola [magui];

/* conventional-closing */
$fc = adios [magui];

/* response acknowledgement or yes answer */
$ky = okei|vale|de_acuerdo|si|afirmativo;

/* no answer */
$nn = no;

/* action-directive */
$ad = avanza | retrocede | gira_a_la $side |
      (levanta | baja) el brazo $side;

/* side */
$side = izquierd(a|o) | derech(a|o);

/* turn-exit */
$e = sa(l|lir)|termin(a|ar);
```

Así se muestran varias reglas que conformarían una gramática completa, que haría sensible al reconocedor de habla a locuciones del usuario que incluyan algunos de los siguientes actos: saludos, despedidas, negaciones, afirmaciones y algunas órdenes concretas.

El resultado que da el reconocedor ante una gramática literal, es una lista de los mejores resultados palabra por palabra. Estos resultados se denominan `N-best` y puede configurarse el número de resultados mejores obtenidos. Es decir que devuelve las mejores secuencias de palabras que se ajustan, por un lado a la gramática cargada en el reconocedor, por otro al análisis acústico de los enunciados del usuario.

A éste resultado lo acompañan dos tipos de parámetros que miden el grado de acierto o de confianza del resultado del reconocimiento. Por un

lado, se muestra el grado de confianza de la frase en su totalidad, por otro se muestra el grado de confianza de cada una de las palabras reconocidas.

Naturalmente, cuanto más compleja es una gramática, esto es, cuanto más vocablos incluye y/o cuanto más complejo es el árbol de posibilidades que genera, más fácil resulta que aparezcan errores en el reconocimiento. Esto se soluciona cargando y descargando gramáticas dinámicamente dependiendo del contexto en el que se encuentre el proceso del diálogo. También es más fácil que aparezcan errores en reglas que incluyan vocablos cortos (como por ejemplo afirmación con el vocablo "si" o negación con "no"), o vocablos que tengan cierta afinidad fonética (por ejemplo con vocablos como "Javi", "Magui", "fácil", etc). No hay un método sistemático para solucionar estas dificultades que dependen de la construcción de la gramática, aunque sí se han desarrollado herramientas que realizan una estimación previa de éstos problemas y ayudan a hacer un diseño lo más óptimo posible.

4.1.4. Interpretación semántica

Con la gramática literal conseguimos que el reconocedor automático del habla obtenga, en una frase, una secuencia "literal" de las palabras que más aproximan lo que el usuario ha enunciado con el lenguaje formal descrito por la gramática. Pero este tipo de información no resulta relevante *per se* para la aplicación que utiliza el reconocedor. De alguna manera, dicha aplicación requiere algún tipo de interpretación o "traducción" del resultado literal *externo* a algún tipo de estructura *interna* a la aplicación. Esto se realiza mediante la gramática semántica.

Existe la descripción de un estándar para la interpretación semántica en reconocimiento automático del habla desarrollado por el W3C. Se denomina `SISR` (Semantic Interpretation for Speech Recognition) [2] y establece la posibilidad de encapsular en etiquetas `<tag>` código en lenguaje `ECMAScript` dentro de una gramática en formato XML descrito en el estándar `SRGS` (Speech Recognition Grammar Specification)[3]. Los conceptos que introduce son también utilizados por otros lenguajes de interpretación semántica dependientes del fabricante del reconocedor. En este trabajo se van a explicar estos conceptos según la implementación del reconocedor de habla comercial utilizado: `Loquendo-7.1` [Technology and Services., 2010], cuyas gramáticas "literal-semánticas" están basadas en el estándar `ABNF`, comentado en la sección 4.1.3.

[2] http://www.w3.org/TR/semantic-interpretation/
[3] http://www.w3.org/TR/speech-grammar/

La gramática semántica permite incorporar dos tipos de interpretaciones semánticas dentro de cada regla de la gramática literal:

- `<@atributo=valor>`, donde `@atributo` es una *variable de regla* y `valor` es un dato de tipo cadena de caracteres, numérico o booleano, que representa un *valor semántico*.
- `alpha:a`,donde `alpha` es una cadena de símbolos terminales y/o no terminales, y `a` es un no terminal. Esta asignación establece una especie de interpretación del texto reconocido.

Así por ejemplo, en la siguiente regla de una gramática semántica,

```
$ky = (okei|vale|de_acuerdo|si|afirmativo){<@da "ky">};
```

cuando el usuario enuncia alguna de las frases "okei", "vale", "si", etc, el reconocedor no solo devuelve como resultado del reconocimiento el símbolo terminal correspondiente a la frase enunciada (`okei`, `vale`, `si`, etc) sino que además, asigna a la variable `da` el valor fijo, del tipo "cadena de caracteres", `ky`. De este modo, la aplicación que esté utilizando el reconocimiento automático del habla, analizando el valor de dicha variable, puede utilizarlo para saber que, simplemente, el usuario ha realizado una afirmación y así, actuar en consecuencia.

En la siguiente regla de una gramática semántica,

```
$nd = ("([baile | secuencia] simple)":simpleseq |
       "([varios] [bailes | secuencias] en paralelo)":atonce |
       "(seleccion [de (bailes | secuencias) )":selseq)
       {<@da "nd"><@type $value>};
```

se realizan los dos tipos de asignaciones comentados. Ante un enunciado del usuario como "baile simple" o "secuencia simple" la regla convierte la cadena de no terminales reconocidos a un solo valor, en este caso, `simpleseq` que es el que devuelve el reconocedor. Lo mismo ocurre ante un enunciado como "selección de bailes" o "selección de secuencias" en los que la regla convierte la cadena de no terminales reconocidos al valor `selseq` . Estos valores de conversión son accesibles en la propia regla mediante una variable especial de las gramáticas semánticas que es `$value` y que puede ser utilizada más adelante dentro de la gramática.

Además se producen dos asignaciones más, por un lado se asigna a la variable `da` el valor fijo del tipo cadena de caracteres `nd`, y a la variable `type` el valor de la variable especial `$value`. Como ya se ha comentado, en

el caso de que el usuario haya enunciado la frase "baile simple", este valor es `simpleseq` y en el caso en que haya enunciado la frase "selección de bailes", este valor es `selseq`. Por tanto, la aplicación que utiliza reconocimiento de voz puede evaluar cualquiera de las variables `type` o `da` para conocer el tipo de enunciado que el usuario ha realizado.

Por último, cabe destacar la existencia, por parte de la mayoría de los reconocedores automáticos del habla, de ciertas variables especiales, como es `$GARBAGE`. Esta variable permite la inclusión de sonidos fonéticos que no correspondan con ninguno de los vocablos literales incluidos en la gramática. Así por ejemplo la siguiente gramática:

```
$body = $GARBAGE ( brazo | hombro |cabeza | ojo ) $GARBAGE;
```

puede reaccionar ante enunciados como "*en el* brazo", "brazo *izquierdo*", "ojo *mbfhff*".

Esta variable sirve así como una especie de comodín ante sonidos no contemplados explícitamente en la definición de la gramática. Naturalmente el resultado en el no terminal `$body` no incluirá ninguno de los vocablos que se han contenido en `$GARBAGE`. El uso de esta variable empeora de manera drástica el grado de confianza del reconocimiento, por lo que debe ser utilizada con atención.

Una desventaja que suele ir asociada al uso de este tipo de variable es que el nivel de confianza suele disminuir bastante cuando se incluye.

4.2. Habilidad de reconocimiento automático del habla: `asrSkill`

En esta sección se explican los detalles de una de las habilidades automáticas con las que cuenta el robot: `asrSkill` (Habilidad de Reconocimiento Automático del Habla). Esta habilidad se encarga de tomar los datos de audio del habla del usuario, a partir de los cuales construye un resultado literal y semántico, que puede ser utilizado por cualquier otra habilidad dentro de la arquitectura, como por ejemplo, por el sistema gestor del diálogo, que también se explica al final de este capítulo.

Esta habilidad cuenta con una parte interna encargada del reconocimiento en sí: el motor de reconocimiento, y con una parte externa que permite compartir los datos del reconocimiento con el sistema de diálogo.

4.2.1. Motor de reconocimiento automático del habla: `Loquendo ASR-7.7`

El motor de reconocimiento se implementa utilizando una herramienta de reconocimiento comercial de la empresa Loquendo, *Voice Technology and Service* [4]. Para realizar el reconocimiento, esta herramienta admite tanto la utilización de `N-Grams`, como de gramáticas literales y/o semánticas. Lo primero implementa un modelo del lenguaje, y sirve para un reconocimiento del habla continua con un modelo probabilístico de secuencias de palabras. Las gramáticas permitidas por la herramienta pueden estar implementadas tanto en formato `ABNF`, que ha sido introducido en la sección 4.1.3, como en los formatos `SRGS` y `SISR`, también explicados anteriormente. La habilidad `asrSkill` utiliza gramáticas semánticas en formato `ABNF`.

Las principales ventajas de `Loquendo-7.7` frente a otros reconocedores de habla pueden resumirse en las siguientes:

- **Independencia del usuario hablante.** Es decir, que no es necesario un entrenamiento previo y específico por parte de cada usuario que va a utilizar la herramienta. No obstante, la herramienta permite realizar tal entrenamiento y ganar así en grado de acierto.
- **Reconocimiento continuo del habla.** Es la propia herramienta quien se encarga a la vez de la toma de datos de audio y la obtención de resultados mediante cómputo contra la gramática cargada. Lo primero se

[4] Loquendo: Sociedad global - soluciones automáticas y tecnología vocal, http://www.loquendo.com/es/

realiza mediante una interfaz de funciones "callback" que, vienen implementadas en un paquete a parte, si bien pueden ser implementadas por el programador.

- **Alto grado de acierto en el reconocimiento.** Ante enunciados con vocablos sintácticamente estructurados conforme a las reglas de la gramática, se viene obteniendo un grado de acierto (tanto literal como semántico) de más de un 90 %. Naturalmente este acierto disminuye cuando el usuario no enuncia palabras dentro de la gramática o las enuncia en un orden no contemplado por la misma; o también, ante gramáticas más complejas. De ahí, la importancia en el diseño previo de las gramáticas.

- **Alto grado de confianza en el reconocimiento.** Cuando hay un acierto, la herramienta suele devolver un grado de confianza en torno al 80 %. Esto muestra que el número de posibles resultados alternativos es muy bajo, con lo que la herramienta muestra bastante univocidad en el resultado (no solo acierta, sino que lo hace con seguridad).

- **Rapidez de respuesta.** Para la complejidad de gramáticas que se vienen utilizando, los resultados suelen estar disponibles en un rango de unos $50 - 150\ ms$.

- **Identificación del usuario.** Esta capacidad se realiza mediante previo entrenamiento. Cada usuario graba unas "huellas" del habla que son utilizadas para identificarle acústicamente.

- **Gramáticas dinámicas.** La herramienta permite la carga y descarga de una gramática por otra en tiempo de ejecución.

Loquendo admite hasta 100,000 palabras en la construcción de las gramáticas utilizadas, y 50,000 palabras en los árboles probabilísticos de las `N-Grams`.

4.2.2. Funcionamiento interno de la habilidad

En la figura 4.3 se muestra una representación gráfica del esquema interno de la habilidad: su funcionamiento y su comunicación con el resto de la arquitectura: sistema de eventos y memoria a corto plazo.

El reconocimiento automático del habla se realiza mediante llamadas a funciones de la `API` (Application Programming Interface) del paquete comercial utilizado: `Loquendo-ASR-7.7`.

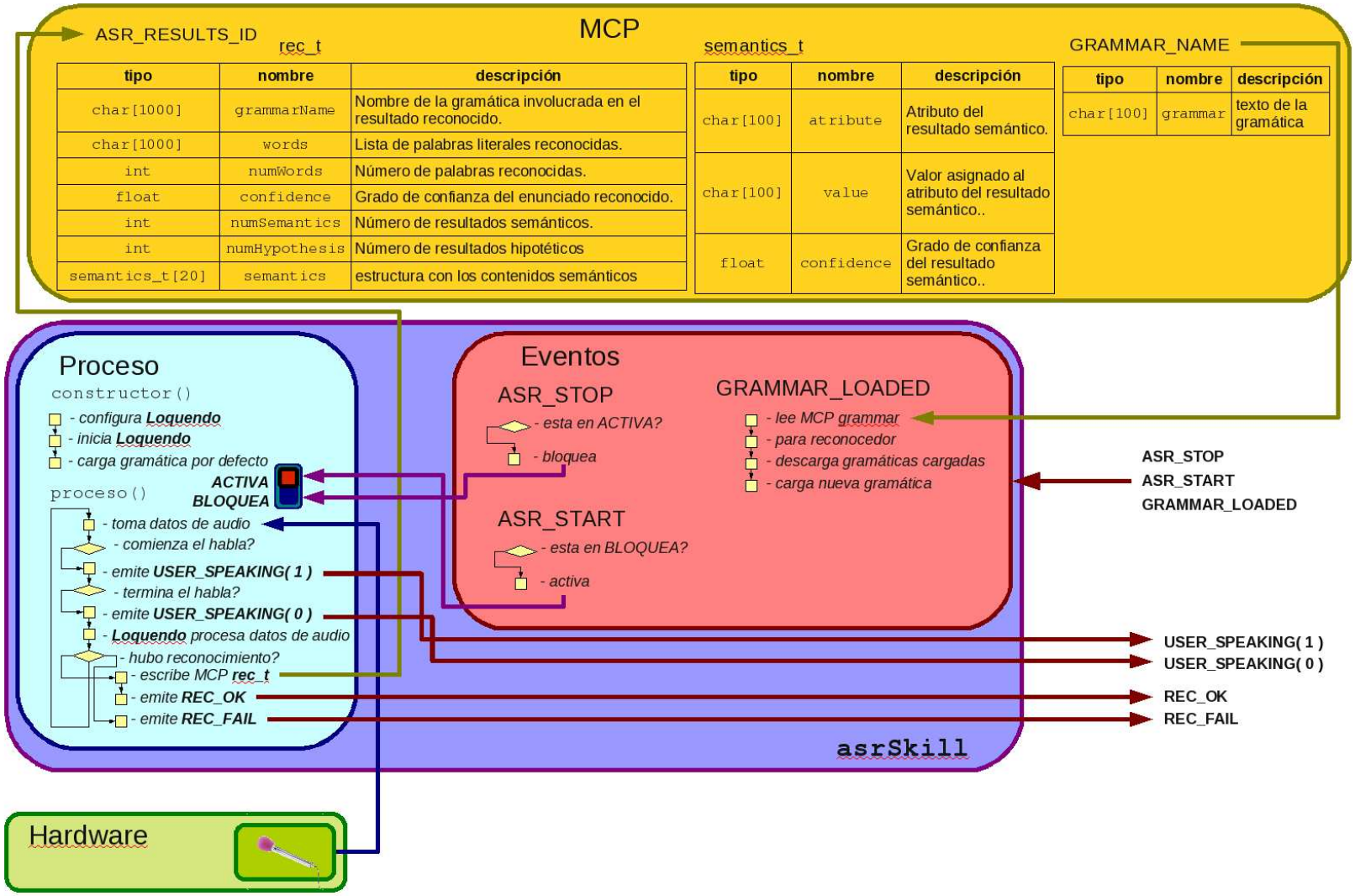


Fig. 4.3: *Esquema interno de la habilidad* `asrSkill`

La habilidad comienza configurando ciertos parámetros del motor de reconocimiento como son el idioma utilizado, el dispositivo de audio, carga de la gramática por defecto, etc. Al arrancar, el motor de reconocimiento construye un *Objeto de Reconocimiento* (RO) a partir de la gramática cargada.

Cuando la habilidad se activa, mediante la llegada del evento `ASR_START` comienza a ejecutar un método en bucle denominado `proceso()` En este método, la habilidad comienza un bucle de toma de datos de audio y cómputo de estos datos. El habla del usuario se detecta cuando la señal de audio sobrepasa un cierto umbral, que es configurable. En ese momento, la habilidad emite el evento `USER_SPEAKING(1)`, que sirve para avisar al resto de la arquitectura que el usuario ha comenzado a hablar. Esto viene indicado por el parámetro `1`, que se emite junto con el evento. La herramienta entonces comienza a comparar las características acústicas del habla del usuario con los distintos árboles construidos a partir del objeto de reconocimiento (la gramática cargada).

Los resultados del reconocimiento se obtienen mediante la correspondiente llamada a las funciones de la herramienta. En la habilidad se hace una espera a que el usuario termine su enunciado. En ese momento se vuelve a emitir el evento `USER_SPEAKING(0)` (donde el parámetro `0` indica la finalización del enunciado del usuario). Una vez computada la información acústica la habilidad emite uno de estos dos eventos: `REC_OK`, si ha habido un reconocimiento satisfactorio, y `REC_FAIL`, en caso contrario.

En caso de reconocimiento, la habilidad escribe los resultados en una estructura que sube al sistema de memoria a corto plazo, y que contiene tanto los resultados literales junto con su grado de confianza, como los resultados semánticos en pares atributo-valor.

Si durante la ejecución del método `proceso( )` la habilidad hubiera recibido un evento `ASR_STOP`, el bucle pararía hasta una nueva activación mediante el evento `ASR_START`.

Por tanto `asrSkill` no solo ofrece información literal y semántica del habla del usuario (información "geométrica"), sino que también, da información, mediante el evento `USER_SPEAKING` de cuándo el usuario comienza y termina de hablar (información temporal).

5. DESARROLLO DEL SINTETIZADOR DE VOZ CON ENTONACIÓN CONTROLADA

En este capítulo, se presenta el sistema de síntesis de voz que ha sido desarrollado. El capítulo está dividido en dos partes principales. En la primera se realiza una breve introducción a las técnicas utilizadas en la síntesis de voz, desde las técnicas utilizadas para la generación del lenguaje hasta el proceso de síntesis sonora de una serie de símbolos del habla. El objetivo principal de todo sistema de síntesis de voz es el de transformar un objetivo comunicativo en una señal de voz.

En la segunda parte se expone la habilidad de síntesis de texto a voz con control prosódico. Esta habilidad integra dentro de la arquitectura de control del robot la capacidad de generar una señal acústica que sea interpretable como habla a partir de una secuencia de cadenas de texto. Además permite variar ciertos parámetros prosódicos (relacionados con la entonación) para así poder expresar con la voz cierto estado emocional.

Los principales requisitos que debe cumplir todo sintetizador de voz son:

- **Inteligibilidad**, que da cuenta del grado de comprensión sonora del texto sintetizado en voz. Está ligada a la capacidad de las palabras del enunciado sintetizado de ser entendidas por el usuario.

- **Naturalidad.** El habla natural es muy compleja. Tal y como se ha explicado en el apéndice A, en la propia comunicación hablada existe información no verbal que llega a influenciar el significado del enunciado y que está relacionada con la intención del hablante, su estado de ánimo, su actitud, su personalidad, etc.

En la práctica resulta difícil encontrar un método que cumpla simultáneamente ambos requisitos: un sintetizador de voz que sea inteligible pierde en naturalidad y viceversa. Por tanto no hay una elección única que dependerá del uso que se quiera dar a la herramienta de síntesis.

5.1. Técnica en la síntesis de voz.

5.1.1. Generación del lenguaje

Tal y como se describe en [Jurafsky and Martin, 2000], la generación del lenguaje natural (Natural Language Generation, NLG) viene a ser el proceso de construcción de enunciados hablados a partir de una intención comunicativa y un conocimiento previo.

Podría decirse que la generación de lenguaje natural es el proceso inverso al entendimiento del lenguaje natural (Natural Language Understanding, NLU). Efectivamente, tal y como se describió en la sección 4, el entendimiento del lenguaje natural parte de unas entradas acústicas cargadas de *ambigüedad*, de las cuales se realizan unas *hipótesis* devolviendo como salida una interpretación literal o semántica.

Ahora bien, en la generación del lenguaje ocurre todo lo contrario, la entrada del sistema es un objetivo comunicativo que está *bien especificado y bien formado* (no como la representación digital de las señales acústicas del habla del usuario), y de ellas, el sistema escoge entre una serie de *alternativas* que devienen en un mensaje lingüístico. Este proceso pasa por los siguientes:

1. **Selección de contenido**. Dada una especificación del objetivo comunicativo (contestar a una petición del usuario, realizar una presentación o una exposición de posibilidades, etc), el sistema debe escoger el contenido apropiado teniendo en cuenta también al tipo de usuario con el que interactúa: su nivel de conocimiento del sistema, el tiempo que lleva dialogando, el conocimiento en común que se haya establecido, etc. Tal y como se explica en el apéndice A.1.4 a partir de las ideas descritas en [Paul Watzlawick, 1967], es precisamente el aspecto relacional de la comunicación lo que clasifica el aspecto de contenido.

2. **Selección del léxico**. Es decir, la elección de unos términos apropiados con los cuales poder expresar el contenido escogido. Esta elección puede ir del caso más simple, en el que cada contenido ya tiene asociado unos términos unívocos, o bien, tener distintas alternativas para el mismo contenido. En este sentido, cada alternativa puede clasificarse según distintos grados dentro de ciertos parámetros como son el estilo (más o menos coloquial), el uso del término para otros contenidos, o el nivel de conocimiento del usuario del tema que se está compartiendo.

3. **Estructura del enunciado**. Mediante adherencia, el contenido seleccionado se agrega dentro de una frase, una cláusula o enunciado, de

un tamaño dado. Resulta importante también gestionar cómo se realizan expresiones que referencien elementos y objetos que están siendo tratados en la interacción. Para ello es común el uso de gramáticas generativas, análogas a las gramáticas de contexto libre descritas en la sección 4.1.3, que son obtenidas también, del estudio de un corpus.

4. **Estructura del discurso**. Una vez obtenidos varios enunciados, estos deben ser articulados de modo que mantengan un máximo de coherencia y una estructura discernible. Existen varios patrones de estructura del discurso. Por ejemplo, en [McTear, 2004] se nombran esquemas concretos que especifican los principales componentes del texto y que reflejan cómo el texto se organiza secuencialmente. Estos esquemas funcionan como identificación, ilustración particular, analogía o comparación.

No existe un consenso general sobre los métodos que realizan estas tareas, ni siquiera si son éstas las principales. Esto es debido a que la generación de lenguaje natural es un campo relativamente nuevo.

Los primeros métodos más simples son dos:

- **Texto enlatado**, es decir, que el texto que se sintetiza en voz está puesto directamente por el programador. Este método es el más sencillo, es fácil de implementar y es muy eficaz en el sentido comunicativo, dado que el sistema parlante dice exactamente lo que el programador ha establecido que tiene que decir. Ahora bien, la gran desventaja es que este método no permite adaptar el habla del robot a situaciones nuevas si no es con la intervención del programador.

- **Plantillas**. La parte fija del texto generado que establece el programador es una plantilla con huecos de información. El sistema completa estos huecos con cadenas que obtiene en tiempo de ejecución, bien de una base de datos, bien de la propia interacción con el usuario.

Estos dos métodos son los que se han utilizado para el sistema de síntesis de voz en el presente trabajo. Se propone como trabajo futuro el desarrollar un sistema más adaptativo a la generación de lenguaje para la expresión verbal.

5.1.2. Síntesis de texto a voz

Una vez obtenido el discurso en forma de cadena de caracteres, ésta debe ser sintetizada en una señal acústica. Para ello primero se realiza un análisis del texto en los siguientes pasos:

1. **Segmentación y normalización del texto**, en frases unitarias, cuyos bordes sonoros estén claramente definidos. En esta fase, se transcriben también acrónimos, números y abreviaturas.

2. **Análisis morfológico**. Dado que no todas las palabras posibles están presentes en los diccionarios fonéticos, este análisis se hace necesario para poder deducir la transcripción fonética de una palabra derivada de otra.

3. **Etiquetado** de ciertas partes del texto que requieran una pronunciación especial. Por ejemplo, en la distinción de homógrafos (mismas letras, pero distinta pronunciación).

4. **Modelado** de los efectos del habla continua. Una vez realizada la transcripción fonética de una palabra en una serie de fonemas, hay que tener en cuenta que la elección de dichos fonemas se realiza para la palabra aislada. Sin embargo, en el habla natural continua, dado que las palabras no están aisladas, su transcripción fonética normalizada puede sufrir modificaciones.

Mediante este análisis se obtienen una serie de fonemas a partir del texto, sin embargo, esto no resulta suficiente para la generación de voz, dado que el habla natural cuenta con otras ciertas características sonoras como son el ritmo, el cambio de entonación, el volumen, etc. Para ello se requiere una segunda fase en la que se realiza una descripción prosódica del texto. Primero se escogen los fonemas más adecuados según un modelo acústico, para después se parametrizan en unos valores que son utilizados por un método concreto de síntesis sonora.

Modelo acústico: del texto a los fonemas

El modelo acústico se definió en la sección 4.1.2, como el inventario de fonemas de una lengua, lo cual es estudiado por la fonética. En aquella sección, dicho inventario se utilizaba para encontrar la hipótesis que mejor se ajustara, dados unos observables fonéticos, con símbolos del inventario. En esta sección el proceso es inverso: dados dichos símbolos (el texto escrito) el

sistema escoge qué lista de fonemas resulta más adecuada para una correcta locución.

Este proceso se realiza gracias a unos alfabetos fonéticos que describen la pronunciación de la lengua, y que son utilizados tanto para reconocimiento como para síntesis de voz. Los diccionarios de pronunciaciones más simples constan de una lista de palabras junto a su transcripción fonética.

Algunos de estos diccionarios, para inglés, son PRONLEX[1], CMUdict[2] y CELEX[3] (que también incluye diccionario para alemán y holandés). La nomenclatura estándar más utilizada en fonética fue establecida por el Alfabeto Fonético Internacional (AFI), que trata de realizar un compendio de transcripciones de sonidos de cualquier lengua oral. Dado que los símbolos utilizados muchas veces no son cómodos desde el punto de vista informático, también se suele utilizar la nomenclatura ARPAbet que representa los fonemas en código ASCii[4].

El problema fundamental que ocurre en este nivel de transcripción es que un mismo símbolo de texto no corresponde con un solo fonema, sino que depende de su lugar dentro de la palabra o frase que ocupa. Esto es así en todas las lenguas, si bien, en español, al carecer de palabras homográficas este problema es mucho menor a otras lenguas, como el inglés o el francés, en las que la dependencia del contexto dentro de la frase hace que no exista una relación formal entre el texto y su transcripción fonética.

Para resolver esta ambigüedad en la transcripción, a partir de estos diccionarios fonéticos se construyen máquinas de estados finitos que relacionan las distintas opciones fonéticas que hay para cada carácter, permitiendo así realizar la elección del mejor alófono que se ajuste al contexto del carácter en el enunciado.

Generación de voz: de los fonemas a la señal acústica

La generación de voz implica el paso de la lista de fonemas devueltos por el análisis del texto a una representación continua parametrizada, que permita la síntesis sonora. Esta síntesis puede realizarse por varios métodos. Los principales son:

- **Síntesis articulatoria**, que modela las características físicas y fisiológicas del tracto vocal. Este método quizás es el más antiguo, llevado

[1] http://www.cs.cmu.edu/afs/cs/user/ahlen/www/pronlex.html
[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[3] http://www.ldc.upenn.edu/Catalog/readme_files/celex.readme.html
[4] http://en.wikipedia.org/wiki/Arpabet

a cabo por máquinas acústicas, por simulaciones electrónicas o bien, como se presenta en [Sawada et al., 2008], mediante una combinación de ambos.

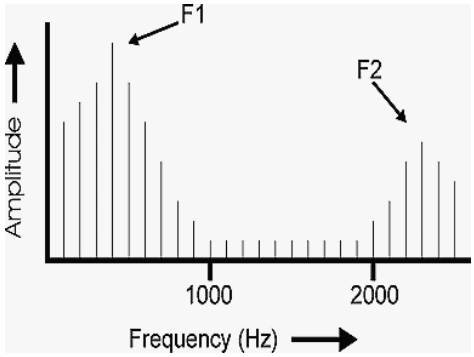
- **Habla concatenada.** Este método utiliza una base de datos de distintas unidades de habla pregrabadas. Estas unidades pueden ir del fonema a frases enteras, pero lo más típico es el uso de *difonos* o pares fonéticos. El número de estos difonos depende mucho de la lengua utilizada. Así, mientras que para el español con unos 800 difonos suele ser suficiente, el alemán requiere de unos 2500 difonos. Un algoritmo selecciona cómo unir estas unidades del modo más efectivo posible teniendo en cuenta el contexto de los fonemas dentro del enunciado.

  El grado de inteligibilidad en la síntesis concatenada puede resultar más baja que en la síntesis por formantes, debido a la aparición de ciertos chasquidos que empeoran la calidad de la señal de audio sintetizada. Sin embargo, la inteligibilidad depende mucho de la calidad del audio de las muestras utilizadas para crear la base de datos de difonos.

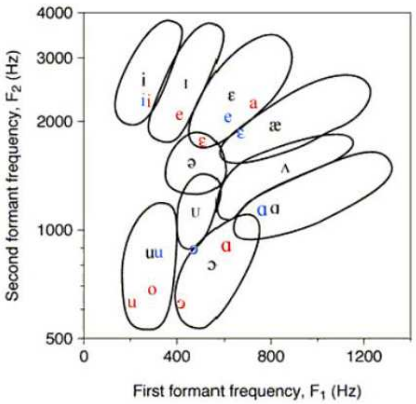
  El habla presenta gran naturalidad y parecido humano, lo cual resulta lógico teniendo en cuenta que el material de partida son muestras de audio del habla natural humana. Un inconveniente, en este sentido, es que el sistema de síntesis puede resultar muy pesado dado la cantidad de datos que conlleva: larga serie de muestras de difonos, diccionarios de pronunciación, etc.

- **Síntesis por formantes.** Modela las características acústicas de la señal. Un formante coincide con alguno de los máximos en el espectro de un fonema, por tanto está identificado por una frecuencia central y un ancho de banda o rango del pico. Ocurre que cada fonema puede ser descrito con cierta precisión mediante un patrón de formantes.

  Este método tiene la ventaja de no tener que utilizar gran cantidad de datos, respecto al habla concatenada. Además, se caracteriza por un alto grado de inteligibilidad, dado que evita la aparición de chasquidos en el sonido. Aunque este método presenta una naturalidad baja, y el habla resultante da una impresión artificial y mecánica, puede resultar de gran interés para aplicaciones que no requieran una síntesis demasiado natural, como es el caso, en el que se busca un habla con cierto carácter robótico.



(a) Dos primeros formantes de una señal



(b) Distinción entre vocales según la relación entre los dos primeros formantes

5.1.3. Control de la entonación para una expresión emotiva

La expresividad del habla está relacionada con la *prosodia*:

Definicion 5.1.1. ***Prosodia:*** *parte de la fonología dedicada al estudio de los rasgos fónicos que afectan a unidades inferiores al fonema, como las moras, o superiores a él, como las sílabas u otras secuencias de la palabra u oración*[5]

Mediante los parámetros prosódicos se describen aspectos de la pronunciación del enunciado que no están presentes en la secuencia de fonemas que se ha obtenido en los pasos anteriores. En la definición de esos "rasgos fónicos" existe información no verbal muy relevante tanto para el contenido del enunciado como para la relación que se establece entre los interlocutores (véase sección A.1.4).

La prosodia involucra los aspectos melódicos y rítmicos del habla. Es en estos aspectos donde reside la expresividad y el grado afectivo del enunciado, informando sobre el estado mental y emocional del hablante, así como su intencionalidad en el habla. Por tanto es de suma importancia en la interacción humano robot.

[5]Definición extraída del diccionario de la R.A.E

Ejemplo para una síntesis por formantes

Un estudio muy completo sobre la relación entre las características prosódicas y la expresividad emocional ha sido realizado en [Cahn, 1990] a partir de su tesis doctoral. Realiza una interesante clasificación de los principales parámetros asociados a la prosodia, en cuatro grupos, según pongan el énfasis en un aspecto fonético u otro:

- **Tono:** variación para el acento, tono central (F0), pendiente del contorno, decaimiento final, rango de variación del tono y línea de referencia del tono central.
- **Tiempo:** exageración, frecuencia de pausas entre unidades fonéticas, frecuencia de pausas dentro de una unidad fonética, velocidad del habla, radio de estrechamiento para acentos,
- **Calidad:** grado de soplido, brillo, fuerza, grado de discontinuidad en las pausas, grado de discontinuidad en los cambios de tono y trémolo o vibración. Estos parámetros suelen marcar la identidad del hablante, así como su estado afectivo.
- **Articulación:** precisión, esto es, grado de arrastre de las palabras en todo el enunciado.

Todos estos parámetros se normalizan en un rango de -10 a 10, siendo el 0 un valor neutro que anula el efecto del parámetro. La autora realiza manualmente una correlación de los valores de estos parámetros con una lista de hasta seis emociones: miedo, enfado, disgusto, alegría, tristeza y sorpresa.

El sistema es implementado mediante la herramienta comercial DECtalk[6], y una adaptación de los parámetros enumerados a otros que incorpora este sintetizador. DECtalk utiliza una síntesis por formantes, siendo muy inteligible y ofreciendo un timbre muy "robotizado". [7]

Para evaluar hasta qué punto un sintetizador expresa una emoción u otra se suele utilizar una matriz de confusión. En cada fila se coloca la emoción que quería expresarse y en las columnas el porcentaje de percepción para todas y cada una de las emociones utilizadas. Ocurre que emociones acústicamente parecidas suelen confundirse en porcentajes en torno al 30 %: tristeza con miedo y enfado con alegría.

[6] http://www.dectalk.com/

[7] El timbre de esta herramienta ha venido a hacerse muy famoso, al ser el sintetizador de habla que acompaña a Stephen Hawking.

Ejemplo para una síntesis concatenada

Otro trabajo de especial interés en el control de la entonación para una mayor expresividad, es el realizado en [Oudeyer, 2003]. El objetivo principal es el de realizar una síntesis con control en el aspecto emocional orientado para personajes de dibujos animados. El sistema no parte de un enunciado concreto sino que genera cadenas de difonos aleatoriamente, con lo que el resultado no es una frase inteligible, sino que se pone el énfasis en todo el aspecto no verbal del enunciado.

Se parte de un conjunto más estrecho de parámetros, que podemos dividir en tres categorías, según a qué aspecto fonético modifiquen:

- **Tono:** que establecen la frecuencia o tono principal ($F0$), `MEANPITCH`. Rango de variación sobre esa frecuencia, `PITCHVAR, MAXPITCH`. Y la forma del contorno para vocales y para la última palabra del enunciado: `DEFAULTCONTOUR, CONTOURLASTWORD`.
- **Tiempo:** duración media de cada difono y rango de variación `MEANDUR, DURVAR`.
- **Acentuación:** parámetros que establecen si la última palabra debe ser acentuada o no y la probabilidad de que un difono tenga o no acento, `LASTWORDACCENTED, PROBACCENT`.
- **Intensidad:** volumen total del enunciado, `VOLUME`.

Cada uno de estos parámetros se asocia con unos valores para un total de cinco emociones básicas: alegría, enfado, tristeza, calma y confort. El ajuste también se realiza manualmente mediante prueba y error.

Los parámetros son utilizados por un algoritmo determinista que calcula el tono y duración de cada fonema del enunciado, así como el volumen total de la frase. La lista de fonemas con su duración y tono se introduce en una herramienta de síntesis por concatenación denominada MBROLA[8].

[8] http://tcts.fpms.ac.be/synthesis/

MBROLA toma como entrada un fichero de texto dividido en filas. Cada fila contiene un fonema, su duración y los valores del tono o frecuencia con los que se debe sintetizar el fonema a lo largo de su duración. Así por ejemplo, el siguiente texto:

```
-
o  448 20 150 40 188 90 145
l  135 55 220
a  537 22 140 35 154 76 165 90 177
```

se configura la síntesis del enunciado *"hola"*, del siguiente modo: el primer fonema `o` durará `448 ms`. Pasado un `20 %` de su duración (`448ms * 0.20 = 90ms`) el fonema será sintetizado a la frecuencia de `150 Hz`, pasado un `40 %` de su duración, se pondrá a `188 Hz`, en el `90 %` a `145 Hz`, etcétera.

Para la síntesis, MBROLA carga una base de datos de difonos asociada a un personaje y a un idioma concreto. La herramienta construye un fichero de audio con el enunciado. En esta construcción, MBROLA admite varios parámetros relacionados con el volumen, la velocidad y el tono principal del enunciado.

En la evaluación de todo el sistema, la matriz de confusión muestra que alegría y enfado a veces se confunden, lo mismo que calma y confort.

5.2. Habilidad de síntesis de texto a voz: ettsSkill

En esta sección se explican los detalles de la habilidad que permite al robot expresarse verbalmente con cierto control en la entonación de la voz sintetizada: `ettsSkill`. Esta habilidad recibe del resto de la arquitectura, un texto con ciertas características emotivas y se encarga de sintetizarlo en voz, controladamente. Los sucesivos textos que van llegando a la habilidad se van encolando y la habilidad va sintetizando uno u otro de acuerdo a su carácter urgente.

5.2.1. Motor de síntesis de voz. Loquendo TTS-7.1

Esta habilidad, lo mismo que la de reconocimiento automático del habla, utiliza también una herramienta de Loquendo para la síntesis de voz. El método de síntesis es por concatenación, con lo que se fundamenta en una base de datos de unidades fonéticas según el idioma que se quiera utilizar.

El motor TTS que incluye Loquendo permite escoger entre varios personajes para la síntesis. Además permite crear nuevos personajes a partir de los existentes realizando cambios en parámetros prosódicos como el volumen, la frecuencia principal de habla, la velocidad de habla, etc. Estos parámetros relacionados con la prosodia de la síntesis de voz también pueden ser modificados en tiempo de ejecución. Todo ello permite implementar cierta expresividad emocional en el habla del robot.

Además, el motor de reconocimiento permite emitir ciertos sonidos no verbales como suspiros, risas, carcajadas, toses, etc, e incluso procesar el habla con algún efecto acústico como reverb, ecos, balance, etc.

5.2.2. Funcionamiento interno de la habilidad

Análogamente a `asrSkill`, la habilidad de síntesis de habla con emociones `ettsSkill` está diseñada en dos partes: una interna que se encarga de la síntesis en sí misma, y otra externa que se encarga de los mecanismos de comunicación para con el resto de la arquitectura: sistema de eventos y memoria a corto plazo.

La síntesis de habla se realiza mediante llamadas a funciones de la `API` (Application Programming Interface) del paquete `Loquendo-TTS-7.1`.

En la figura 5.1 se muestra un esquema de la habilidad y cómo se comunica con el sistema de eventos y de memoria a corto plazo. La habilidad consta de dos partes diferenciadas, una de ejecución o `proceso` y otra de gestión de

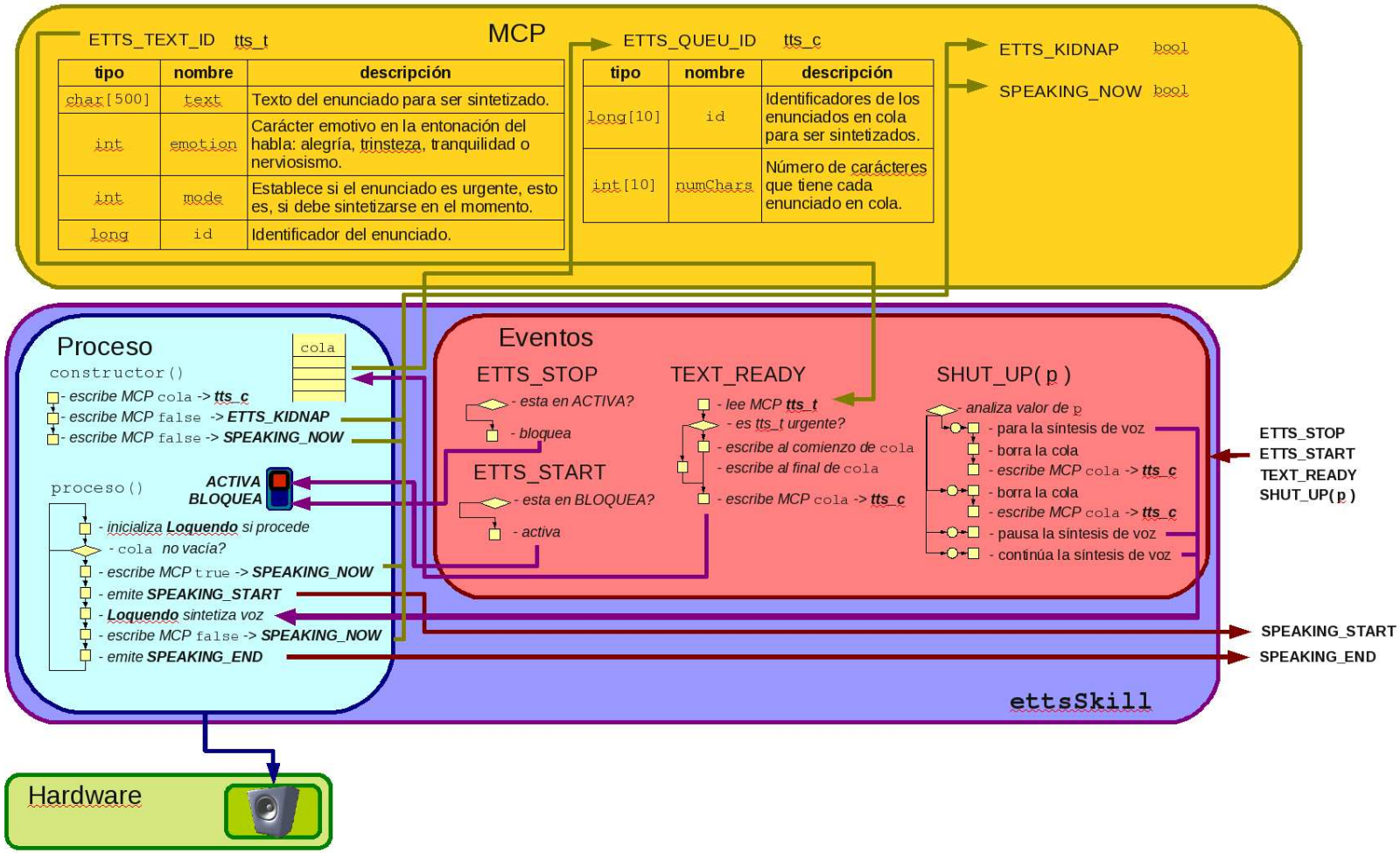


Fig. 5.1: *Esquema interno de la habilidad* `ettsSkill`

eventos. En la MCP se muestran las distintas estructuras que maneja esta habilidad:

- `ETTS_TEXT_ID`. Este dato de tipo `tts_t` contiene el texto que va a ser sintetizado en voz, así como un parámetro que establece cierta intención emocional, y otro parámetro que establece si el texto debe ser sintetizado con "urgencia" o no.

- `ETTS_QUEU_ID`. Este dato de tipo `tts_c` contiene información sobre los distintos textos para ser sintetizados en voz, que todavía permanecen en cola.

- `SPEAKING_NOW`. Este dato es un booleano que informa, al resto de habilidades de la arquitectura, cuándo el robot está hablando y cuándo está en silencio.

- `ETTS_KIDNAP`. Este dato también de tipo booleano maneja el mecanismo de "secuestro" de la habilidad. Tal mecanismo se explica con detalle más adelante.

Mecanismo de síntesis desde otra habilidad

La ejecución de la habilidad utiliza fundamentalmente una `cola` donde se van guardando los textos que se quieren sintetizar. La habilidad puede encolar un texto tanto al comienzo como al final de la cola. Cuando una habilidad cualquiera dentro de la arquitectura, llamémosla `otherSkill`, quiere sintetizar texto a voz utilizando `ettsSkill` el proceso sigue los siguientes pasos:

1. `otherSkill` comprueba que `ettsSkill` no está "secuestrada" por otra habilidad. En tal caso, la síntesis no se puede realizar.

2. Si es la primera vez que `otherSkill` va a enviar texto a la habilidad de habla, previamente, envía el evento `ETTS_START` para asegurarse de que la habilidad `ettsSkill` está activa.

3. `otherSkill` escribe en el dato `ETTS_TEXT_ID` de la MCP un dato de tipo `tts_t` donde se incluye el texto para ser sintetizado.

4. `otherSkill` envía el evento `TEXT_READY` para avisar a `ettsSkill` de que tiene un nuevo texto en la MCP.

5. `ettsSkill` recibe tal evento y ejecuta la correspondiente función manejadora. Esta función lee de la MCP el dato `ETTS_TEXT_ID` y lo escribe en la cola de textos para ser sintetizados. La escritura se realiza en el primer elemento o en el último, según un parámetro del dato `ETTS_TEXT_ID` que establece si el texto es urgente o no. De modo que si es urgente se escribe en el primer elemento de la cola y se adelanta así al resto de datos en cola.

6. `ettsSkill` en un bucle constantemente está comprobando si hay o no datos en cola. Cuando hay un dato, lo desencola y realiza la síntesis de voz, del texto contenido en el dato. Al desencolar un dato de la cola, no solo es leído, sino que es borrado de la cola.

7. Al comenzar la síntesis de voz, `ettsSkill` emite el evento `SPEAKING_START` y cuando la frase termina de ser sintetizada emite el evento `SPEAKING_END`.

8. Asimismo, mientras `ettsSkill` está realizando la síntesis de voz también pone la bandera `SPEAKING_NOW` a `true`.

Los parámetros prosódicos utilizados para la síntesis de voz dependen del campo `emotion` contenido en la estructura que alberga el texto para ser sintetizado (`ETTS_TEXT_ID`). Este campo puede tomar un valor de cuatro posibles correspondientes a cuatro aspectos emocionales del habla: alegría, tristeza, nerviosismo o tranquilidad: `HAPPINESS_EMOTION`, `SADNESS_EMOTION`, `NERVOUSNESS_EMOTION` y `TRANQUILITY_EMOTION`.

¡Silencio por favor!

En la interacción presencial entre humanos, muchas veces los turnos de diálogo se intercambian aun cuando el enunciado en marcha no ha sido terminado. Esto ocurre habitualmente, por ejemplo, cuando el hablante se ve interrumpido por el que estaba escuchando, o porque, simplemente el que escucha corta la comunicación porque se marcha o comienza a realizar otra actividad. También suele ocurrir que es el propio hablante quien realiza una interrupción a sí mismo en mitad de un enunciado. Por ejemplo, porque se percata de que lo que está diciendo no debe ser dicho, porque le llega algún recuerdo inesperado, o porque quiere repensar su enunciado, etc.

Por tanto, en el diseño de una habilidad de habla para interaccionar con el humano, hay que tener presente que tan importante como llegar a sintetizar voz y controlar la prosodia del habla, es también controlar con precisión cuándo el robot debe dejar de hablar.

ettsSkill permite controlar la interrupción de una frase en proceso de síntesis a través del manejo del evento SHUT_UP(p) Este evento es enviado por una habilidad externa. Se envía con un parámetro que da cuenta de qué mecanismo de control de la interrupción del habla quiere dicha habilidad. Este mecanismo puede ser de cuatro tipos:

- Interrupción pura. La habilidad inmediatamente interrumpe la síntesis en marcha y además, borra todas las frases encoladas para ser sintetizadas. El robot queda totalmente en silencio.

- Interrupción suave. La habilidad simplemente borra el resto de frases que deben ser sintetizadas, pero no interrumpe la frase cuya síntesis está en marcha, dejándola terminar.

- Pausa. La habilidad interrumpe inmediatamente la síntesis en marcha, pero no borra ninguna frase que haya en cola.

- Reanudar. La habilidad reanuda una frase interrumpida por una pausa.

Mecanismo de "secuestro" de la habilidad

En los procesos de interacción natural entre humanos, el acto de hablar puede considerarse circunstancial, al contrario que la escucha que es continua. Es decir, que en la expresión hablada, no solo resulta importante establecer *qué* se va a decir o *cómo*, sino que igual de importante es el *cuándo* va a producirse el habla. Así por ejemplo, hay frases, que si no son dichas en su momento carecen de sentido, y no deben ser dichas más tarde.

Uno de los problemas que plantea el carácter circunstancial o discontinuo del habla mediante esta habilidad es la centralización de dicho mecanismo. En el modelo humano, encontramos mecanismos asociativos de habla que compiten en paralelo para hacerse con el sistema fisiológico articulatorio (cuerdas vocales, lengua, labios, etc) de modo que al final una sola idea, acto comunicativo o enunciado es articulado. Análogamente, en nuestra arquitectura, diversas habilidades pueden estar queriendo hablar simultáneamente. Es necesario cierto control central. Por un lado, contamos con el buen diseño que realice el programador a la hora de hacer que una habilidad hable: utilizar frases más o menos cortas y tener en cuenta qué otras habilidades pueden estar hablando en ese instante. Por otro lado, ettsSkill cuenta con un mecanismo de "secuestro", que también requiere un correcto uso por parte del diseñador.

El mecanismo de "secuestro" permite a una habilidad de la arquitectura hacerse con el control de la habilidad de habla, `ettsSkill`. De este modo, solamente la habilidad que ha secuestrado a `ettsSkill` puede hacer uso de la síntesis de voz.

El secuestro se realiza mediante la bandera `ETTS_KIDNAP`, un dato booleano en la MCP, que avisa al resto de habilidades que `ettsSkill` está temporalmente inutilizada.

6. EDITOR VERBAL NO INTERACTIVO DE SECUENCIAS ALTERABLES

En este capítulo se describe la implementación del sistema de edición verbal de secuencias, donde todavía no se ha incorporado la parte interactiva de iniciativa mixta entre robot y humano. La incorporación de esta capacidad en el robot se explica en el capítulo 8.

Podríamos plantear nuestro problema a resolver mediante la siguiente cuestión: *¿cómo describir verbalmente al robot, un diagrama funcional representado gráficamente?* Así en este capítulo se da la respuesta a esta cuestión cuando la interacción entre el usuario y el robot es muy elemental, esto es, el lazo interactivo está abierto: no hay diálogo. El usuario se limita a construir una secuencia mediante enunciados y recibe del robot lo que éste ha entendido (a modo de eco) así como los movimientos asociados a la ejecución de la secuencia construida.

El capítulo comienza explicando cómo se llegan a identificar y definir las operaciones más elementales en la edición o construcción de una secuencia desde cero. Asimismo, a estas operaciones se asocian conjuntos de posibles enunciados hablados por el usuario. De estos conjuntos de enunciados posibles se induce el contenido de las gramáticas de contexto libre necesarias para el reconocedor automático del habla. Tal y como se ha explicado en las secciones 4.1.3 y en 4.1.4, las gramáticas construidas constan de dos tipos de contenido: contenido literal y contenido semántico. Mediante las gramáticas así construidas se une el reconocimiento de habla con las operaciones elementales de construcción de secuencias.

Más adelante se explica cómo se ha implementado el sistema completo: reconocedor y sintetizador de habla, con el constructor y ejecutor de la secuencia construida.

Por último se exponen algunos ejemplos reales de construcción de secuencias mediante voz.

6.1. Operaciones en la construcción de una secuencia

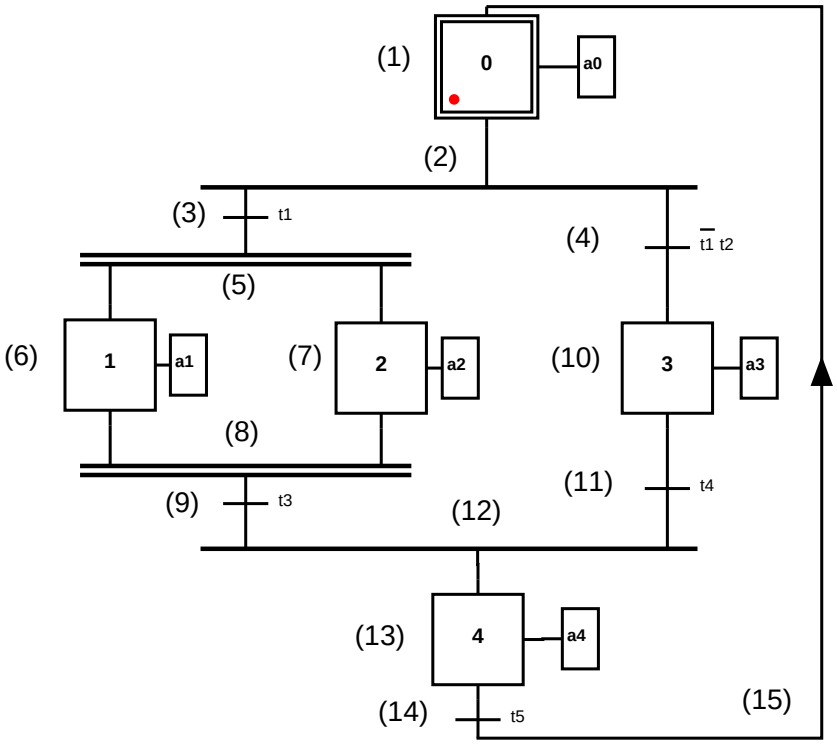


Fig. 6.1: *Ejemplo de una secuencia*

6.1.1. Construcción sucesiva de una secuencia.

Como el objetivo del presente trabajo es el de realizar un constructor verbal de secuencias, resulta importante realizar un análisis previo de cómo es el proceso de construcción de una secuencia. Para ello, suponemos que el diseñador ya tiene una idea interna más o menos clara de la secuencia que quiere construir. Por tanto vamos a analizar el proceso de construcción sin suponer que en tal proceso el diseñador realiza modificaciones, correcciones o deshace parte de la secuencia creada, y que el proceso de construcción de la secuencia es sucesivo.

Para realizar tal estudio previo, vamos a partir de la descripción verbal de la construcción, paso a paso, de una secuencia de ejemplo, para luego

realizar, mediante intuición, las conclusiones oportunas y generalizaciones del proceso.

Ejemplo

En la figura 6.1 se representa un ejemplo de una secuencia, `GRAPHET` o diagrama funcional, que incluye todas las formas o estructuras básicas descritas en B.3. Éstas son: secuencia simple, selección de secuencias en varias ramas, y varias ramas secuenciales en paralelo. Vamos a seguir paso por paso una posible manera de construir sucesivamente esta secuencia ejemplo. Así los pasos a seguir podrían ser:

1. Añadir una etapa de comienzo asociada a la acción `a0`. Esta etapa estará activa.
2. Comenzar una selección de secuencias de 2 ramas.
3. La primera rama comienza si ocurriera la condición `t1`.
4. La segunda rama comienza si ocurriera la condición `t2` y además no ocurriera la condición `t1`.
5. En la primera rama comienzan 2 ramas simultáneas.
6. En la primera de estas ramas, se añade una etapa cuya acción es `a1`.
7. En la segunda de estas ramas, se añade una etapa cuya acción es `a2`.
8. Se concluyen las ramas simultáneas.
9. Las dos ramas simultáneas concluyen en una transición que tiene asociada la condición `t3`
10. En la segunda rama de la selección, se añade, después de la condición que la activa, una etapa asociada a la acción `a3`.
11. Después se añade una transición que tiene asociada una condición `t4`.
12. Se concluye la selección de secuencias.
13. Las dos ramas selectivas concluyen en una etapa asociada a la acción `a4`.
14. Después se añade una transición que tiene asociada la condición `t4`.

15. Después se vuelve al comienzo.

Naturalmente, el orden de las distintas operaciones realizadas puede variar, también la descripción concreta de cada operación. Hemos experimentado en diversos ejemplos distintos procesos de creación de secuencias distintas, más o menos complejas, realizados por distintas personas, y podemos afirmar que el anterior ejemplo, *grosso modo*, representa, en la construcción de una secuencia, todos los elementos esenciales que pasamos a dilucidar.

Conclusiones

Lo primero que observamos al realizar una descripción explícita de cada paso atómico necesario para ir construyendo la secuencia es que esta descripción es mucho más compleja y difícil de seguir que la simple representación gráfica de la figura 6.1. Aludimos de nuevo a la necesidad del presente trabajo en realizar una descripción secuencial de una estructura gráfica.

Respecto al léxico y a las estructuras gramaticales utilizadas encontramos que el lenguaje es medianamente especializado: "rama", "acción", "condición", etc no son vocablos en general bizarros, pero sí específicos en la nomenclatura de diagramas funcionales. Por otro lado, las referencias dentro de una rama contenida en otra, etc implican cierta complejidad jerárquica, contextual o semántica que requieren de concentración, por parte del hablante, para no perder el contexto.

A partir de este ejemplo además podemos distinguir dos tipos de operaciones principales. En el primer tipo englobaríamos la adición de acciones o transiciones (nodos) También incluimos en este grupo las acciones de apertura y conclusión de varias ramas en paralelo ya sean ramas selectivas o ramas simultáneas, así como la de terminar la secuencia en un bucle. Vamos a pasar a denominar *operaciones estructurales* a este tipo de operaciones. Las denominamos así porque al añadir un nodo o una apertura o conclusión de varias ramas en paralelo, se está realizando una modificación de la estructura de la secuencia.

De este modo, podemos enumerar las operaciones estructurales en las siguientes:

- **Adición de un nodo:** que será una etapa o transición, y tendrá asociada una acción o una condición, respectivamente.

- **Apertura de varias ramas en paralelo:** en selección o en ejecución concurrente. En el primer caso sólo una rama será ejecutada de acuerdo

con la transición correspondiente. En el segundo, todas las ramas serán ejecutadas.

- **Clausura de varias ramas en paralelo:** operación que concluye en una sola rama, la estructura de distintas ramas en paralelo abierta previamente.

- **Bucle:** después de un nodo, el control puede pasar a otro nodo anterior.

En el segundo grupo de operaciones que observamos del ejemplo, englobamos un conjunto de operaciones más intrínsecas que las anteriores y que, en general, están más ocultas. Son *operaciones referenciales* o de establecimiento del "foco de atención" que se realizan sobre una parte de la secuencia creada. Las operaciones referenciales establecen dónde se va a realizar una operación estructural. De modo que las operaciones estructurales modifican la secuencia en la parte de la misma referenciada mediante una operación referencial. Así por ejemplo, a la hora de añadir un nodo, es necesario tener claro en qué parte de la secuencia ya creada va a ser añadido, etc.

De este modo, podemos enumerar las operaciones referenciales en las siguientes:

- **Referencia a un nodo**, que estará en alguna parte de la secuencia creada.

- **Referencia a una rama**, como modo de distinguir una rama de otra dentro de una estructura de varias ramas en paralelo.

- **Referencia a una estructura de varias ramas en paralelo**, para, por ejemplo, realizar una conclusión de las diversas ramas en paralelo en una sola rama.

Por tanto quedan ya identificadas las operaciones elementales de creación de una secuencia desde cero: operaciones estructurales y operaciones referenciales.

6.2. Elaboración de gramáticas semánticas para la edición verbal de secuencias

Las gramáticas semánticas no solo permiten hacer al robot sensible frente a cierto conjunto de enunciados relevantes que el usuario expresa, sino que además, permiten dotarle de una capacidad de interpretación interna de dichas locuciones (de ahí el término "semánticas").

En el diseño de estas gramáticas se ha procedido en varias etapas. Primero se ha realizado un análisis del habla natural en escenarios donde se expresa verbalmente una secuencia. De ahí se ha obtenido una clasificación de las locuciones y enunciados empleados según la información que manejan. De esta clasificación podemos elaborar expresiones regulares que forman el cuerpo literal de la gramática de contexto libre. Por último se ha realizado una lista de variables semánticas a modo de "conceptos" denominados atributos, en los que se guarda la información de los enunciados del usuario que resulta básica, fundamental o relevante para la edición de una secuencia.

Para la construcción de la parte literal de la gramática se tendrán en cuenta ciertos elementos o vocablos principales dentro de los enunciados del usuario. Pero, en el habla natural, además de este léxico específico, aparecen elementos colaterales o complementos, como son artículos, preposiciones o proposiciones complementarias y secundarias como "por favor", "primero", "Maggie", "el","la","lo", etc, que carecen de valor informativo o interesante para el objetivo del enunciado para con el robot. Estos complementos no nos dan un significado relevante de cara a la representación interna de la acción, de la condición o de la estructura secuencial que se quiere editar. Sin embargo sí que aparecen en el habla natural y, para poder hacer la gramática más robusta de cara al reconocimiento automático del habla, estos elementos secundarios van a quedan incluidos en la variable del entorno gramatical denominada `$GARBAGE` explicada en 4.1.4.

6.2.1. Descripción verbal en la construcción sucesiva de una secuencia.

A la hora de expresar verbalmente una secuencia, podemos considerar distintos niveles de complejidad que supone el entendimiento de las locuciones o enunciados del usuario, por parte del robot. Desde un punto de vista topológico, espacial o de contenido, tal como se ha explicado en A.2, el lenguaje natural tiende a ser ahorrativo en todos sus niveles fonético, sintáctico, semántico y pragmático. También puede haber distintos grados de complejí-

dad en cuanto al número de elementos referidos en un solo enunciado, y la relación entre ellos.

Consideremos por ejemplo el siguiente enunciado: "quiero que subas el brazo izquierdo a la vez que el derecho" Este enunciado aparentemente tan sencillo contiene mucha información relevante para el robot. No solo informa de ciertos deseos del usuario para con el robot, sino que hace mención a un movimiento cualitativo: subir ambos brazos al mismo tiempo, que debe ser traducido a comandos cuantitativos internos a la arquitectura interna del robot. Es decir, identificar la acción que se quiere realizar, identificar los GDL involucrados, en este caso, los motores que levantan los brazos del robot, y además levantarlos hasta cierta posición concreta y con una velocidad concreta. Toda esta información cuantitativa, en general, no es explícita en el enunciado del usuario y, por tanto, plantea el problema de cómo convertir éste enunciado en información cuantitativa interna al robot.

Ejemplos

Veamos algunos ejemplos de enunciados posibles en la edición verbal de una secuencia. En estos ejemplos toda la información que se quiere enviar al robot es verbal y, por tanto, monomodal. Es decir que no se consideran enunciados cuyo significado se extraiga con información no verbal, como el caso de enunciados que utilizan deixis anafórica (Ejem: "acércame *eso*")

1. *Maggie levanta el brazo izquierdo 90º*
2. *Cuando te toque el hombro derecho te pones a girar.*
3. *Después dime el estado de tus baterías.*
4. *Cuando termines lo primero que te dije, paras.*
5. *Después consideras cinco opciones.*
6. *Después de subir el brazo, en vez de ponerte a girar, lo bajas.*

De este modo:

1. Hace mención explícita a una acción por ejecutarse (levantar brazo izquierdo 90^{o}) dando datos cuantitativos.
2. Hace mención tanto a una condición por ocurrir (hombro derecho tocado) como a una acción por ejecutarse (comenzar a girar) Además establece una unión entre ambas.

3. Menciona una acción por ejecutarse (decir estado de las baterías) y establece una unión con una referencia implícita (*"después"*) a una acción ya mencionada.

4. A través de una etiqueta (lo primero que dije) hace mención a una condición ya referenciada. También hace referencia a una acción por incluir ("parar") en la secuencia. Además se establecen varias uniones entre condiciones y acciones mencionadas y nuevas.

5. Hace mención a una apertura de selección de cinco secuencias exclusivas entre sí.

6. Se hace una referencia a una etapa (comenzar a girar) a través de una transición que la precede (terminar de subir brazo), para luego realizar la operación de borrar dicha etapa y añadir en su lugar otra, ésta además se referencia mediante el pronombre clítico "lo", que hace referencia al brazo mencionado.

Conclusiones

Se observan las dificultades que entraña la interpretación del lenguaje natural. Para resolver estas dificultades se ha optado por considerar algunas simplificaciones y limitaciones sobre los enunciados que el sistema va a tener en consideración. Las limitaciones consideradas en el análisis del habla del usuario son:

1. Cada locución debe referirse a una sola operación estructural o referencial. Por ejemplo, no se tendrá en cuenta, la adición explícita de dos nodos en un solo enunciado.

2. El habla del usuario no debe contener ningún elemento deíctico: pronombres, anáforas, etc. Por ejemplo, no se tendrán en cuenta enunciados del tipo: "*lo* subes un poco" (uso de un pronombre) o "Llegas hasta *ahí*" (uso de un elemento deíctico).

No obstante, aun dentro de estas limitaciones, el lenguaje del usuario plantea una serie de dificultades en su reconocimiento. Por ejemplo, el usuario puede emitir un enunciado que esté incompleto; puede haber problemas de reconocimiento del enunciado completo, o parte de él; o puede perderse el hilo y la coherencia del proceso de creación verbal de la secuencia. Estos problemas se intentarán superar mediante interacción verbal o diálogo, lo cual

será explicado en el capítulo siguiente, cuando se hable de la incorporación de la capacidad interactiva en el sistema.

Siguiendo un análisis de todos los tipos posibles de enunciados en este sentido, podemos realizar una división pragmática[1] de los enunciados del usuario en los siguientes grupos:

- **Enunciados de edición de la secuencia**, que permiten añadir, sustraer y/o unir etapas y transiciones. Se distinguen los siguientes comandos:
  - Mención a una acción. Ejem: "Levanta el brazo izquierdo"
  - Mención a una condición. Ejem: "Si te toco la cabeza"
  - Comienzo de una selección de secuencias. Ejem: "Consideras 5 opciones"
  - Conclusión de una selección de secuencias. Ejem: "Terminas las 5 opciones..."
  - Comienzo de una serie de secuencias simultáneas. Ejem: "Hacer 4 cosas a la vez"
  - Conclusión de una serie de secuencias simultáneas. Ejem: "Terminas las 4 cosas..."
- **Enunciados para establecer la referencia en alguna parte de la secuencia.** Referencia a una rama dentro de una selección de secuencias o dentro de una serie de secuencias simultáneas. "En la primera opción..."
- **Enunciados para manejar la ejecución de la secuencia**, que permiten los siguientes comandos:
  - Ejecutar una secuencia ya formada. Ejem: "Ejecuta la secuencia"
  - Pausar su ejecución, manteniendo cierta información del contexto de la ejecución. Ejem: "Para la secuencia"
  - Parar y borrar la ejecución de la secuencia. Ejem: "Borra la secuencia"

[1] Esto es, referida a las consecuencias del enunciado sobre "la conducta" del robot (en este caso entendida como construcción de la secuencia)

- **Enunciados para la depuración de la secuencia**, que permiten tanto realizar cambios sobre la secuencia (como en la edición de secuencia) como modificar las propiedades de sus partes. Ejem: "Subes el brazo un poco más"

6.2.2. Gramática para las operaciones estructurales

En esta sección se justifica cómo se han construido las gramáticas semánticas encargadas de hacer accesible al usuario, desde le punto de vista verbal, las operaciones estructurales de edición de una secuencia. Se explica tanto la parte literal de la gramática como la elección de los pares semánticos atributo-valor.

Mención a una acción

Tal y como se ha explicado en 3.5, una mención a una acción ha de constar de, al menos, el nombre de la acción y de sus parámetros. Como punto de partida consideremos acciones de movimiento del robot: subir, bajar, girar, avanzar, retroceder, etc. Dependiendo de la acción misma, los parámetros considerados corresponden con el complemento directo de la acción. Éstos no pueden ser otros que partes del cuerpo del robot: brazo izquierdo/derecho, párpado, cabeza, etc.

En el caso de adición de una acción a la secuencia se consideran los siguientes valores semánticos:

- `@type`, se refiere al tipo de "acción sobre la secuencia" que en este caso es la adición de una etapa (con su acción asociada) y por eso toma el valor de "action"
- `@verb`, se refiere al nombre de la acción que se va añadir.
- `@body` y `@side`, se refieren a la parte del cuerpo del robot involucrada en la acción.

Por tanto, siguiendo la descripción formal de las gramáticas para el reconocedor de voz, la gramática referida a la adición de una acción queda de la siguiente forma:

```
$verb = ("levanta|levantas|sube|subes|abre|abres":rise
        | "baja|bajas|cierra|cierras":down
        | "gira|giras":spin\ | "mueve":move) {<@verb $value>};
```

```
$body = "cabeza":head | "ojo|ojos|parpado|parpados":eye
        | "cuello":neck | "hombro":shoulder | "brazo":arm
        | "mano":hand | "base|(cuerpo)":base {<@body $value>};
$action = $verb $body {<@type "action"};
```

Obsérvese que en la gramática final `$action`, el único campo que se coloca como obligatorio es el de `$verb` quedando como opcionales los dos campos que se refieren a la parte del cuerpo involucrada en la acción: `$body` y `$side`. Esto se hace de este modo por dos razones principales: primero porque hay acciones que no utilizan parámetros (por ejemplo, la acción "rotar") y segundo porque de este modo se permite el caso natural de que el usuario nombre una acción sin parámetros. Éstos pueden luego ser completados, en caso de necesidad, mediante diálogo tal y como se mostrará más adelante, en el siguiente capítulo.

Mención a una condición

Tal y como se explica en B.3, una condición dentro de una secuencia está asociada a una transición: cada transición tiene una función asociada que evalúa si una condición es verdadera o falsa. En dicha condición están representados, en forma de expresión algebraica, eventos provocados en el entorno del robot, así como eventos provocados en el propio robot. Esta expresión puede contener constantes, parámetros y variables del robot y de la propia secuencia. Por tanto podemos considerar, al menos, dos entornos de datos utilizables en la función de transición de una condición concreta: un entorno local a la secuencia (variables, constantes y parámetros visibles dentro de la secuencia concreta) y un entorno global al robot (variables, constantes y parámetros visibles en toda la arquitectura del robot)

Ahora bien, ¿cómo se han construido las gramáticas para que el usuario pueda añadir condiciones a la secuencia? Se parte de un dominio, conjunto discreto de condiciones posibles. Por ejemplo, consideremos las condiciones asociadas a que el usuario toque al robot en uno o varios de sus sensores de tacto. La habilidad `tactileSkill` se encargaría de enviar el evento asociado: `TOCADO` junto con un número entero que da cuenta del estado de todos los sensores. Además, la habilidad escribe en MCP el último estado de los sensores, para que cualquier parte de la arquitectura pueda acceder a dicho estado de un modo asíncrono, es decir, independientemente de si se está tocando o no al robot. Para condiciones que involucren otros eventos en el robot se procedería de modo análogo.

En el capítulo 3 se dividían las condiciones en dos tipos: explícitas e implícitas. Las condiciones explícitas que se han considerado contienen las variables semánticas `$verb`, `$body` y `$side`. La asignación semántica se realiza tal y como expresa la gramática referida a la adición de una condición explícita:

```
$verb = "toco|toque" {<@verb $value>};
$body = "cabeza":rise | "hombro|brazo":shoulder |
       | "mano" |  "espalda":back {<@body $value>};
$side = "izquierda":left | "derecha":right {<@side $value>};
$condition = [si|cuando] [te] $verb [el|la] $body $side
            {<@type "condition">};
```

Obsérvese, que las gramáticas, tanto para añadir acciones como para añadir condiciones explícitas son muy parecidas. Ambas contienen las variables `$verb`, `$body` y `$side`.

Por otro lado considerábamos condiciones implícitas que son las referidas a la terminación de una acción con final. Por ejemplo, el usuario puede decir "Levanta el brazo izquierdo" seguido de "después...". Este adverbio de tiempo implícitamente hace mención a la condición de que la acción anterior haya finalizado. En el ejemplo, que el robot haya alcanzado la posición del brazo izquierdo de "levantado". La gramática asociada a las condiciones de este tipo tiene la siguiente forma:

```
$condition = despues | luego | cuando termines [de $action];
```

Así esta gramática consideraría enunciados como "después de levantar el brazo izquierdo", "cuando termines", "cuando termines de subir la cabeza", etc.

Apertura y edición de varias ramas en paralelo.

En cualquier punto de la secuencia, el usuario puede abrir varias ramas en paralelo, con el fin de, o bien, crear una selección de una entre varias secuencias simples o bien, ejecutar varias secuencias simples en paralelo simultáneamente.

La información relevante, que necesita saber el sistema es: número de ramas que se van a crear y si la estructura que se va a crear es una selección (`selseqinit`) o son varias ramas simultáneas (`atonce`) Así la parte literal de la gramática pertinente sería:

```
$selseqinit = (abres | consideras | creas) $cardinal opciones;
$atonce = (a_la_vez [haces] $cardinal [cosas])|
          ([haces] $cardinal [cosas] a_la_vez);
```

En el primer caso, la gramática haría capaz al sistema de reconocer frases del tipo "Abres 4 opciones", "Consideras 5 posibilidades", ... En el segundo caso, la gramática permitiría frases del tipo "A la vez, haces 5 cosas" o bien "Haces 4 cosas a la vez" incluso "A la vez 5" (que es más simple pero menos natural).

Para la apertura de varias ramas en paralelo, se consideran los siguientes valores semánticos:

- `$cardinal` , valor entero correspondiente al número de ramas de la selección.
- `$type`, tipo de operación estructural sobre la secuencia. La asignación es como sigue:
  - `$type="selseq"`, si la apertura es una selección de secuencias.
  - `$type="atonce"`, si la apertura es de varias secuencias simultáneas.
  - `$type="conclution"`, si por el contrario se trata de una clausura de varias ramas.

Así, la gramática semántica para la apertura de varias ramas en paralelo queda del siguiente modo:

```
$cardinal = ( "un | uno | una":1 | "dos":2 | ... ){<@cardinal \$value>};
$selseqinit = ((( abres | consideras | creas ) $cardinal opciones) |
(puedo hacer $cardinal cosas)){<@type "selseq">};
$atonce =(a_la_vez [haces] $cardinal [cosas]){<@type "atonce">};
```

Finalización de varias ramas en paralelo

Para finalizar una estructura secuencial formada por varias ramas en paralelo la gramática construida tiene la siguiente forma:

```
$ending = ("por ultimo" | "para terminar" | "despues [de [todo|hacer]]"
         [$action]);
$conclution = (("cierras|terminas") ["las" $cardinal] ["ramas"]) |
| (["esperas a que"] $condition]);
$grammar = $ending [$conclution];
```

De este modo, el usuario puede cerrar tanto ramas de una selección de secuencias como ramas de secuencias simultáneas, con el mismo conjunto de locuciones. Naturalmente el sistema identifica un caso u otro por el contexto de edición. Es decir, si el contexto o foco de la edición reside en una rama dentro de una estructura secuencial de distintas secuencias en paralelo o si reside en una rama dentro de una selección de secuencias.

6.2.3. Gramática para las operaciones referenciales

Para editar cada rama, el sistema cuenta con el denominado *foco* o lista de nodos en contexto. Desde dicho foco se añaden partes de la secuencia. El usuario puede cambiar dicho foco mediante locuciones que menden un ordinal. Así, para editar la primera rama, el usuario tendría que decir frases como "en la primera..." Nótese que el sistema permite mover el foco de una rama a otra en cualquier momento, esto es, sin necesidad de haber terminado de editar una rama entera para poder editar otra.

La etiqueta más sencilla que permite referenciar una rama dentro de un conjunto es un numeral asociado a la rama: "primera, segunda, etc" Por tanto, el concepto semántico relevante en las locuciones cuya intención es la de colocar el foco de edición en una u otra rama para editarla es `$ordinal`.

```
$ordinal = primer(o|a):1 | segund(o|a):2 ... {<@ordinal = $value>};
```

De este modo, la gramática detectaría enunciados como "en el primero", "en el segundo", etc.

6.3. Descripción global del sistema no interactivo de edición de secuencias

En el esquema de la figura 6.2 observamos tres capas diferenciadas: una capa de hardware, otra de habilidades automáticas y otra capa donde se representa el sistema de edición de secuencias. En la capa de hardware se incluyen los sensores y actuadores del robot, que se comunican con el resto de la arquitectura a través de habilidades automáticas. Así, en la figura se han representado las habilidades automáticas `asrSkill` (véase apartado 4) y `ettsSkill` (véase apartado 5) que se encargan, respectivamente, del reconocimiento automático del habla y de la síntesis de voz. Además, se representan las habilidades encargadas de conectar con los sensores y los actuadores del robot.

El sistema de edición de secuencias consta de las siguientes partes:

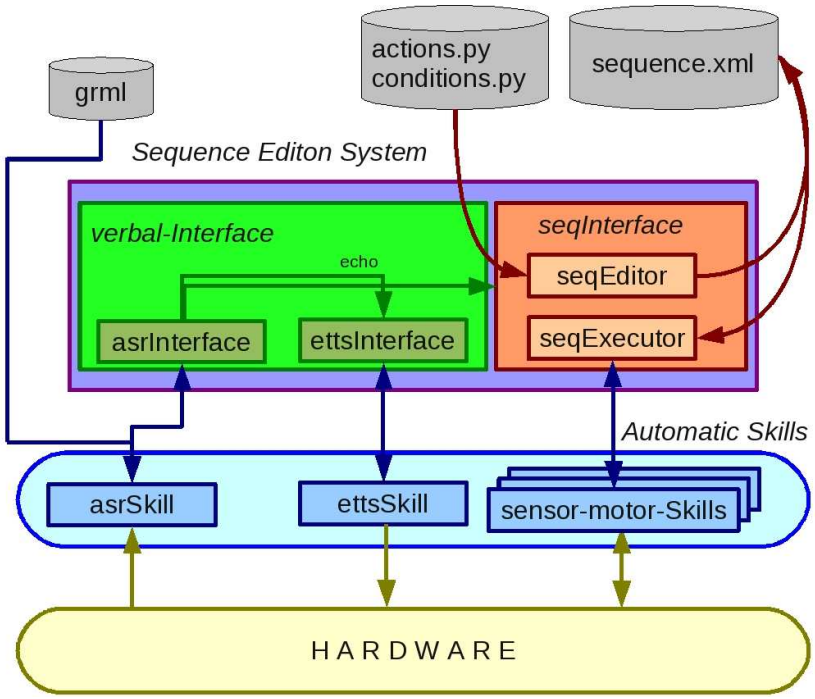


Fig. 6.2: *Esquema general del sistema de edición de secuencias*

- Interfaz de voz, que permite al sistema el acceso a las habilidades de reconocimiento y síntesis de voz.
- Intérprete semántico, que transforma la información que le llega del reconocimiento automático del habla en comandos relacionados con la edición y ejecución de la secuencia.
- Ejecutor de la secuencia, consta de un secuenciador que se encarga de cargar y ejecutar la secuencia creada.

6.3.1. Interfaz de voz. Reconocimiento y ecolalia

La habilidad de reconocimiento automático del habla carga del sistema la gramática semántica expresada en el apéndice E y cuyo diseño se ha explicado anteriormente.

Aunque en la presente explicación no se incluye un sistema de interacción entre el humano y el robot, con el fin de que el usuario pueda saber si el robot está reconociendo o no sus enunciados correctamente, se ha implementado un simple sistema que realiza un *eco* de lo que el robot entiende del enunciado

del usuario. Para ello, el resultado literal de la habilidad de reconocimiento automático del habla se reestructura y es enviado a la habilidad de síntesis de voz.

A su vez, los resultados tanto literales como semánticos del reconocimiento llegan al editor de secuencias, que realiza la correspondiente interpretación.

6.3.2. Intérprete semántico. Construcción de la secuencia

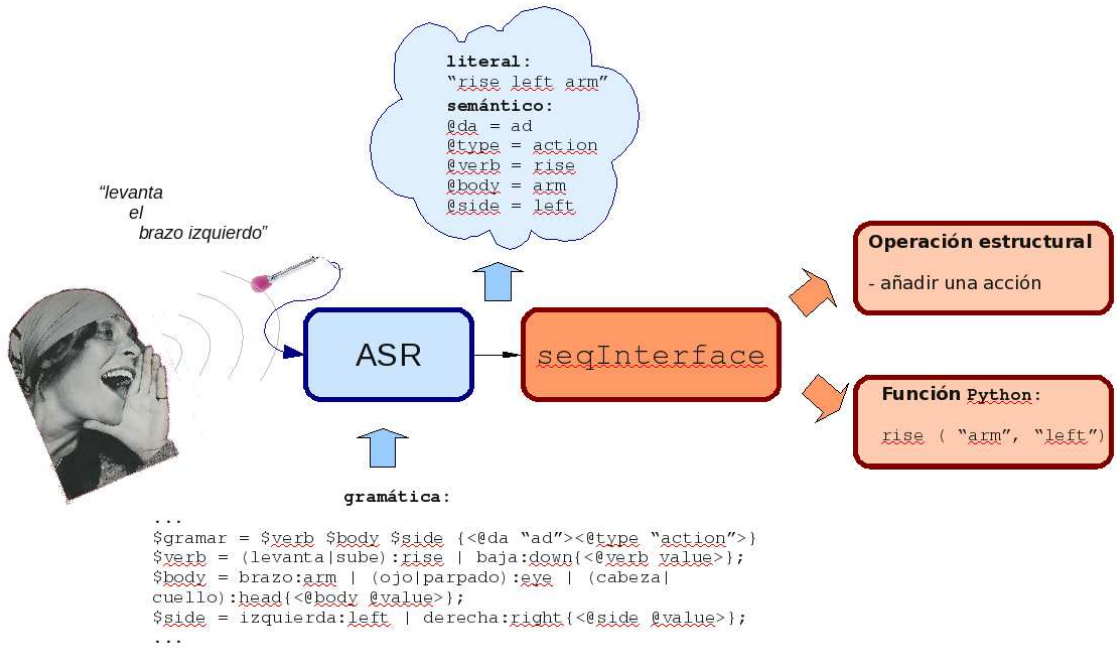


Fig. 6.3: *Esquema de funcionamiento de la interpretación semántica de un enunciado del usuario*

Los comandos estructurales y funcionales asociados a la edición o creación de la secuencia se llevan a cabo mediante comunicación con la correspondiente interfaz `seqEditor`. Esta interfaz, a través de los resultados que obtiene de la habilidad de reconocimiento automático de habla, crea la secuencia como un fichero XML, que en la figura viene representado por `sequence.xml`.

La secuencia se construye incluyendo en ella las correspondientes funciones `Python` asociadas a las acciones y condiciones que ha mencionado el

usuario. En esta sección se explica el enlace existente entre el cuerpo gramatical y los distintos elementos de la secuencia.

Interpretación de operaciones estructurales Editar una secuencia es realizar alguna *operación estructural* sobre ella: añadir un nodo o una estructura básica. Para distinguir cada comando de edición, esto es, si se trata de la adición de una acción o de una condición o de una selección de secuencias, etc, se utiliza un valor semántico común a todas las gramáticas `$type`, que permite al sistema distinguir entre distintas operaciones estructurales sobre la secuencia.

Las gramáticas permiten que el usuario acceda a todas las funcionalidades de edición de una secuencia. El sistema ahora es capaz de saber si una locución se refiere o bien a la adición de una acción o una condición o bien a la apertura o conclusión de varias ramas secuenciales en paralelo.

Interpretación de la adición de una acción En 3.5 se explicó que cada acción en el robot viene implementada por una función `Python` que se incluye, como una función de activación, en el fichero XML que representa la secuencia. La gran ventaja de utilizar gramáticas semánticas y de haberlas diseñado como se ha hecho es que el resultado semántico permite obtener directamente tanto el nombre de dicha función `Python` como sus parámetros. Para las acciones del robot se utilizó el atributo semántico `$verb` que toma valores que coinciden con el nombre de las distintas funciones `Python`. Las funciones `Python` se nombran con el mismo nombre que los posibles valores semánticos involucrados. De este modo, la traducción es inmediata, es decir, que el resultado en formato texto que devuelve el reconocedor corresponde con la propia función `Python`.

Así por ejemplo, en la figura 6.3 se representa un esquema de funcionamiento de la interpretación semántica para la acción "levantar brazo izquierdo". La función `Python` sería `rise(arm,left)` que se obtiene de los correspondientes valores semánticos. Para el resto de acciones se procede de forma análoga.

Interpretación de la adición de una condición Tal y como se ha explicado en 3.5, una condición en la secuencia queda representada por una transición y una función de evaluación, que devuelve un valor de verdadero o falso. La correspondiente función se construye igual a como se hacía para una acción. La función `Python` asociada a una condición explícita que-

da `$verb,($body,$side)$`donde los valores de estas variables semánticas se completan con su correspondiente valor dado en el reconocimiento de voz.

Así por ejemplo, la función `Python` asociada a la locución "si te toco el hombro izquierdo" es `$touch(shoulder,left)`.

En el caso de una condición implícita asociada a la terminación de una acción con final, la función `Python` tiene el mismo formato que el de dicha acción. Así, para la acción "levantar brazo izquierdo" la función `Python` sería `$rise(arm,left)` tanto para la propia acción como para comprobar que ha terminado.

Aunque de este modo se realiza una correspondencia directa entre los valores que toman las variables semánticas (conceptos) con el propio código `Python`, las funciones en `Python` están implementadas a priori. Esto viene representado en la figura 8.1 mediante los objetos de datos `actions.py` y `conditions.py`, donde están implementadas las funciones `Python` asociadas al dominio de acciones y de condiciones, respectivamente.

Ahora bien, al tratarse de un lenguaje interpretado, las propias funciones pueden modificarse en tiempo de ejecución, tal y como ya se ha comentado, previamente. Naturalmente la construcción funcional asociada a una acción o una condición no tiene porqué hacerse exclusivamente de este modo. La arquitectura permite empotrar cualquier código en `Python` en la secuencia cuando esta es editada, no solo funciones predefinidas.

6.3.3. Ejecución de la secuencia

La secuencia creada, será interpretada y ejecutada por un secuenciador mediante la interfaz `seqExecutor`. Ésta interfaz contiene un objeto especial denominado `Secuenciador`, en el que se carga la secuencia y que interpreta y ejecuta su contenido. La ejecución de la secuencia implica la ejecución de las distintas funciones incorporadas en sus etapas y transiciones. Desde el punto de vista de la arquitectura, la ejecución supone el envío y percepción de comandos a las habilidades automáticas sensorimotoras del robot; envío y recepción de eventos, así como, escritura y lectura en el sistema de memoria a corto plazo.

6.4. Construcción de secuencias mediante habla. Ejemplos

Una vez diseñadas las gramáticas semánticas, así como las funciones `Python` que representan el conjunto de acciones y condiciones posibles, la

construcción de una secuencia mediante voz se reduce a ir creando el fichero XML donde la secuencia queda implementada e ir modificándolo según se vayan sucediendo las locuciones del usuario. En esta sección se explica mediante ejemplos cómo es la construcción de una secuencia a partir de las distintas locuciones del usuario. Como se señala en B.4, a partir de cinco estructuras básicas se puede construir cualquier diagrama funcional SFC. Por tanto, los ejemplos que aquí se exponen están escogidos representando la construcción de estas estructuras básicas.

Como se explicó anteriormente, a la hora de realizar un comando estructural sobre una secuencia hay dos aspectos diferenciados en la realización del comando: uno relacionado con el foco o referencia o nodos desde los cuales se va a modificar la estructura de la secuencia, y otro relacionado con los propios elementos que se van a modificar (añadir o sustraer) Así por ejemplo, añadir un nodo (acción o condición), comenzar o finalizar una estructura de varias ramas en paralelo, crear un bucle, etc implica tener una referencia o foco a partir del cual se va a realizar la adición. En las figuras también se ha indicado dónde está el foco de la operación en cada escena de edición de la secuencia.

6.4.1. Construcción de una secuencia simple con y sin bucle

El usuario puede añadir una acción o una condición (nodo) a la secuencia simplemente nombrando la propia acción o condición. Una vez realizada la detección oportuna por el reconocedor de voz, la adición de un nodo se realiza en varios pasos. Primero, se construye el nodo en sí, completando su función de activación de deactivación o de evaluación con la correspondiente función `Python` relacionada con el resultado semántico del reconocedor. Luego, se procede a realizar las uniones pertinentes para incluir el nuevo nodo en la existente secuencia del siguiente modo:

- Si no hay más nodos, el nodo nuevo es el primero. Si se trata de una acción, esta llevará una marca. Si se trata de una condición, se procede a añadir una acción trivial para que toda secuencia cumpla con que ha de comenzar con un nodo del tipo acción.

- Si el nuevo nodo es una acción y el foco es una condición, el nuevo nodo se añade después de la condición. Si el foco es otra acción, se procede a añadir una condición trivial para mantener la consistencia estructural estándar de acción-condición-acción.

- Si el nuevo nodo es una condición, se procede análogamente al punto anterior. Si el foco es una acción, el nuevo nodo se añade después de la acción, si el foco es una condición, se añade una acción trivial por consistencia.

Nótese que el foco puede estar constituido por varias acciones y/o condiciones, como por ejemplo en la conclusión de una selección de secuencias, etc.

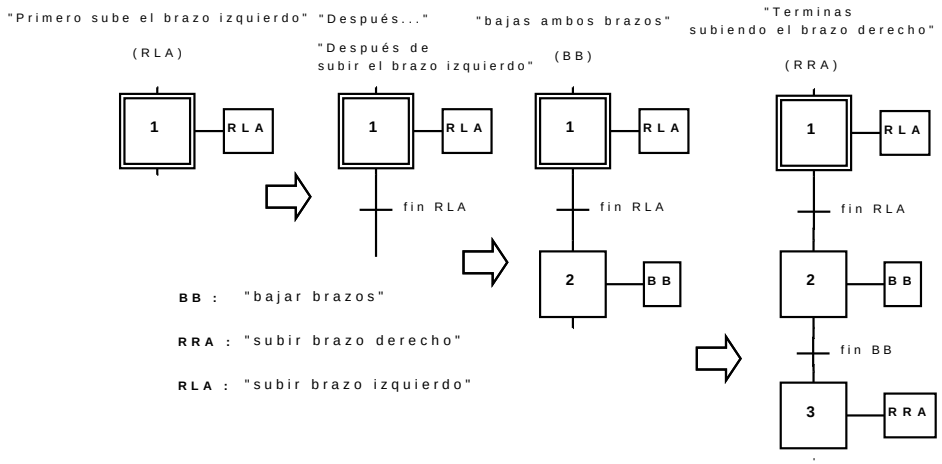


Fig. 6.4: *Ejemplo de construcción de una secuencia simple*

En la figura 6.4 puede verse cómo a cada locución corresponde un comando estructural de añadir un nodo (acción o condición) Puede observarse cómo el sistema ante la locución "después" automáticamente añade la condición `fin RLA` que comprueba si ha finalizado la acción "levantar brazo izquierdo", dado que en ese momento el foco de edición reside en dicha acción. También puede verse cómo el sistema, ante la locución "terminas subiendo el brazo derecho", añade la condición `fin BB` que comprueba si ha finalizado la acción "bajar ambos brazos" Esto se debe a que intrínsecamente ya se sabe que la última acción que se está añadiendo ("subir brazo derecho") ha de ejecutarse si y solo si la acción anterior ("bajar ambos brazos") ha finalizado.

6.4.2. Construcción de varias secuencias en paralelo

En cualquier punto de una secuencia, el usuario puede abrir varias ramas secuenciales que se ejecutan concurrentemente. Cada rama se puede editar por separado mediante locuciones que cambien el foco de una a otra ("en la primera rama...", "en la segunda...", etc) Cada rama, por tanto, ha de contener al menos una secuencia simple y debe comenzar y terminar con una acción. En la figura 6.5 puede verse un ejemplo de construcción de este tipo de estructura. En cada etapa de construcción, queda señalado en color rojo los nodos que tienen el foco de edición desde los cuales se conectan los nodos nuevos que se añaden. Obsérverse cómo a partir de las locuciones que contienen un pronombre ordinal ("primero", "segundo", etc) el foco de edición cambia. En la locución "Terminas todo..." el foco se coloca en las últimas acciones de cada rama. Si alguna rama, antes de volver a unirse con el resto, no terminara en una acción, el sistema añadiría automáticamente una acción trivial para, una vez más, mantener la consistencia de la red. En este sentido, en el ejemplo, también se añade automáticamente una transición trivial antes de la última acción "terminar de girar" Obsérvese que la acción final en una estructura de secuencias en paralelo solamente se ejecuta cuando la ejecución en todas y cada una de las ramas ha llegado al final.

6.4.3. Construcción de una selección de secuencias

A diferencia de la estructura de varias secuencias en paralelo, en la selección de secuencias, solamente una de las ramas es ejecutada. Por tanto, cada rama debe comenzar por una condición. En la figura 6.6 se representa un ejemplo de cómo es el proceso de construcción de una selección de secuencias. En el paso octavo, se observa un cambio de foco de edición de la tercera a la primera rama en la que se añade una acción más. La secuencia final esperaría uno de estos tres eventos: que el usuario toque al robot en la cabeza, en el hombro derecho o en el hombro izquierdo. Si le toca en la cabeza comenzaría a girar, si le toca en el hombro derecho levanta y baja el brazo derecho y si le toca en el hombro izquierdo levanta el brazo izquierdo. Después de que se ejecute alguna de estas tres opciones la secuencia termina con la acción de "levantar ambos brazos"

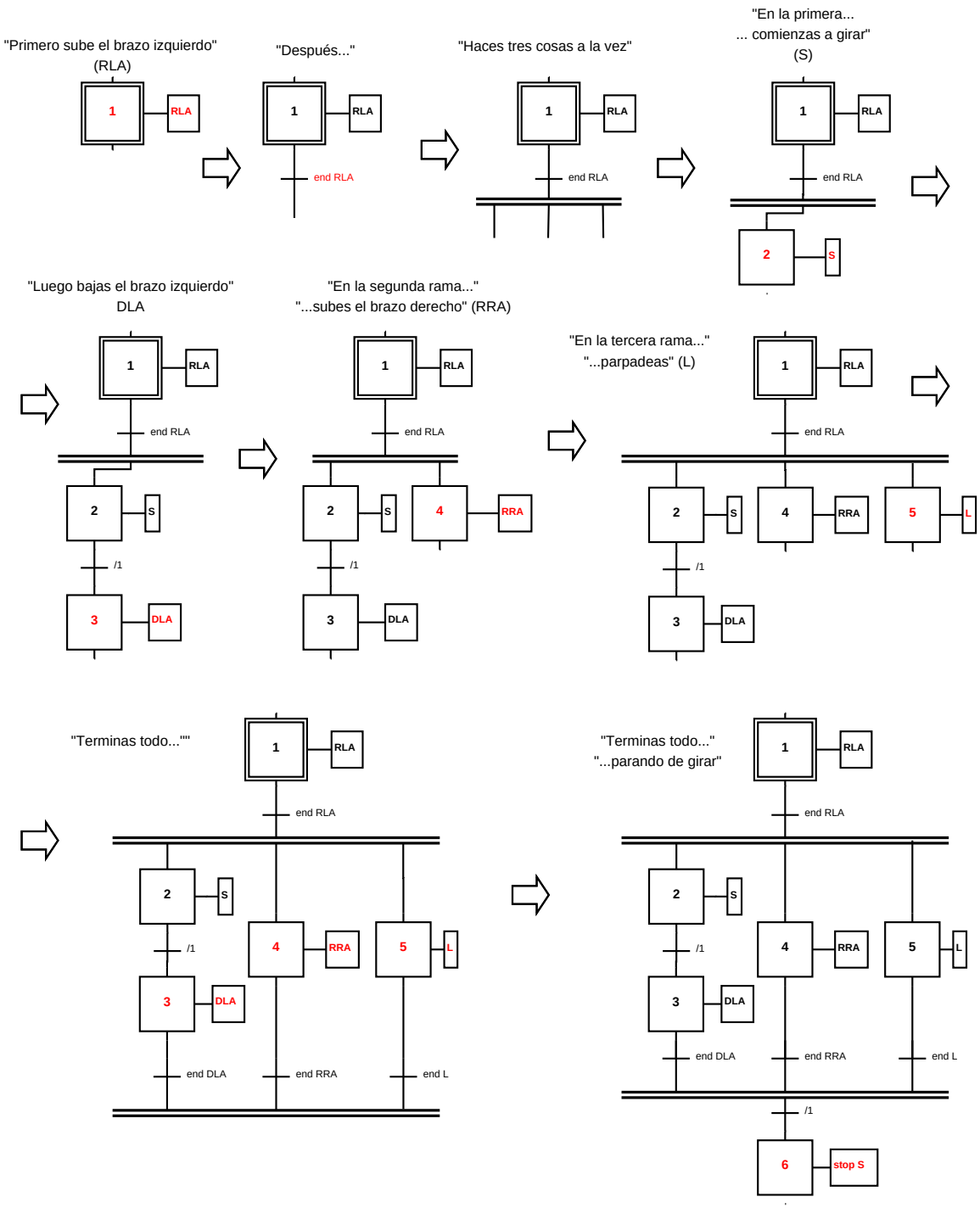


Fig. 6.5: *Proceso de creación de tres secuencias simultáneas*

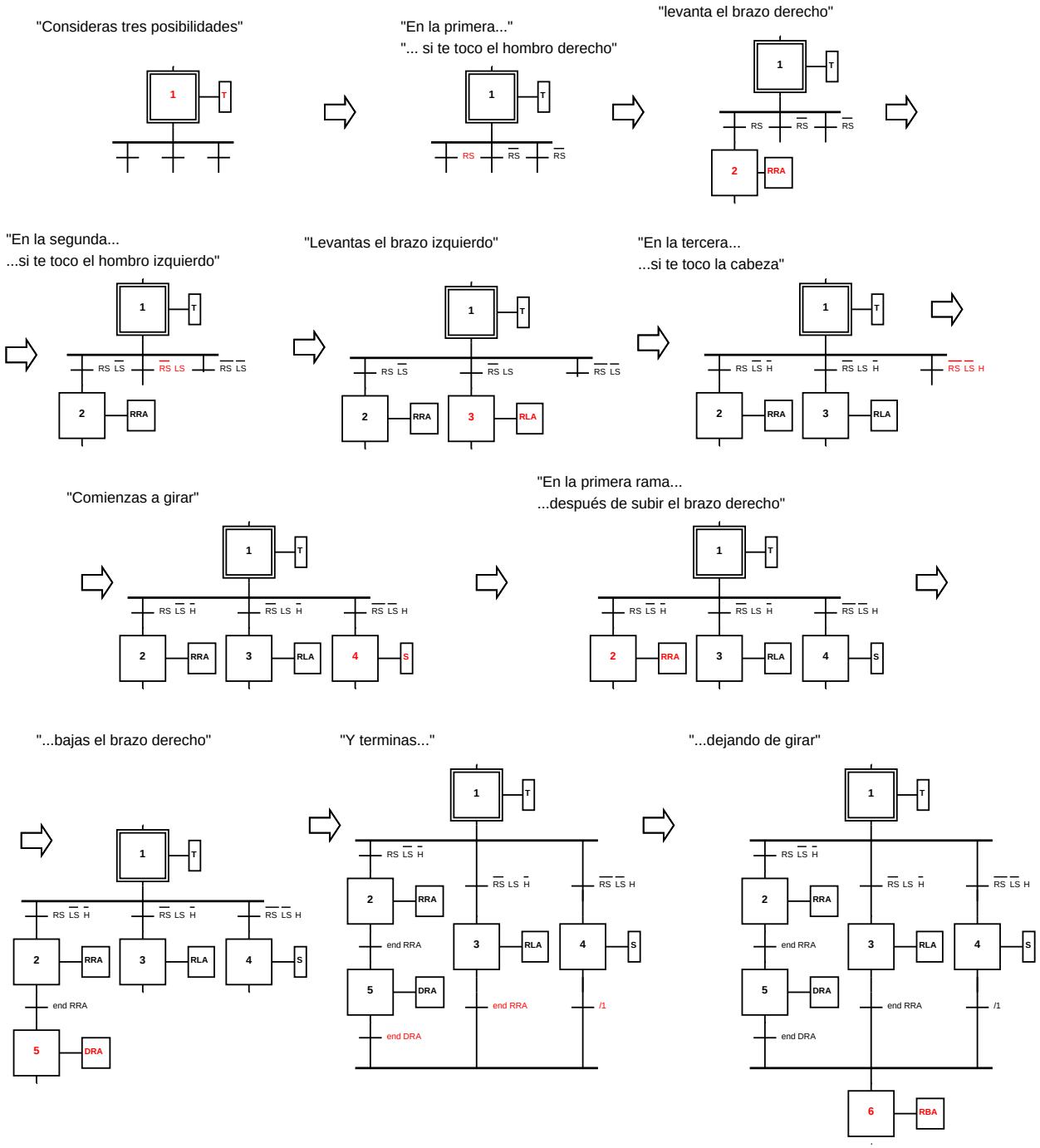


Fig. 6.6: *Proceso de creación de una selección de secuencias*

7. SISTEMA DESARROLLADO DE GESTIÓN DEL DIÁLOGO

Podemos definir "diálogo (hablado)" como un proceso de intercambio verbal (acción o efecto de hablar o platicar) entre dos o más interlocutores con algún propósito de transacción de información. Probablemente el diálogo puede considerarse como el modo de interacción más importante entre los seres humanos, en general y entre el humano y el robot en particular.

En sentido estricto o formal, el modelo de diálogo utilizado es muy distinto al diálogo natural. El intercambio de información en el modelo formal se caracteriza por utilizar un solo canal, el habla, y por ser un proceso relativamente ordenado en turnos. Distinguiríamos así a un hablante y a uno o varios oyentes. Esta concepción del diálogo es válida en el ámbito de la literatura, del cine o del teatro, y corresponde a una visión tradicional del diálogo. Sin embargo, desde los trabajos en investigación sobre la comunicación no verbal [Davis, 1971] [Birdwhistell, 1970] se llega a la conclusión de que el diálogo verbal natural no puede ser entendido sin considerar otros canales de intercambio (movimientos corporales, entonación de la voz, etc)

Por otro lado, atendiendo al aspecto dinámico de diálogos reales, también se observa que los papeles de hablante y oyente están difuminados, es decir, que el hablante al hablar también está escuchando al oyente y a sí mismo y que el oyente al escuchar también está comunicando.

Así, una de las diferencias mayores entre el diálogo natural y su modelo formal reside en cómo se realiza la articulación en uno frente a cómo se realiza en el otro. Veamos un ejemplo de un diálogo formal obtenido de una obra de teatro [1]

VLADIMIR: [*(levanta la mano)*] ¡Escucha!
ESTRAGON: No oigo nada.
VLADIMIR: ¡Pssst! [*(Escuchan.* ESTRAGON *pierde el equilibrio, casi se cae. Se agarra al brazo de* VLADIMIR*, quien se tambalea. Escuchan, apretados uno*

[1] "Esperando a Godot" (Samuel Beckett)

contra otro, mirándose fijamente a los ojos.)] Yo tampoco [*(Suspiros de alivio. Desahogo. Se separan*]

ESTRAGON: Me asustaste.

VLADIMIR: Creí que era él.

ESTRAGON: ¿Quién?

VLADIMIR: Godot.

...

Los actores en el escenario articularán en orden ensayado cada una de las frases como una partitura, no se pisarán las palabras, ni habrá dudas, pausas, frases incompletas, coletillas, ni repeticiones, autocorrecciones o sonidos que no correspondan con palabras conocidas. Al contrario los gestos y el habla estarán medidos con precisión y coordinadamente acompasados. Todo lo contrario a la articulación natural.

Aunque en la presente tesis se parte de la concepción tradicional de diálogo como "comunicaciones por turnos" la evolución del trabajo nos irá ir contemplando aspectos más relacionados con la concepción del diálogo como un proceso cooperativo coordinado.

7.1. Base teórica del sistema desarrollado

7.1.1. Características del diálogo formal

Tal y como se enumera en [Jurafsky and Martin, 2000], así como en [McTear, 2004] es posible caracterizar un diálogo mediante los siguientes conceptos:

- **Enunciado**, distinguido del concepto de frase escrita o sintagma.
- **Turnos**, entre el hablante y el que escucha.
- **Conocimiento de base**, o conocimiento compartido entre los interlocutores.
- **Sentido**, esto es, información adicional a todo enunciado, que se infiere.

A continuación se pasa a explicar con brevedad estos conceptos.

Enunciado hablado vs. frase escrita

En la sección 4 ya se introdujo este concepto de *enunciado hablado*[2] como una entidad lingüística muy semejante a lo que es una frase o sintagma escrito, pero con diferencias sustanciales que añaden cierta dificultad en el reconocimiento automático del habla. Un sintagma escrito es una forma abstracta ideal que cumple con reglas sintácticas y semánticas bien definidas, un enunciado hablado es la realización de un sintagma dentro de un contexto. Esta realización, al ser en el espacio real de diálogo interactivo y con una duración en el tiempo, presenta las siguientes características:

- Tendencia a ser corto. Es muy raro que al hablar, el emisor articule un enunciado largo. El orden de magnitud de la duración de un enunciado podría estimarse alrededor del segundo.
- Suele constar de una sola cláusula, es decir, que cada enunciado hablado forma un sentido completo.
- Sustitución de entidades referenciadas por pronombres.
- Uso de deixis anafórica y catafórica.
- Uso de coletillas, y sonidos no lingüísticos (como suspiros, chasquidos, marcas debido a dudas) De rellenos, pausas, reparaciones, repeticiones, falsos comienzos o comienzos repetidos, frases incompletas, etc.

Estas características del enunciado hablado han de tenerse en cuenta, no solo en el reconocedor automático del habla, sino en la gestión del diálogo en sí, para lo cual no hay un modelo único. La resolución de pronombres o deixis requiere que el gestor de diálogo vaya memorizando de alguna manera las entidades que van apareciendo, y alguna técnica que identifique semánticamente la referencia con la entidad. En [Grosz and Sidner, 1986], se propone dividir en tres las características del discurso. Así, se presenta un sistema que utiliza tres colas donde va guardando información referente a las entidades del diálogo tratadas, o foco de atención, información referente a las intenciones del hablante e información estructural del discurso. Esta memorización de la información permitiría aplicar algún algoritmo de asociación de referencias pronominales y deícticas con entidades del diálogo.

Por otro lado, las disfluencias comentadas juegan roles importantes en el diálogo. Así por ejemplo, el silencio como respuesta, indica una respuesta

[2]Traducido del inglés, *utterance*

negativa o una intención deliberada de no querer responder a la pregunta. En un diálogo, un silencio mayor a un segundo provoca cierta tensión. Otro papel que juega el silencio, es el de indicar el final de un turno o la incapacidad, por parte del hablante, de finalizarlo.

Asimismo, el uso de coletillas del tipo *umm*, *eee*, etc, indican que se está pensando en lo que se quiere decir, pero no se quiere ceder el turno. También pueden indicar una corrección o reparación de una parte del enunciado por el siguiente, marcando un punto de interrupción, como por ejemplo en la frase:

Quiero que levantes... [eeee] bajes el brazo derecho

Ahí, la palabra *"levantes"* es sustituida por *"bajes"* y la sustitución está marcada por una coletilla.

Intercambio de turnos

Aunque en los modelos más actuales sobre la interacción presencial humana (véase apéndice A) se defiende una idea de la interacción como una coordinación de comportamientos, en la que el papel del que habla y del oyente están difuminados, está claro que en la mayoría del tiempo de un diálogo, hay un solo interlocutor hablando, el cual, por tanto, tiene el turno en el diálogo.

Naturalmente este turno tiene una duración finita y cambia de un interlocutor a otro. La realización de dichos cambios está ligada a información tanto verbal como no verbal. Verbalmente, según el hablante va finalizando su exposición semántica, el o los oyentes van percatándose de que el turno va a finalizar, pudiendo actuar en consecuencia [Davis, 1971].

Pero también, no verbalmente hay ciertas claves lingüísticas que caracterizan el cambio de turno. Así por ejemplo, en la entonación, el tono del habla suele disminuir en frases enunciativas y aumentar en frases interrogativas cerca de la conclusión del turno. Desde el punto de vista no verbal, se realiza una división temporal del turno en distintas fases. Cuando el hablante comienza el turno, el oyente suele mostrar una *sincronía amplificada* e incluso repeticiones durante unos segundos iniciales, indicando así que está prestando atención. Después el oyente se aleja un poco y permanece inmóvil, escuchando. Pero cuando el hablante, en un momento dado, comienza a indicar que su discurso va a concluir, el oyente vuelve otra vez a moverse visiblemente imitando el ritmo del hablante, si bien no exactamente. Esta vez, hay ciertas diferencias en esos movimientos. Ocurre, que si el hablante opta por adaptar sus movimientos a lo que está proponiendo el oyente, estará cediendo así su turno. Si esto ocurre, el oyente lo percibe y exagera un poco más su nuevo

ritmo indicando que ha percibido que el turno le ha sido cedido. Justo en ese momento el oyente puede comenzar a emitir verbalmente el comienzo de su discurso.

El cambio de turno también puede no ser tan suave y ser más brusco. Por ejemplo, mediante una simple interrupción, el oyente puede pasar a ser hablante, realizando así una irrupción.

En un turno, es habitual que haya varios enunciados distintos. Así por ejemplo, en un mismo turno podría escucharse: *"Primero giras a la derecha, luego avanzas y al final levantas ambos brazos"*. Asimismo, un solo enunciado, puede estar dividido en varios turnos:

FOO: Primero giras ...
MOO: Aha, giro ...
FOO: ..., eso es. Giras, a la derecha
MOO: a la derecha ...
FOO: Luego ...
MOO: ¿Aha?
FOO: Luego avanzas.
MOO: Vale.
FOO: Y al final.
MOO: Al final ... después de avanzar.
FOO: Eso es. Al final, levantas ambos brazos.
MOO: Vale, termino levantando ambos brazos.

En este ejemplo de diálogo, el enunciado *"Primero giras a la derecha"* está dividido en dos turnos.

Establecimiento del conocimiento de base en común

El diálogo, en tanto *acto colectivo* entre, al menos, dos interlocutores, implica que ambos, para poder dialogar con efectividad deben establecer un conocimiento de base en común[3]. Definimos este conocimiento como *el conjunto de cosas que debe creer mutuamente.* En este conocimiento en común se encuentran las propias reglas de diálogo utilizadas: la lengua que se utiliza, el tono del habla, etc.

El establecimiento del conocimiento de base, suele realizarse mediante unos indicadores, también denominados continuadores que pueden clasificarse de acuerdo a una jerarquía según la intensidad del indicador:

[3]Del ingles *common ground*

1. **Indicadores de atención**, por parte del oyente. No solo indican que el oyente está atendiendo, sino que también está entendiendo y reteniendo la información que le es dada.

2. **Expresión** seguida de la contribución más relevante. Es una indicación de atención que se realiza con mayor intensidad en puntos relevantes en lo que es la exposición del hablante.

3. **Realimentación mediante asentimiento**. Este asentimiento puede ser mediante gestos o incluyendo habla, y suelen ir seguidos de una petición de confirmación por parte del hablante.

4. **Demostración** de todo o parte de lo que ha entendido. En este caso, el oyente llega a terminar razonamientos comenzados por el hablante.

Como puede observarse, en esta jerarquía el nivel de implicación verbal del oyente es creciente. Esto coincide con la idea que se propone en [Paul Watzlawick, 1967], que defiende que los interlocutores suelen recurrir al lenguaje verbal cuando el lenguaje no verbal resulta insuficiente.

Sentido solapado

Cualquier unidad lingüística con significado: lexema, enunciado, discurso, etc. posee un significado denotativo y otro connotativo. El primero es objetivo, consensuado, externo; el segundo subjetivo, individual, interno. Esta propiedad dual del significado y su relación con las distintas partes del cerebro humano ha sido estudiada con profundidad por Luria [Luria, 1980] y Vygotsky [Vygotsky, 2003].

Según Vigotsky, se entiende por sentido como una *designación de las relaciones afectas al contenido señalado en la palabra, que surgen en el proceso de la experiencia individual del sujeto en las cuales se reflejan las partes más importantes de los fenómenos designados por éste.* ([Vygotsky, 2003])

El hablante comunica más información que la que parece estar presente en un primer plano. Cualquier enunciado siempre tiene un significado literal, pero también un *sentido solapado* que surge de la experiencia individual del enunciado. Así por ejemplo, si un hablante dice a una chica que le acaba de hacer algo desagradable la siguiente frase: *"Maggie, vete de paseo, por favor."* no será igualmente interpretado que en el caso en que un usuario quisiera enviar un comando de movimiento a un robot, usando el mismo enunciado.

Por tanto, el oyente siempre selecciona, del sistema de posibles significados, los que mejor correspondan a sus necesidades y tengan para él, un

especial interés. De aquí surge el carácter subjetivo de todo pensamiento expresado, o, también denominado *significado interno.*

Tal y como se ha venido a explicar en [Luria, 1980], el sentido solapado es lo que permite que exista una relación entre elementos lingüísticos consecutivos y, por tanto, permite la transmisión de la información entera. Esto demuestra que el sentido concreto de una unidad léxica está relacionado, influenciado e incorpora el sentido de las unidades léxicas precedentes. Así por ejemplo, a la palabra *"invierno"* va asociado un significado de *sentimiento de frío* que irá incorporado a cualquier palabra que venga detrás.

De este modo, en el diálogo natural, resulta muy frecuente encontrar enunciados con doble sentido. Por ejemplo, en una sala donde la calefacción está muy alta decir *"Hace calor aquí, ¿no?"*; o en un despacho donde se suele ir a bajar a tomar café en grupo decir *"Voy a bajar a tomar un café"*, etc.

7.1.2. Actos del diálogo

La concepción del habla como acto surge al asumir que un locutor al expresarse verbalmente está realizando una acción [Ausitn, 1962] [Searle, 1970] Austin comienza introduciendo el concepto de "enunciado performativo" como una locución que no describe algo o es evaluable en términos de veracidad, sino que, realiza algo, es un acto en sí mismo. Ejemplos de enunciados performativos son: "Te nombro Queen Elizabeth", "¡Vete!", "Te acepto como esposo", etc.
Más adelante la idea de enunciado performativo evoluciona hacia el concepto de "acto ilocucionario". El análisis del enunciado entonces ya no se hace en un solo nivel lingüístico como se venía haciendo, esto es, analizando su estructura morfosintáctica, su semántica, etc, (nivel locucionario) sino que, todo enunciado, en tanto acto de habla, puede analizarse en dos niveles más: como *acto ilocucionario* y como *acto perlocucionario* El primer nivel analiza el significado real o intencionado del enunciado. El segundo analiza el efecto real del acto ilocucionario en el oyente. Una promesa es el ejemplo más claro de un acto ilocucionario, ya que, no solo no tiene sentido evaluarla en términos semánticos de veracidad, sino que tiene una clara intención transformadora en la relación entre el hablante y el oyente. Otro ejemplo de acto ilocucionario sería cuando un alcalde en una boda enuncia "os declaro marido y mujer" No está describiendo el estado o un evento del mundo, sino que la locución en sí es un enunciado performativo, esto es, un acto: la unión legal en matrimonio de dos personas, etc.

Considerando todas las dimensiones de la comunicación humana, y no solo

su canal oral representada por el diálogo en sentido estricto, del acto verbal pasaríamos al "acto comunicativo". Sigue considerándose como un acto, pero en un sentido muy amplio. Primero, porque involucra articulación de gestos y demás actos no verbales. Segundo, porque emisor y receptor no son roles exclusivos en un instante dado, sino que sus acciones forman parte de un sistema realimentado (véase apéndice C.1) Es decir, el emisor también está percibiendo al otro y a sí mismo, cuando actúa y el receptor también está actuando mientras percibe. Así, a lo largo del tiempo de su articulación, el acto comunicativo se ve influenciado por la percepción que el emisor va recibiendo sobre sí mismo, sobre el receptor y sobre el entorno.

Searle llega a realizar una caracterización de los actos de habla mediante las reglas que gobiernan su uso [Searle, 1970] Surge así el concepto de "dispositivo índice de fuerza ilocucionaria" (IFID) que corresponde con las características lingüísticas de un enunciado que permiten al oyente reconocer la fuerza ilocucionaria de un acto del habla.

Si atendemos a los cambios que un enunciado produce en el diálogo en sí llegamos al concepto de "acto de diálogo" introducido en [Bunt, 1979] Se distinguen dos partes diferenciadas en el acto de diálogo: una *función comunicativa* y un *contenido semántico* Éste último corresponde con la parte locucionaria del acto: contenido proposicional del acto de habla. La función comunicativa equivale a la parte ilocucionaria del acto de habla en términos de cambio en el contexto del diálogo, marca la intención del enunciado dentro del diálogo (ver ejemplos más adelante) Un mismo enunciado puede suponer varios actos de diálogo al mismo tiempo y análogamente, un acto de diálogo puede ser fruto de varios enunciados de algún modo sucesivos. Así mismo, para identificar qué acto de diálogo está asociado a un enunciado, muchas veces es necesario estudiar los enunciados posteriores o anteriores, para conocer qué efectos o qué causas están asociados al enunciado en cuestión.

A partir de estos conceptos surgen múltiples lenguajes de anotación de los actos de diálogo, por ejemplo, `DAMSL` (Dialogue Act Markup in Several Layer)[4] es quizás el estándar más utilizado a la hora de etiquetar un diálogo para su estudio. De este estándar surge una variación más moderna `SWDB-DAMSL`. Mediante estas etiquetas se indican ciertas características de cada enunciado en sí, de las intenciones del hablante y del contenido del enunciado. El estándar propone tres categorías de etiquetas distintas:

[4] http://www.cs.rochester.edu/research/speech/damsl/RevisedManual/

- **Estado del proceso de comunicación.** Para informar sobre la inteligibilidad del enunciado y si este está completo o no.
- **Función progresiva.** Informa de cómo el enunciado afecta a las creencias y acciones futuras de los participantes para con el diálogo.
- **Función regresiva.** Informa de las relaciones del enunciado con aspectos pasados del diálogo.

Estado del proceso de comunicación

Si bien todos los enunciados, de un modo u otro, tienen algún contenido que trata el proceso de la comunicación en sí, bajo esta categoría consideramos solamente los enunciados cuyo aspecto comunicativo resulta esencial para entender su función en el diálogo.

- **No interpretable** `[%]` El enunciado no es comprensible. Ejem: *Pero, oh..., síi...*
- **No verbal** `[x]` Sonidos no lingüísticos. Ejem: *risas, llantos*
- **Abandonado** `[%-]` Se comenzó un enunciado que luego se ha interrumpido para recomenzar otro enunciado distinto. Ejm:- *Voy a coger... esto... ¿podría coger este lápiz?*
- **Soliloquio** `[t1]` El enunciado está dirigido al propio hablante. Ejem: *¿Dónde habré puesto las gafas?*
- **Terceras partes** `[t3]` Enunciados que hablan sobre un tercero no presente en el diálogo. Ejem: *Entonces, Cristina ahora aprueba todas...*

Función progresiva

Esta categoría analiza el efecto que tiene el enunciado sobre las partes subsiguientes del diálogo, así como la influencia sobre el oyente.

- **Declaración.** Enunciados que expresan algo sobre el mundo.
  - **Afirmación sin emisión de juicio** `[sd]` Ejem: *Hoy me he levantado a las 9:00*
  - **Opinión declarativa** `[sv]` Ejem: *Hoy he madrugado muchísimo*

- **Influencia en la dirección de la acción futura del oyente.** Enunciados que tratan de influir en los actos no comunicativos que el oyente va a realizar en el futuro mediante un comando, petición, invitación, sugestión o incluso súplica, etc.
  - **Cuestión de "sí o no"** `[qy]` Cuestión cuya respuesta más directa es un "sí" o un "no" Ejem: *¿Quieres que levante un brazo? ¿Me has entendido?*
  - **Cuestión de "qué-quién-cuándo-cómo-dónde"** `[qw]` Cuestión planteada con alguna de esas partículas interrogativas. Ejem: *¿Qué hora es? ¿Qué has dicho? ¿Cuando les has visto?*
  - **Cuestión abierta** `[qo]` Pregunta cuya respuesta esperada no tiene un formato concreto. Ejem: *¿A lo mejor es que ahora sí?*
  - **Cláusula "o"** `[qrr]` Preguntas que plantean otra opción a lo que se estaba tratando. Ejem: *¿O es que hay más?*
  - **Cuestión declarativa de "sí o no"** `[qy-d]` Es una cuestión de "sí o no" que además incluye una afirmación o declaración. Ejem: *Entonces ¿tu puedes entender lo que te digo?*
  - **Cuestión declarativa de "qué-quién-cuándo-cómo-dónde"** `[qw-d]` Cuestión que además incluye una afirmación o declaración. Ejem: *¿Qué tipo de "personajillo" eres?*
  - **Cuestión coletilla** `[-g]` Para mantener el ritmo del diálogo, ceder el turno u ocultar algo que no se quiere decir. Ejem: *¿Vale? ¿De acuerdo?*
  - **Directiva de acción** `[ad]` Enunciados que tratan de provocar en el oyente algún tipo de acción no comunicativa. Ejem: *¿Por qué no levantas el brazo izquierdo? Cuando te toque la cabeza te pones a girar*
  - **Cuestión sobre un tema pasado** `[bh]` Preguntas con referencias a algo que se acaba de decir, como realimentación para establecer cierto conocimiento común. Ejem: *¿Es esto correcto? Entonces, primero hago el baile en el que me pongo a girar, ¿no?*
  - **Cuestión retórica** `[qh]` Pregunta que no necesariamente implica una respuesta. Ejem: *Qué cosas tiene la vida... Cuántas veces te lo habré repetido...*
- **Acción futura del hablante.** Enunciados donde el hablante se compromete, con mayor o menor fuerza, a una acción futura.

- **Ofertas, opiniones, cometidos** `[oo, cc, co]` Ejem: *Yo me encargo*

- **Otros.**

  - **Apertura convencional** `[fp]` Saludos y enunciados que sirven para comenzar a establecer un diálogo. Ejem: *Hola, ¿qué tal? Buenos días*
  - **Clausura convencional** `[fc]` Despedidas y enunciados cuya intención es la de finalizar un diálogo. Ejem: *Bueno, espero que nos veamos pronto.*
  - **Agradecimientos** `[ft]` Ejem: *Muchas gracias.*
  - **Disculpas** `[fa]` Ejem: *Lo siento.*

Función regresiva

Las etiquetas en esta categoría dan cuenta de la relación del enunciado con aspectos pasados del diálogo.

- **Acuerdo.** Enunciados que establecen en qué cosas y en qué medida los participantes están de acuerdo.

  - **Aceptación total** `[aa]` Ejem: *Estoy totalmente de acuerdo.*
  - **Aceptación parcial** `[aap/am]` Ejem: *Opino algo parecido.*
  - **Rechazo** `[ar]` Ejem: *Pues yo creo que no.*
  - **Paralizado** `[-h]` Ejem: *¡No me lo puedo creer!*

- **Entendimiento.** Son enunciados que sirven a los participantes para saber hasta qué punto están siendo comprendidos entre ellos.

  - **Señal de no entendimiento** `[br]` Ejem: *¿Perdón?*
  - **Repetición de la frase** `[b-m]` Ejem: *Ah, girar a la derecha*
  - **Conclusión colaborativa** `[-2]` Ejem: *¿Quién no está contribuyendo?*
  - **Acuse de recibo** `[b]` Ejem: *A-ha, okay.*
  - **Resumen o reformulación** `[bf]` Ejem: *Quieres decir que ahora no puedes.*
  - **Apreciación** `[ba]` Ejem: *Ya imagino.*

  - **Minimizar** `[bd]` Ejem: *Todo bien.*

- **Respuesta**
  - **Respuesta "si"** `[ky]` Ejem: *Sí.*
  - **Respuesta "no"** `[nn]` Ejem: *No.*
  - **Respuesta afirmativa (distinta a "si")** `[na, ny-e]` Ejem: *Así es.*
  - **Respuesta negativa (distinta a "no")** `[ng, nn-e]` Ejem: *No mucho.*
  - **Otras respuestas** `[no]` Ejem: *No lo sé.*
  - **Respuestas con matices** `[arp, nd]` Ejem: *Hombre, pues no tanto.*

Otras etiquetas

Aquí se colocan etiquetas que no tienen cavidad en alguna de las categorías anteriores.

- **Citas.** `[-q]` Partes que son de un texto o de otro hablante. Ejem: *Por sus frutos los conoceréis.*
- **Rodeos** `[h]` Ejem: *No se si lo que digo tiene algún sentido.*

7.2. Gestor del diálogo desarrollado

El sistema de diálogo consta de un motor principal que interpreta los ficheros en formato `voiceXML`. El intérprete se ha obtenido de la empresa "Commetrex" [5], que incluye un paquete de código abierto que incorpora, si no todas (como viene en la documentación de la empresa), sí la mayoría o las funcionalidades más importantes del estándar.

En el apéndice D se incluye un manual práctico de uso de dicho estándar para la implementación de un sistema de diálogos. En esta sección se explica el algoritmo `FIA` (Form Interpretation Algorithm) que dirige el flujo del diálogo según se hayan completado o no ciertos huecos de información especificados en los ficheros `voiceXML`. Por tanto, el sistema de diálogo en sí, está representado en dichos ficheros, que pueden ser modificados en tiempo de ejecución, por lo que es posible generar distintos diálogos sin necesidad de realizar modificación alguna en el sistema gestor del diálogo.

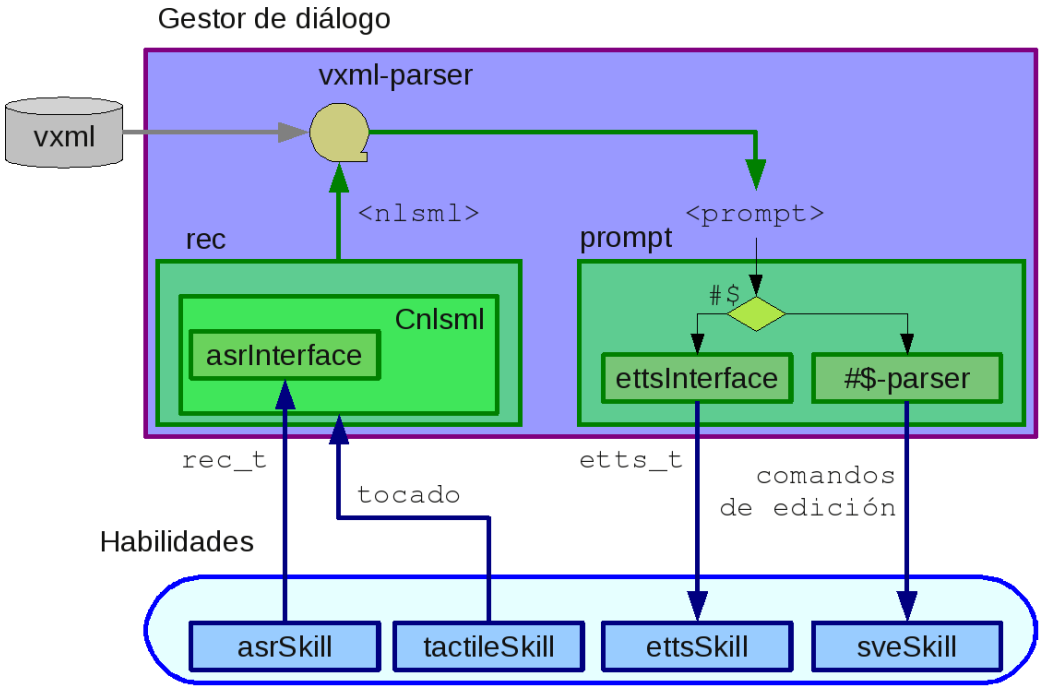


Fig. 7.1: *Esquema del manejador de diálogos*

Tal y como se puede apreciar en la figura 7.1, a este motor principal

[5] http://www.commetrex.com/products/ctmiddleware/bladewarevxml.html

se unen las interfaces de comunicación con las habilidades automáticas. El intérprete cuenta con un módulo de entrada: `rec` y otro de salida `prompt`.

Al módulo de entrada, denominado `rec`, se le ha incorporado un objeto constructor de datos en el estándar NLSML (Natural Language Semantic Markup Language). Tal y como se explica en la sección 7.3.2, este objeto se encarga de fusionar en dicho formato la información asociada a percepciones multimodales. Por ejemplo la información referida al reconocimiento automático del habla, así como los asociados a la habilidad de tacto. De este modo, al sistema de diálogo llega la información de las habilidades `asrSkill` y `tactileSkill`.

Además del resultado de reconocimiento, tal y como se ha explicado en el apartado 4, la habilidad `asrSkill` emite el evento `USER_SPEAKING` junto con un parámetro booleano, que indica cuándo el usuario ha comenzado o ha terminado de hablar. Este parámetro, es utilizado por el manejador de diálogo de dos modos. Por un lado sirve para resetear el cronómetro interno encargado de emitir el evento `noinput`. Por otro lado, el evento `USER_SPEAKING` es utilizado para interrumpir la voz del robot ante una entrada del usuario. Por tanto el sistma manejador del diálogo cuenta con la posibilidad de atender irrupciones del usuario en una interacción de iniciativa mixta.

En el módulo de salida, se incluye la posibilidad de realizar llamadas tanto al sistema de síntesis de voz como a otras partes de la arquitectura, gracias a un protocolo específico que será explicado más adelante.

7.2.1. Aspecto temporal en la implementación

Estrictamente hablando, tal y como se ha diseñado e implementado, el sistema de diálogo no gestiona exclusivamente diálogos formales, es decir, diálogos solamente hablados, donde los turnos están claramente diferenciados, etc. Existe también un control en los tiempos asociados a la interacción hablada. Son parámetros temporales relacionados con la velocidad de articulación de una frase, el tiempo de espera ante un silencio, interrupciones y cambios de turno, coordinación rítmica, etc.

Tiempo de espera

En `voiceXML`, el tiempo de espera ante una entrada del usuario se controla mediante un parámetro denominado `timeout`. Este parámetro puede establecerse de manera global para toda una aplicación, o bien, redefinirse para un diálogo (formulario o menú) concreto. Al tratarse de un parámetro que llega

al intérprete voiceXML se configura mediante el elemento <property> Así por ejemplo, la línea

```
<property name="timeout" value="5s"/>
```

configura un tiempo de espera de cinco segundos, para que el usuario diga algo. Pasado este tiempo, si el usuario no dice nada, el manejador de diálogo genera y gestiona el evento de voiceXML <noinput> El modo en que el manejador de diálogo detecta la entrada de voz del usuario es mediante el evento de la arquitectura USER_SPEAKING, que es emitido por la habilidad asrSkill (véase apartado 4) El manejo del evento noinput se configura en el propio fichero vxml.

Así por ejemplo, en la parte de código voiceXML

```
<noinput count = '1'>
  <prompt>Hay alguien aqui?</prompt>
  <prompt>\item=por favor!</prompt>
</noinput>

<noinput count = '2'>
  <reprompt/>
</noinput>

<noinput count = '3'>
  <prompt>
    \item=Breath_06 En fin, voy a esperar a que venga alguien
  </prompt>
</noinput>

<noinput count = '4'>
   <prompt>\item=Breath_01</prompt>
   <exit/>
</noinput>
```

se configuran cuatro maneras distintas de manejar la falta de entrada verbal por parte del usuario. El parámetro count dentro del elemento noinput permite especificar un manejo distinto ante distintas emisiones seguidas del evento noinput. Obsérvese, que en el ejemplo, después de cuatro esperas seguidas el intérprete saldría de la sesión de diálogo mediante el elemento <exit/>

Control en la síntesis de voz

El tiempo de una frase, el hecho de interrumpir una frase con otra de mayor prioridad, o la posibilidad de interrumpir una frase en curso, puede controlarse mediante la habilidad `ettsSkill` (véase 5.2.2) Para acceder a dichas funcionalidades desde el manejador de diálogo, se envían los comandos pertinentes, que son interpretados por el `intérprete-#$`, que es explicado en la sección 7.3.1. En el siguiente ejemplo,

```
<prompt>#emotion$HAPPINESS_EMOTION<prompt>
<prompt>#mode$URGENT_MODE<prompt>
<prompt>
 Esto es una frase alegre que interrumpe a cualquier frase en curso.
</prompt>
```

las dos primeras líneas realizan asignaciones a las propiedades `emotion` y `mode` de la habilidad de síntesis de voz `ettsSkill`. Estas propiedades configuran el tipo de entonación que se realiza en la síntesis de voz de la frase del siguiente elemento `<prompt>`.

7.3. Interacción multimodal: amplicación del sistema de gestión del diálogo

En esta sección se presentan dos ampliaciones que se han realizado en el sistema de diálogo para una interacción multimodal. Por un lado el gestor del diálogo es capaz de percibir información no verbal gracias a una interfaz que interpreta en lenguaje natural dicha información. Por otro lado, el gestor de diálogo es capaz de activar habilidades en el sistema que se encarguen de la expresión multimodal mediante un protocolo específico que se añade a la etiqueta `<prompt>` de `voiceXML`.

7.3.1. Vía de expresión multimodal: `protocolo-#$`

El módulo de salida del sistema de diálogo se denomina `prompt`. Este módulo envía el texto para ser sintetizado a la habilidad de síntesis de texto a voz `ettsSkill`. Además, a este módulo, se le ha incorporado un objeto que sirve de interfaz para el resto de la arquitectura. Como se explica en la sección 7.3.2, mediante este objeto se enviarán comandos a los módulos encargados de editar y ejecutar la secuencia. Esta es la principal función del `intérprete-#$`, si bien, su funcionalidad nos permite enviar comandos,

desde el propio fichero `vxml`, a cualquier otra habilidad automática, u otra parte de la arquitectura de control.

De este modo, el gestor de diálogo es capaz de enviar comandos a otras habilidades de la arquitectura gracias a un intérprete especial del texto asociado al elemento `<prompt>` Si bien, en la definición del estándar de `voiceXML`, este elemento sirve exclusivamente para enviar texto al sintetizador de voz del sistema, nosotros, además, hemos incorporado en el elemento `<prompt>`, la posibilidad de enviar otros datos a otras habilidades. Esto se realiza colocando los caracteres especiales # y $, en el texto asociado al elemento. Así por ejemplo, veamos como se envía el comando que abre cinco ramas secuenciales simultáneas, al editor de secuencias, desde un fichero `vxml`. El funcionamiento de este comando, se explicará más tarde en la sección 8.

```
<prompt>#atonce$5</prompt>
```

En este ejemplo, `el intérprete-#$`, hace que el gestor de diálogo no envíe el texto *"#atonce$5"* simple y llanamente a la habilidad `ettsSkill`, sino que, después de realizar la interpretación, ejecutará el comando *atonce*(5).

En todo diálogo hay dos aspectos distintos a tener en cuenta: uno que se refiere a los tiempos de interacción, *aspecto temporal*; y otro, que se refiere al contenido del diálogo, a la información conceptual, al conocimiento que se comparte, en sentido estricto. Este otro aspecto se ha venido a denominar, con más o menos acierto, *aspecto topológico o espacial*.

Así, mientras que en el aspecto temporal incluimos parámetros como los tiempos de duración de una frase, ritmo de la expresión, cambios de turno, interrupciones, irrupciones, tiempos de espera ante el silencio, etc, en el aspecto topológico se consideran los actos de diálogo realizados, el tema de la conversación, la información conceptual relacionada con el léxico y su relación sintáctica y semántica.

7.3.2. Vía de percepción multimodal: representación estándar del lenguaje natural

Si bien, el lenguaje natural es complejo y todavía sigue siendo objeto de investigación y modelado (ver A.2), el `W3C` ya ha desarrollado el borrador de un estándar basado en etiquetas para representar datos en lenguaje natural: `NLSML` (Natural Language Semantics Markup Language)[6] Este estándar permite encapsular en una sola estructura todos los pares *atributo-valor* definidos en la gramática semántica, que computa la habilidad de reconocimiento

[6] http://www.w3.org/TR/nl-spec/

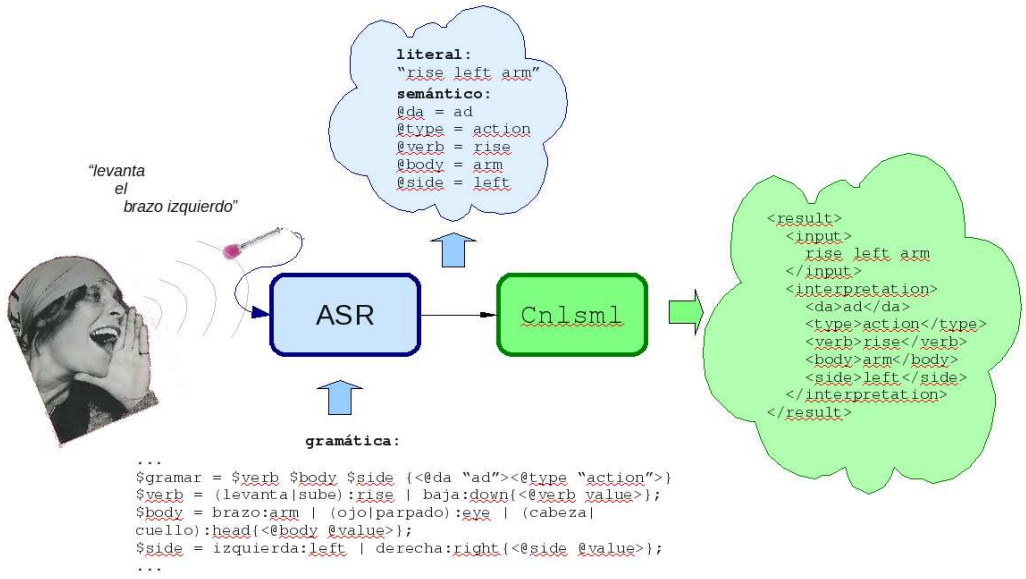


Fig. 7.2: *Sistema de creación del resultado de reconocimiento del habla en formato* `NLSML`

automático del habla. De este modo, el resultado del reconocimiento tiene una estructura, una forma bien definida, que facilita su interpretación dentro de la aplicación que utilice reconocimiento automático del habla. En el presente trabajo el resultado del reconocimiento expresado en esta estructura se ha utilizado para el sistema de diálogo, que será explicado más adelante, en la sección 7.

El estándar cuenta con las siguientes etiquetas:

- `<result>`, es el elemento raíz de la estructura `NLSML` Incluye una o más interpretaciones, que se verán más adelante.

- `<interpretation>`, define cada una de las interpretaciones, si las hubiere, del resultado del reconocimiento. Incluye los pares *atributo-valor* comentados, en la forma `<atributo>valor</atributo>` Habrá tantas de estas etiquetas `<atributo>` como atributos en el resultado semántico.

- `<model>`, permite definir un modelo estructurado de datos en forma de

formulario[7]. No se ha visto necesario utilizar esta opción.

- `<instance>`, contiene una instancia concreta de alguno de los modelos desarrollados con la etiqueta anterior.
- `<input>`, contiene el resultado literal de los símbolos terminales reconocidos.

En el presente trabajo se ha implementado un objeto específico para la construcción de la estructura en formato del estándar `NLSML`. El objeto se denomina `Cnlsml`, y se ha conectado con dos habilidades: con `asrSkill`, encargada del reconocimiento automático del habla y con `tactileSkill`, una habilidad que se encarga de recoger datos de los sensores capacitivos de la carcasa del robot.

El objeto `Cnlsml` toma los valores literales y semánticos devueltos por el reconocedor automático del habla y construye la estructura en lenguaje natural. En la figura 7.2 se muestra un ejemplo de procesamiento del enunciado "levanta el brazo izquierdo". La habilidad de reconocimiento automático del habla devuelve un resultado, literal y semántico, de acuerdo a la gramática que mejor se adapta al enunciado. Éste resultado es recogido por el objeto `Cnlsml` que crea una nueva estructura de datos siguiendo el estándar `NLSML`.

Además de tomar datos de la habilidad de reconocimiento automático del habla, también se ha conectado el objeto `Cnlsml` con la habilidad de tacto: `tactileSkill`. De este modo, al sistema de diálogo puede llegar información verbal y no verbal, abriendo así la puerta a una interacción multimodal.

La clase `Cnlsml` va a servir como interfaz de integración de la habilidad de reconocimiento automático del habla `asrSkill` en el sistema de gestión del diálogo en `voiceXML`.

[7]El tipo de modelos sigue otro estándar todavía en desarrollo denominado `XForms`

8. EDITOR VERBAL INTERACTIVO DE SECUENCIAS

En este capítulo se explica cómo se ha incorporado el sistema de diálogo al editor verbal de secuencias. Las ventajas de esta incorporación son claras. Desde el punto de vista interactivo, naturalmente, el usuario siempre va a percibir más empatía, cercanía e igualdad con un robot que dialoga. Se crea así una relación más cercana más "de igual a igual". En ese propósito de acercar el robot personal al usuario, en el sistema de diálogo han de considerarse, como se verá en este capítulo, protocolos sociales naturales, como son los saludos, las confirmaciones, etc.

Además, desde el punto de vista más técnico u orientado a llevar a cabo la tarea de crear una secuencia, el diálogo, al incluir realimentación en el sistema, aumenta la estabilidad en el desempeño de esta tarea: el robot puede preguntar sobre partes del tema de conversación que no entienda y viceversa, el robot puede dar informar al usuario sobre partes del discurso que no hayan sido comprendidas.

En esta versión del editor de secuencias, respecto a la versión explicada en el capítulo 6, se ha utilizado otra manera de acceder a las acciones del robot, utilizando clases, métodos y parámetros asociados a atributos en las gramáticas semánticas. Por tanto, se explica con detalle cómo se han diseñado estas gramáticas y su relación con las funciones `Python` que se incorporan en la secuencia.

Las gramáticas, además, ahora cuentan con material para percibir actos del diálogo, material para realizar operaciones estructurales sobre la secuencia: adición de un nodo, secuencias simultáneas, etc, y material para ejecutar, pausar, borrar, etc. la secuencia creada.

El capítulo termina con una serie de ejemplos sencillos de lo que se ha venido a denominar "diálogos protocolarios" que se relacionan de soslayo con el proceso de creación de una secuencia. En el capítulo 9 se presentan ejemplos más complejos y más centrados en la edición y ejecución de una secuencia, que por su extensión, se ha escogido dedicarle un capítulo a parte.

8.1. Escenarios elementales en el diálogo para la construcción interactiva de una secuencia

El objetivo de esta sección es el de mostrar o deducir, a través de tres ejemplos sencillos, ciertas características esenciales o elementales, que pueden aparecer en un diálogo común entre una máquina y un ser humano. De este modo se obtendrá un conjunto discreto de escenarios elementales de interacción que servirán de ayuda, más tarde, en el diseño e implementación de los distintos diálogos del sistema de edición interactiva de una secuencia.

Se han escogido tres ejemplos inventados pero realistas, que resultan ilustrativos. El primero, es un diálogo de un usuario con un sistema de reserva de billetes de vuelo. Éste corresponde a un tipo de diálogo habitual, en el que también podríamos englobar diálogos con operadores automáticos existentes de algunas empresas, en las que se requiere una atención al cliente. El segundo ejemplo, es un diálogo que podría mantener un usuario que comienza a interactuar con un robot personal ideal. El tercer ejemplo, el más cercano al presenta trabajo, es un diálogo posible que podría mantener un usuario que quiere editar una secuencia o diagrama funcional.

A modo de conclusión, la sección termina enumerando los escenarios de interacción que van a ser utilizados para la edición mediante diálogo de la secuencia.

8.1.1. Ejemplos de diálogos entre humano y máquina

Diálogo con un sistema de reserva de billetes de vuelo

El siguiente diálogo podría corresponderse perfectamente a un ejemplo de diálogo mantenido entre un usuario que quiere realizar la reserva de un billete y un sistema automático de reserva de billetes, como por ejemplo MERCURY [Seneff and Polifroni, 2000], que implementa un sistema de diálogo semejante al que se ha presentado en la sección 7.

MÁQUINA: Bienvenido al sistema automático de reserva de billetes de vuelo. Diga, con la máxima claridad, los datos del vuelo que quiere realizar. Para pedir ayuda, diga la palabra "ayuda".

USUARIO: Quiero volar de España a Japón.

MÁQUINA: Los siento, no he entendido los datos del vuelo que quiere realizar.

USUARIO: Ayuda.

MÁQUINA: Debe decir los nombres de las ciudades o aeropuerto de origen y destino, así como las fechas del vuelo de ida y del vuelo de vuelta.

USUARIO: Quiero volar de Madrid a Kyoto.

MÁQUINA: He entendido que la ciudad origen es Madrid. Diga si es correcto.

USUARIO: Lo es.

MÁQUINA: No he entendido la respuesta. Por favor, diga sí, si la ciudad origen es Madrid.

USUARIO: Si.

MÁQUINA: De acuerdo. También he entendido que la ciudad de destino es Tokyo. Diga si es correcto.

USUARIO: No. Es Kyoto.

MÁQUINA: De acuerdo. La ciudad de destino es Kyoto. Diga si es correcto.

USUARIO: Si.

MÁQUINA: De acuerdo. Ahora, diga las fechas del vuelo de ida y las del vuelo de vuelta, si quiere vuelo de vuelta.

USUARIO: Quiero viajar este Lunes a eso de las ocho, pero no se cuando quiero volver.

MÁQUINA: Lo siento. No he entendido las fechas de los vuelos. Por favor, diga el número de día y el mes de la fecha de ida.

USUARIO: Quiero viajar el Lunes día veintidós de Mayo.

MÁQUINA: De acuerdo. He entendido veintidós de Mayo. Diga si es correcto.

USUARIO: Si.

MÁQUINA: De acuerdo. Ahora, diga la fechas del vuelo de vuelta, desde Kyoto, si quiere vuelo de vuelta.

USUARIO: Solamente quiero un vuelo de ida.

MÁQUINA: No he entendido la fecha. Diga si quiere vuelo de vuelta.

USUARIO: No.

MÁQUINA: De acuerdo. Los datos del billete son, billete de solo ida desde Madrid a Kyoto el día 22 de Mayo. Diga si es correcto.

USUARIO: Si.

...

En el anterior diálogo, puede observarse cómo el sistema de diálogo va llevando al usuario por la vía correcta para completar los datos necesarios del vuelo. Estos datos, podrían ser: *ciudad de salida*, *ciudad de llegada*, *fecha de*

ida y *fecha de vuelta* si la hubiere. Por tanto, el sistema de diálogo parte de un conjunto de *atributos a los que el usuario tiene que completar* mediante diálogo. El ejemplo, además, muestra que la iniciativa la toma la máquina no el usuario, y que ésta realiza una serie de preguntas para obtener cada uno de aquellos atributos.

Otro elemento importante que aparece, es *la gestión de la falta de comprensión al usuario* por parte del sistema. Estos fallos del reconocedor son habituales, tal y como se ha visto después de haber introducido las generalidades de un sistema automático de reconocimiento de habla, en la sección 4. El alto grado de fallos del reconocimiento es patente, bien por la mala calidad de la señal de entrada, o bien porque el enunciado del usuario contiene un léxico o una estructura sintáctica no contemplada en la limitada gramática literal del reconocedor. Al final, aunque hay mal entendidos entre los interlocutores, el diálogo es estable, gracias a cómo el sistema lo va reconduciendo.

Así, en este diálogo, los distintos escenarios por los que pasan el usuario y la máquina están condicionados por los datos que ésta necesita para completar la reserva del billete, así como la petición de ayuda del usuario y preguntas de confirmación.

Diálogo de un primer encuentro entre un robot personal y una persona

En el siguiente ejemplo se muestra un primer diálogo posible entre un usuario no experto con un robot personal, que se encuentran por primera vez.

ROBOT: Hola, ¿hay alguien?
USUARIO: [*Silencio*]
ROBOT: Voy a esperar a que venga alguien ...
USUARIO: Hola Robot, ¿cómo te llamas?
ROBOT: ¡Hola! Me llamo Émpati. ¿Y tu?
USUARIO: Yo me llamo Cris.
ROBOT: ¡Anda! Pues no tengo tu nombre en mi lista. ¿Me lo podrías deletrear, porfi? Es que todavía estoy aprendiendo, y desde que me programaron todavía ha pa ...
[*El* ROBOT *se ve interrumpido por el usuario*]
USUARIO: C.
ROBOT: Si ...
USUARIO: ...R. I. S.

ROBOT: Encantada de conocerte Cris.

USUARIO: ¡Ven, acércate!

[*El* ROBOT *se acerca.*]

[*Pasa un tiempo de silencio ...*]

ROBOT: ¿Y cuántos años tienes, Cris?

USUARIO: Quince, pero voy a cumplir dieciséis muy pronto.

[*Pasa un tiempo de silencio ...*]

USUARIO: Bueno y ¿qué sabes hacer?

ROBOT: Soy un robot personal pensado para jugar con niños. ¿Quieres jugar conmigo?

USUARIO: Vale.

ROBOT: ¡Bien! Pues mira, para jugar ...

[*El* USUARIO *interrumpe de nuevo al robot.*]

USUARIO: ¡Uy! ...

ROBOT: ¿Qué pasa?

USUARIO: Que me tengo que ir. ¡Adiós!

ROBOT: Adiós, espero que nos veamos pronto.

Con este ejemplo, lo primero que se quiere es hacer hincapié en la parte dinámica o *temporal* del diálogo y cómo esta influye en la parte del diálogo más "de contenido" o *topológica.* Tanto usuario como robot llevan la iniciativa en distintos momentos (diálogo de *iniciativa mixta*) Asimismo, el usuario, a veces, interrumpe al robot, el cual deja de hablar cediendo el turno del habla a aquel. Esta posibilidad de interrupción en el diálogo podemos denominarla irrupción [1].

En el diálogo también aparece un acto comunicativo digno de mención: el usuario, en medio del propio diálogo, da la orden al robot de que se acerque. Con este acto no solo está solicitando al robot el movimiento mecánico de trasladarse hacia el usuario, sino que está obligando al robot a mantener el proceso de diálogo, tal y como marcan las implícitas normas sociales.

Además, en el diálogo se realiza una *adición de una palabra nueva* al léxico del robot, lo cual se hace, en este caso, deletreando dicha palabra.

Así, en este diálogo, los distintos escenarios por los que pasan el usuario y la máquina son: un saludo inicial con el que el diálogo comienza, una despedida con la que el diálogo termina, emisión de una orden, así como escenarios de "conocimiento mutuo" o protocolarios.

[1] del inglés: *barge-in*

Posible diálogo para la edición de una secuencia

En el siguiente ejemplo, un usuario que ya conoce ciertas funcionalidades del robot, de su sistema de diálogo y de las posibilidades que ofrece juega a "crear un baile" (traducción más natural de lo que sería una "edición verbal de secuencias mediante interacción")

MAGGIE: Hola. ¿Hay alguien ahí?
USUARIO: Hola. ¿Qué tal, Maggie?
MAGGIE: Hola. ¿Te apetece jugar conmigo a que me enseñas un baile?
USUARIO: Vale.
MAGGIE: ¡Fenomenal! Te explico. Tienes tres opciones. O bien puedes crear un baile seguido, una selección de varios bailes o bien, crear varios bailes en paralelo ...
USUARIO: Yo quiero que bailes mucho.
MAGGIE: Vale, vale. Pero ahora tienes que elegir una de las tres opciones, para poder enseñarme un baile ...
USUARIO: Vamos a crear varios bailes en paralelo.
MAGGIE: De acuerdo. ¿Cuántos bailes en paralelo quieres crear?
USUARIO: Quiero crear tres bailes en paralelo.
MAGGIE: Vale. Me has dicho que creemos tres bailes en paralelo. Cierto, ¿no?
USUARIO: Si.
MAGGIE: ¡Ya está! A continuación dime cómo es el primer baile.
USUARIO: No entiendo ...
MAGGIE: Sí. Tienes que irme diciendo las acciones y condiciones del primer baile. Puedes decirme, por ejemplo que levante un brazo, que gira la cabeza, o que después de que me toques un hombro avance, etc. Si quieres conocer todo lo que puedo hacer, di "ayuda".
USUARIO: Vale. Levanta.
MAGGIE: ¿Que quieres que levante? Dime la parte del cuerpo involucrada.
USUARIO: Levanta el brazo.
MAGGIE: Estás añadiendo la acción levantar brazo. ¿Izquierdo o derecho?
USUARIO: Derecho
MAGGIE: ¡Hecho!. Levanto el brazo derecho. ¿Después qué?
USUARIO: Después, esperas a que te toque la cabeza.
MAGGIE: Estás añadiendo la condición "tocar cabeza". Cierto, ¿no?

USUARIO: Si.
MAGGIE: ¿Y ahora?
USUARIO: Después bajas la cabeza.
MAGGIE: Estás añadiendo la acción "bajar cabeza". Cierto, ¿no?
USUARIO: Si.
MAGGIE: ¿Y ahora?
USUARIO: En el segundo baile ...
MAGGIE: Me fijo en el segundo baile ...
USUARIO: Giras a la derecha
MAGGIE: Estás añadiendo la acción "girar a la derecha" ...
[*el* USUARIO *interrumpe a* MAGGIE]
USUARIO: OK
MAGGIE: ¿Y ahora?
...
[*El* USUARIO *termina de crear toda la secuencia*]
USUARIO: Haz el baile
MAGGIE: Hago el baile. ¡Que voy!
USUARIO: Ya me he cansado de jugar. Adiós Maggie.
MAGGIE: Espero que lo hayas pasado bien. ¡Adiós! ¡Un besote!

En este ejemplo, de nuevo aparecen escenarios muy semejantes a los dos ejemplos anteriores: saludos, despedidas, peticiones de ayuda, y escenarios específicos para completar un atributo concreto, necesario para que el robot complete la tarea de crear un baile, perdón..., una secuencia. Asimismo aparecen ambas propiedades dinámicas del diálogo: la iniciativa mixta y la irrupción.

8.1.2. Conclusiones. Escenarios elementales en la edición interactiva de una secuencia

Estos ejemplos muestran un pequeño conjunto de escenarios interactivos elementales en los que podemos subdividir el proceso de diálogo que será entre el robot y el humano, cuando éste quiera editar una secuencia. Así, podemos subdividir estos escenarios elementales en dos tipos:

- **Escenarios protocolarios**, es decir, aquellos que no están orientados a la realización de una tarea específica, sino que, más bien, ayudan a establecer algún tipo de relación entre los interlocutores o a establecer

un conocimiento en común. En este conjunto entrarían escenarios del tipo, saludos, intercambio de datos personales, despedidas, peticiones de confirmación, de repetición de algo o de información relacionada con el estado comunicativo, etc. Los escenarios de este tipo, que se han tenido en cuenta en el presente trabajo son:

  - **Saludos y toma de contacto.** Comienzo del proceso interactivo.
  - **Despedidas y alejamiento.** Final del proceso interactivo.
  - **Ayuda.** El usuario puede encontrarse perdido en algún punto de la interacción y pide ayuda para conocer las opciones comunicativas que tiene.
  - **Pérdida de la coherencia en el diálogo.** Donde se incluyen escenarios donde o bien el usuario no sigue un *tema* establecido cooperativamente con anterioridad, o bien, propone un tema que no puede ser entendido por el robot, y por tanto, no puede ser seguido. Esto suele solucionarse mediante diálogos de confirmación.
  - **Respuesta concreta**, ante una pregunta que no es de tipo sí o no, sino de tipo quién, cuál, qué, cuándo, etc.

- **Escenarios exclusivos de edición de la secuencia**, es decir, aquellos orientados a la realización de una tarea específica, que en el caso que ocupa, no es sino la edición verbal interactiva de una secuencia. Estos son:

  - **Edición de un nodo:** acción o condición.
  - **Edición de una estructura secuencial.** En un punto de la secuencia puede editarse una estructura básica de un diagrama funcional: secuencia básica, selección de secuencias y secuencias simultáneas.
  - **Referencia a una parte de la secuencia.** Operaciones referenciales que establecen el foco de atención en unas etapas o transiciones concretas, en la edición de la secuencia.

Por tanto, el problema del diseño concreto del sistema de diálogo queda acotado en un conjunto discreto de escenarios, o subdiálogos. El objetivo entonces, se centra en *resolver cómo ha de realizarse el diálogo para cada uno de estos escenarios y cómo se ha de realizar el paso de un escenario a otro, atendiendo tanto al aspecto temporal del diálogo como a su aspecto topológico.*

Naturalmente, este primer conjunto de escenarios esenciales no representan todos los posibles. El usuario puede cambiar a un tema de conversación distinto, o querer realizar con el robot otras actividades, etc. Ahora bien, el método escogido que se presenta en el que se divide el sistema de interacción en escenarios elementales, permite fácilmente añadir nuevos escenarios según los requisitos del sistema, incluso en tiempo de ejecución. Además, el paradigma de diseño que se realiza para cada uno de estos escenarios, una vez llevado a cabo, puede ser utilizado para resolver el diálogo en cualquier otro nuevo escenario.

8.2. Reglas semánticas para escenarios protocolarios

Las gramáticas de estos escenarios constan de dos partes: una parte *literal* y otra *semántica*. La primera establece el conjunto lenguaje formal, o conjunto de enunciados que el reconocedor automático del habla va a permitir percibir (sobre la relación entre gramáticas de contexto libre y lenguaje formal véase el apéndice C La parte semántica permite asignar distintos valores para el acto de diálogo, que se define como un atributo semántico, así como asignaciones a otros atributos que se explican más adelante.

```
$fp = hola [magui]{<@da "fp">};
$fc = adios [magui]{<@da "fc">};
$e = = [quiero] (sal|termina|salir|terminar) {<@da "e">};
$polite = porfi | por_favor;
$h = [$polite] (([necesito] ayuda) | (podrias ayudarme) | ayudame |
 (no [te] entiendo [nada]) | (echame una mano)) [$polite]{<@da "h">};
$ky = (okei|vale|de_acuerdo|si|afirmativo|correcto|
       [que|muy] bien)[magui]{<@da "ky">};
$nn = (no|negativo|incorrecto|[que|muy] mal)[magui]{<@da "nn">};
```

De este modo se hace corresponder el saludo inicial con el acto de diálogo `fp` (apertura convencional); la despedida con los actos de diálogo `fc` (clausura convencional) y `e` (salida del diálogo); la petición de ayuda contextual se corresponde con el acto de diálogo `h` (duda); y las respuestas que permiten realizar una confirmación o negarla son, respectivamente, `ky` y `nn`.

```
$cardinal = ( un:1 | uno:1 | una:1 | dos:2 | tres:3 | cuatro:4
            | cinco:5 | seis:6 | siete:7 | ocho:8 | nueve:9  )
            {<@cardinal $value>};
$ordinal = (primer:1|primero:1|primera:1 | segundo:2|segunda:2
  |tercero:3|tercera:3 | cuarto:4|cuarta:4 | quinto:5|quinta:5
  | sexto:6|sexta:6 | septimo:7|septima:7 | octavo:8|octava:8
  | noveno:9|novena:9 ){<@ordinal $value>};
$cardinalEx = $cardinal {<@da "nd"><@obj "cardinal">};
$ordinalEx = $ordinal {<@da "nd"><@obj "ordinal">};
$bodyEx = ( $eye | $arm | $neck | $hand | $back )
          {<@da "nd"><@obj "body">};
$sideEx = $axis {<@da "nd"><@obj "axis">};
$actionEx = accion {<@da "nd"><@obj "action">};
$conditionEx = condicion {<@da "nd"><@obj "condition">};

$nd = $bodyEx | $sideEx | $actionEx | $conditionEx
      |cardinalEx | ordinalEx;
```

En un momento dado, dentro del diálogo entre el usuario y el robot, éste puede preguntar a aquél información concreta que no haya sido enunciada o, simplemente, que no haya comprendido bien. La percepción de las respuestas del usuario en este sentido se incluyen en el acto de diálogo `nd` (respuesta concreta) En la implementación de la gramática, además, se ha definido el atributo semántico `obj` que representa el objeto de la respuesta concreta.

El atributo semántico `da` se completa así con el valor que corresponda según el enunciado del usuario, y éste, a su vez, pasa a asignarse al campo "`da`" dentro del formulario del fichero en `voiceXML`, el fichero que implementa el diálogo en cuestión.

8.3. Reglas semánticas para escenarios de edición

8.3.1. Adición de una acción

En la adición de una acción en la secuencia, el acto de diálogo involucrado es "`ad`", es decir, una directiva de acción. Además, el tipo de acto de diálogo se asigna a "`action`", para indicar que la directiva de acción comunicativa es la adición de una acción en la secuencia. La adición de una acción, además, involucra otros campos relacionados con la acción concreta que se quiere añadir, como son el verbo, la parte del cuerpo involucrada, etc.

En el editor verbal interactivo de secuencias, la construcción de las funciones en `Python` que son incluidas en la secuencia se realiza directamente a partir del resultado semántico que devuelve el reconocedor automático del habla (sobre la representación en `Python` de las acciones y condiciones del robot véase 3.5.3). En esta sección se explica cómo se han diseñado e implementado las reglas que conforman la gramática semántica que utiliza el reconocedor para el caso en que se quiere añadir una acción.

Para implementar esta gramática es necesario realizar un estudio previo de cómo se van a mencionar las acciones. Se parte del dominio inicial de acciones[2], explicado en la sección 3.5. Para cada acción se considera la información elemental necesaria que, desde el punto de vista funcional, denota totalmente la acción. Por ejemplo, para identificar la acción "levantar el brazo izquierdo" sería necesario identificar la parte del cuerpo involucrada, en este caso el brazo izquierdo, y la posición a la que se quiere mover, en este caso "levantar" implica que el movimiento es hacia arriba.

Una vez identificados todos los parámetros que definen cada acción se procede a identificar los enunciados en lenguaje natural que mencionan dicha información. En las tablas 8.1 se representa la información verbal esencial y necesaria para identificar cada acción dentro del dominio inicial de acciones consideradas. Cada acción va seguida de la parte de código en `Python` que la ejecutaría. Los valores de posición dentro de los métodos de movimiento `move` y `spin`, como son `ArmTop`, `NeckRight`, `BaseBack`, etc, se consideran constantes del sistema.

En dichas tablas puede realizarse fácilmente una asociación entre cada parte del enunciado verbal con cada parte del método en código `Python`. Por ejemplo, para la acción "bajar cabeza", el verbo "bajar" implica que el método debe ser "mover a posición baja", es decir, "`move(...Down)`" y el hecho de

[2]La metodología utilizada que se va a explicar resulta igualmente aplicable a una ampliación de este dominio.

Sintáxis I			Métodos
levantar	brazo	izquierdo/derecho	`left/rightArm.move(ArmTop)`
	cabeza	-	`Neck.move(NeckTop)`
bajar	brazo	izquierdo/derecho	`left/rightArm.move(ArmDown)`
	cabeza	-	`Neck.move(NeckDown)`

Sintáxis II			Métodos
girar	cabeza	izquierda	`Neck.spin(NeckLeft)`
		derecha	`Neck.spin(NeckRight)`
	a la izquierda		`Base.spin(BaseLeft)`
	a la derecha		`Base.spin(BaseRight)`

Sintáxis III	Métodos
Avanzar	`Base.move(BaseFront)`
Retroceder	`Base.move(BaseBack)`

Tab. 8.1: *Estructuras sintácticas en la mención de una acción del dominio*

que sea la cabeza lo que se mueve, implica que el objeto debe ser de tipo `Neck`.

En este sentido han surgido dos problemas principales inherentes al lenguaje natural. El primero, que una palabra en lenguaje natural puede significar cosas distintas según su función sintáctica. Por ejemplo, la palabra "izquierda" tiene funciones distintas en la acción "levantar brazo izquierdo" que en la acción "gira la cabeza a la izquierda". Mientras en el primer caso está complementando al nombre "brazo" y por tanto señalando qué parte del cuerpo, o qué articulación está involucrada en la acción, en el segundo caso está complementando al verbo "girar" diciendo hacia dónde tiene que realizarse el giro.

El otro problema es que asimismo, cada palabra en lenguaje natural puede informar al mismo tiempo de varios parámetros que definen una acción. Por ejemplo, el enunciado "avanza", no sólo indica que la acción es de movimiento, sino que involucra a todo el robot (y por tanto a los motores de las ruedas) y que el movimiento debe realizarse hacia adelante.

Estos problemas se resuelven automáticamente en la implementación de las reglas semánticas dentro de la gramática. Para dicha implementación se han distinguido tres estructuras sintácticas distintas, clasificadas según el

verbo de la acción involucrado, tal y como se presentan en las tablas 8.1. De este modo las reglas semánticas de la gramática, en la parte referida a la mención de una acción quedaría del siguiente modo:

```
$eye = ( ( "ojo" | "parpado" )[$left | $right] ){<@body "Eye">};
$arm = ( ("brazo" | "hombro") [$left | $right] ){<@body "Arm">};
$neck = ( "cabeza" | "cuello" ){<@body "Neck">};
$hand = ( ( "mano" ) [$left | $right] ){<@body "Hand">};
$back = ( "espalda" ){<@body "Back">};

$body4rise = $eye | $arm | $neck;
$body4spin = $neck;
$body4touch = $arm | $neck | $hand | $back;

$left = ( "izquierdo" |"izquierda" ){<@axis "left">};
$right = ( "derecho" |"derecha" ){<@axis "right">};
$straight = ("recto" |"centro" | "centrado"){<@axis "Center">};
$axis = $left | $right | $straight;

$action1 = ( ( $rise | $down ) [$aDet] [$body4rise] )
           {<@sintaxis "1">};
$action2 = ( $spin [$aDet] [$body4spin] [a_la] [$axis] )
           {<@sintaxis "2">};
$action3 = ( $fwd | $rev ){<@sintaxis "3">};
$action = ( $action1 | $action2 | $action3 ){<@type "action">};
```

En estas reglas pueden observarse las definiciones de los siguientes atributos semánticos: `body`, `axis`, `sintaxis` y `type`. El primero completa la parte del cuerpo del robot involucrada en la acción, que siempre va a ser de movimiento. El segundo involucra el carácter direccional de, o bien, la parte del cuerpo que realiza el movimiento, o bien hacia dónde se realiza dicho movimiento. El tercero, identifica el tipo de sintaxis considerado, dentro de los tres expuestos en las tablas 8.1. Por último, el atributo `type`, especifica el tipo de directiva de acción[3] enunciada. En este caso, se trata de la adición de una etapa o acción.

[3]acción de diálogo, se entiende (véase 7.1.2)

8.3.2. Adición de una condición

Las reglas de la gramática semántica que implementa la percepción de un enunciado que quiere añadir una transición o condición es del siguiente modo:

```
$arm = ( ("brazo" | "hombro") [$left | $right] ){<@body "Arm">};
$neck = ( "cabeza" | "cuello" ){<@body "Neck">};
$hand = ( ( "mano" ) [$left | $right] ){<@body "Hand">};
$back = ( "espalda" ){<@body "Back">};

$body4touch = $arm | $neck | $hand | $back;

$touched = ([magui] (cuando | si | esperas [a | que])
           [te] $touch [$body4touch]);

$condition = ( $touched ){<@type "condition">};
```

Nótese que dicha regla se refiere a las condiciones del dominio inicial expuesto en la sección 3.5, análogamente al dominio inicial de acciones.

8.3.3. Operaciones estructurales

En este tipo de operaciones se incluyen la apertura y clausura de varias ramas en paralelo, bien que se vayan a ejecutar concurrentemente, bien que se trate de una selección de secuencias. Asimismo, se incluye la creación de un bucle en la secuencia.

```
$selseqinit = ((( abres | consideras | creas ) $cardinal opciones
              | (puedo hacer $cardinal cosas)){<@type "selseq">};
$atonce = (a_la_vez [haces] [$cardinal] [cosas]){<@type "atonce">};

$conclution=((despues_de [hacer [alguna de]] las [$cardinal] [ramas])
  | ([luego|y] (cierras | unes | terminas) las [$cardinal] opciones))
  {<@type "conclution">};
$loop = [despues | (por ultimo)]
  ((vuelves al principio)|(repites la secuencia)){<@type "loop">};

$ad = ($selseqinit|$atonce|$conclution|$loop){<@da "ad">};
```

En todas estas reglas, el valor del atributo semántico `da` (acto de diálogo) es una directiva de acción (`ad`) En el atributo `type` se especifica de qué tipo de directiva se trata.

8.4. Descripción global del sistema interactivo de edición

En esta sección se explica la arquitectura del sistema global de edición de secuencias mediante interacción. Este sistema se puede dividir en tres: el gestor de diálogo, el secuenciador y el editor de la secuencia. El primero se ha explicado en la sección 7.2; la generación y ejecución de la secuencia también se ha explicado en 6.3.2. Así, que en esta sección se explicará como funciona el editor en su relación con el sistema de gestión del diálogo y el secuenciador.

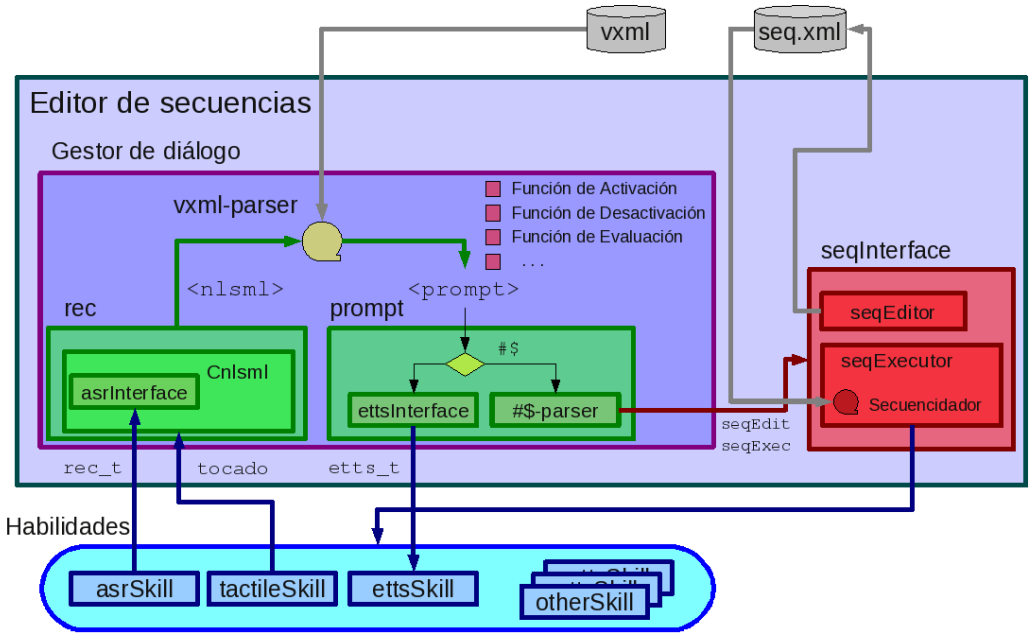


Fig. 8.1: *Esquema general del sistema de edición de secuencias*

En la figura 8.1 se representa un esquema de la arquitectura del sistema de edición de secuencias mediante interacción. El sistema de diálogo se encarga de recopilar toda la información necesaria para la edición de una secuencia. Esta información es de dos tipos: a nivel funcional, qué acciones o condiciones

se están añadiendo, y a nivel estructural, la estructura del diagrama funcional creado.

La habilidad de reconocimiento automático del habla devuelve información que va a ser utilizada para tres propósitos principales: *movimientos en el flujo del diálogo* (mediante los actos de diálogo percibidos), *obtención de información funcional de la secuencia* (mediante los atributos asociados a una acción o a una condición), y *obtención estructural y de ejecución de la secuencia* (mediante los atributos asociados a la creación de las estructuras básicas de una secuencia: secuencia simple, selección de secuencias y secuencias simultáneas).

Los movimientos en el flujo del diálogo permiten que el proceso de interacción que robot y usuario llevan a cabo cambie de un escenario a otro. Así por ejemplo, después de que el robot realice un saludo esperará que el usuario conteste con otro saludo, después de lo cual el robot se moverá a otro escenario, por ejemplo, a una breve presentación del sistema, etc. Estos movimientos en el diálogo dependen de la implementación específica en ficheros `vxml`, que se esté interpretando. Dicha implementación puede ser alterada mientras el propio diálogo se está ejecutando, de modo que pueden crearse ficheros `vxml` y ser cargados en tiempo de ejecución.

Por otro lado, a partir de los resultados semánticos obtenidos del reconocedor automático del habla (en la figura, esta información está en el tipo de dato `rec_t`) el sistema de diálogo construye explícitamente las funciones en lenguaje `Python` que son incorporadas en la secuencia en formato `XML`. Estas funciones corresponden a `Función de Activación` y `Función de Desactivación` para las etapas, y `Función de Evaluación` para las transiciones. El reconocedor automático del habla, en este caso, sirve para completar información funcional de la secuencia.

La información estructural, permite especificar aspectos relacionados con la disposición de las distintas ramas de la secuencia: si se van a abrir varias ramas que se ejecutaran en paralelo, si se va a realizar un bucle, etc. El usuario también puede ordenar la ejecución, pausa o borrado de la secuencia que se esté creando.

8.4.1. Recopilación funcional y estructural de la secuencia

El modo en que un acto de diálogo entra en el sistema de diálogo es gracias a incorporar el campo `da`, dentro de un formulario. Este campo es completado directamente con el valor que se le ha asignado al atributo del mismo nombre,

da, a través del resultado semántico del reconocedor automático del habla. El valor de dicho campo es analizado tal y como se muestra a continuación:

```
...
<form id = "node">
    <initial name = "inicio">
    <prompt>\item=Throat_01</prompt>
    <prompt>Dime una accion o una condicion</prompt>
  </initial>

  <field name = "da">
    ...
  </field>
  <field name = "type" cond = "da == 'ad'">
    <prompt>
 Dime la accion o la condicion que quieres incluir
    </prompt>
    <filled>
      <if cond = "type == 'action'">
  <prompt>marchando</prompt>
      <elseif cond = "type == 'condition'"/>
  <prompt>a adiendo una condici n</prompt>
  <assign name = "sintaxis" expr = "true"/>
  <assign name = "limit" expr = "true"/>
      ...
    </filled>
  </field>
...
```

El campo "type" diferencia el tipo de directiva de acción que el usuario a enunciado. En este caso, lo que interesa es saber si este tipo corresponde a una adición de una acción o de una condición. Nótese que en la declaración del campo "type" se añade la opción "cond" dentro del elemento <field> Esta opción establece una condición que ha de cumplirse para que el algoritmo FIA entre en el campo. En este caso, entraría si y solo si el valor del campo previo, da, es una directiva de acción (da == 'ad').

Una vez completado el campo "type" se ejecuta la correspondiente sección delimitada por el elemento <filled> En ella se analiza si el tipo es una acción (type == "action"), o una condición (type == "condition") En el primer caso el robot contesta con el mensaje al usuario: *"marchando"* para que éste entienda que el robot ha captado la directiva de acción de añadir una acción a la secuencia. En caso de que la directiva de acción sea el añadir una condición el robot confirma con el enunciado *"añadiendo una condición"* Además se realizan dos asignaciones a los campos posteriores sintaxis y limit, dado que son campos no necesarios para añadir una condición. De

este modo, evita que el algoritmo FIA entre en dichos campos.

Los parámetros que denotan cada acción o cada condición han de ser recopilados por el sistema de diálogo. Para ello, se crean más campos dentro del formulario:

```
<field name = "sintaxis">
  <prompt>No acabo de entender la estructura de la frase</
     prompt>
  <filled>
    <log>sintaxis = <value expr = "sintaxis"/></log>
    <if cond = "sintaxis == 2">
      <assign name = "limit" expr = "true"/>
    <elseif cond = "sintaxis == 3"/>
      <assign name = "axis" expr = "true"/>
    </if>
  </filled>
</field>

<field name="verb">
    <prompt>Me falta el verbo.</prompt>
    <prompt cond = "type== 'action'">Dime el nombre de la
       acci n</prompt>
    <prompt cond = "type == 'condition'">Dime que me vas a
       hacer</prompt>
  <filled>
    <log>verb = <value expr = "verb"/></log>
  </filled>
</field>

<field name="body">
  <prompt>Dime la parte del cuerpo involucrada</prompt>
    <filled>
<if cond = "(body == 'Neck')&amp;amp;(sintaxis == 1)">
  <assign name = "axis" expr="true"/>
</if>
    </filled>
</field>

<field name="axis">
  <prompt>Izquierda o derecha?
  </prompt>
  <filled>
    <log>axis = <value expr = "axis"/></log>
  </filled>
</field>

<field name="limit">
```

```
      <prompt>Hasta donde?</prompt>
40    <filled>
        <log>limit = <value expr = "limit"/></log>
42    </filled>
    </field>
```

En este caso los campos `sintaxis`, `body`, `axis` y `limit` corresponden a los atributos con la misma denominación explicados anteriormente en la descripción de las reglas semánticas de la gramática.

Obsérvese que según sea el valor de la variable `sintaxis`, se autocompletan o no otras variables, para que el algoritmo FIA no intente recopilar información que no es necesaria.

Tal y como se realiza esta implementación de ejemplo, mediante un solo enunciado el usuario podría completar todos los campos necesarios. Así por ejemplo, si el usuario realiza el enunciado *gira a la derecha*, tal y como se han construido las reglas semánticas de la gramática, se obtendrían los siguientes atributos:

```
@da = 'ad'
@type = 'action'
@sintaxis = '2'
@verb = 'spin'
@body = 'Base'
@axis = 'right'
```

Estos atributos entran en los distintos campos con igual nombre, dentro del sistema de diálogo implementado en el fichero `vxml`. En el ejemplo, el valor del atributo `sintaxis = '2'`, provoca la asignación `limit = 'true'`, con lo que el algoritmo FIA no entra en ese campo.

Una de las ventajas de recopilar esta información mediante `voiceXML` es que, si, por el contrario, el usuario no menciona todos los atributos necesarios marcados por el formulario, el robot toma la iniciativa y pregunta al usuario por los atributos que falten.

Así por ejemplo, si el usuario realiza el enunciado *gira*, los atributos semánticos devueltos por el reconocedor automático del habla serían:

```
@da = 'ad'
@type = 'action'
@sintaxis = '2'
@verb = 'spin'
@body = 'Base'
```

con lo que en el formulario faltaría por rellenar el campo `axis`. El algoritmo FIA entra pues en dicho campo y ejecuta su contenido. En este caso realiza una consulta al usuario a través del elemento `<prompt>`: *¿izquierda o derecha?*, y se repite mientras el usuario no mencione información que complete el valor semántico `axis` tal y como define la correspondiente regla semántica de la gramática.

8.4.2. Construcción de las funciones de la secuencia

Las funciones de la secuencia se completan a a través de los valores asignados a los distintos campos dentro de un formulario. Más tarde, dichas funciones se envían al editor de la secuencia a través del protocolo-#$.

En la sección 3.5 se explicó cómo se ha implementado el acceso a los sensores y actuadores de Maggie desde el lenguaje interpretable `Python`. Este acceso, permite la comunicación entre el hardware del robot y la secuencia en formato `XML`. En aquella sección se explicaron, para un dominio inicial de acciones, los distintos formatos de las llamadas a los métodos que acceden al hardware del robot. Estos métodos se utilizan en la asignación de las funciones que utiliza la secuencia, que son `FuncionActivacion` y `FuncionDesactivacion` para las etapas, y `FuncionEvaluacion` para las transiciones. En esta sección se explica cómo se crean estas funciones a partir de los resultados semánticos de las gramáticas explicadas anteriormente.

Funciones para una acción o etapa

Después de haber obtenido toda la información necesaria para denotar una acción, se construyen las funciones "FuncionActivacion" y "FuncionDesactivacion". Además, tal y como se explicó en la sección 3.5, cada vez que se añade una acción a la secuencia, el sistema también define una `FuncionTerminacion` que permite a la secuencia consultar si la acción ha terminado de ejecutarse o no. Esta función es utilizada como `FuncionEvaluacion` para las condiciones implícitas posteriores.

Analizamos con detalle el código involucrado en esta construcción. Primero se declaran variables que se van a utilizar más tarde:

```
<var name = "clase"/>
<var name = "method"/>
<var name = "position"/>
<var name = "FuncionActivacion"/>
<var name = "FuncionTerminacion"/>
```

Una vez completados todos los campos del formulario, se realiza un análisis de cada uno de estos campos dado que están relacionados con los parámetros que definen cada una de las funciones `Python`. Los valores asignados a estos campos, han sido obtenidos mediante diálogo a partir de la información semántica que devuelve la habilidad de reconocimiento automático del habla, tal y como se ha explicado anteriormente.

En dicho análisis cabe destacar la distinción entre los tres tipos de sintáxis explicados en las tablas 8.1:

```
<block>
 <if cond = "type == 'action'">
   <if cond = "sintaxis == 1">
      <if cond = "body == 'Arm'">
        <assign name = "clase" expr = "axis + body"/>
      <else/>
        <assign name = "clase" expr = "body"/>
</if>
      <assign name = "method" expr = "verb"/>
      <assign name = "position" expr = "0.7 + '*' + body +
          limit"/>
    <elseif cond = "sintaxis == 2"/>
      <assign name = "clase" expr = "body"/>
      <assign name = "method" expr = "verb"/>
      <assign name = "position" expr = "body + axis"/>
    <elseif cond = "sintaxis == 3"/>
      <assign name = "clase" expr = "'Base'"/>
      <assign name = "method" expr = "verb"/>
      <assign name = "position" expr = "'Base' + limit"/>
   </if>
```

De este modo, si por ejemplo el usuario ha dicho *"Levanta la cabeza"*, los campos completados serían `sintaxis = '1'`, `body = 'Neck'`, `verb = 'move'` y `limit = 'Top'`, a partir de los cuales, se completan las variables del siguiente modo `clase = Neck`, `method = move`, `position = 0.7*NeckTop`.

La definición de las funciones `Python` se realiza mediante el siguiente código:

```
<assign name = "FuncionActivacion" expr = "clase + '.' +
    method + '(' + position + ')\n'"/>
<assign name = "FuncionActivacion" expr = "
    FuncionActivacion + 'returnAD = 1\n'"/>
<assign name = "FuncionTerminacion" expr = "'p = '+ clase +
     '.' + 'get' + verb + 'Pos()\n'"/>
<assign name = "FuncionTerminacion" expr = "
    FuncionTerminacion + 'if( abs( p -' + position + ')/('
    + position + ')&gt;= 0.10 ) :\n'"/>
```

```
4       <assign name = "FuncionTerminacion" expr = "
            FuncionTerminacion + '\\ returnAD = 0\n'"/>
        <assign name = "FuncionTerminacion" expr = "
            FuncionTerminacion + 'else :\n'"/>
6       <assign name = "FuncionTerminacion" expr = "
            FuncionTerminacion + '\\ returnAD = 1\n'"/>
  ...
8   </block>
```

Siguiendo con el ejemplo anterior, en el que el usuario quiere añadir la acción "levantar cabeza", las funciones `Python` quedarían del siguiente modo:

```
FuncionActivacion :=
 'Neck.move( 0.7*NeckTop )'

FunctionTerminacion :=
 'p = Neck.getMovePos()
  if( abs( p - 0.7*NeckTop ) / (0.7*NeckTop) >= 0.10 ) :
     returnAD = 0
  else :
     returnAD = 1'
```

La primera función se ejecutaría cuando la etapa asociada a la acción "levantar cabeza" pasase de estado no-activo a estado activo. La segunda función sirve para comprobar que la acción ha finalizado. Básicamente, en el ejemplo, esta función comprueba que la posición de la cabeza respecto a la "posición objetivo" (0.7*NeckTop) tiene un error relativo menor al 10 %. Esta función de terminación es utilizada cuando se añada una transición implícita, del tipo "nodo then", lo cual es explicado más adelante.

Por último, solo falta comunicar desde el código `voiceXML` a la clase encargada de editar la secuencia la adición de la acción. Para ello se utiliza el siguiente código:

```
0       <prompt>#funcionActivacion$<value expr = "
            FuncionActivacion"/></prompt>
        <prompt>#funcionTerminacion$<value expr = "
            FuncionTerminacion"/></prompt>
2       <if cond = "lastresult$.confidence &lt; 0.5">
          <goto next = "#confirmAction"/>
4       <else/>
          <prompt>#action</prompt>
6         <prompt>ya esta</prompt>
        </if>
```

En este código, el elemento `<prompt>` incluye comandos que son tratados por el intérprete del protocolo `#$` (`#$-parser`) En las dos primeras líneas se envía a la clase que edita la secuencia los valores de las funciones de activación y terminación, respectivamente. Luego, antes de enviar el comando `action` a la clase que edita la secuencia, se comprueba que el margen de confianza del último reconocimiento pasa un umbral del 50 %. Este valor se ha escogido empíricamente de acuerdo al funcionamiento del sistema, hardware y software, implementado en la habilidad `asrSkill`. Si el valor es inferior el flujo del diálogo se mueve al subdiálogo `confirmAction`. Una vez añadida la acción a la secuencia o diagrama funcional, el robot responde con una confirmación: "ya está."

Así por ejemplo, un diálogo siguiendo el código explicado podría ser el siguiente:

MAGGIE: Dime la acción o la condición que quieres añadir
USUARIO: Levanta ...
[*Se completan los atributos* `@da = 'ad'`, `@type = 'action'`, `@verb = 'move'`, `@limit = 'Top'`, `@sintaxis = '1'`]
MAGGIE: Dices, levanta.
MAGGIE: Dime la parte del cuerpo involucrada.
USUARIO: el brazo ...
MAGGIE: Dices, el brazo.
[*Se completan los atributos* `@da = 'nd'`, `@body = 'Arm'`]
MAGGIE: ¿izquierdo o derecho?
USUARIO: izquierdo...
MAGGIE: Dices, izquierdo.
[*Se completan los atributos* `@da = 'nd'`, `@axis = 'left'`, *y, por tanto, todos los atributos necesarios para completar las funciones asociadas a la etapa.*]
MAGGIE: ya está

Nótese que un enunciado tan simple como *"levanta"* completa hasta cinco atributos semánticos, lo cual muestra la gran potencia que tienen las reglas semánticas de la gramática. Aún así, al no completarse los valores del resto de campos o atributos necesarios, el robot pregunta por los que faltan.

Funciones para una condición o transición

Después de haber obtenido toda la información necesaria que denota una condición, se construye la función `Python` asociada, esto es, la función `FuncionEvaluacion` explicada en la sección 3.5.

Analizamos con detalle el código involucrado en esta construcción. Para una condición explícita se tiene el siguiente código:

```
<block>
 ...
 <elseif cond = "type == 'condition'"/>
   <assign name = "FuncionTransicion" expr = "'c = conditions
       .' + verb + '(&quot;' + body + '&quot;,&quot;' + axis
       + '&quot;)\n'"/>
   <assign name = "FuncionTransicion" expr = "
       FuncionTransicion + 'returnAD = c\n'"/>
   <log>funcTransicion= <value expr = "'\n' +
       FuncionTransicion"/></log>
   <prompt>#funcionTransicion$<value expr = "
       FuncionTransicion"/></prompt>
   <prompt>#condition</prompt>
   <assign name = "nConditions" expr = "nConditions + 1"/>
   <prompt>hecho!</prompt>
 ...
```

Una vez detectado que la directiva de acción es de tipo "`condition`" se define la función de transición como una llamada a una función dentro del módulo `Python conditions.py`, que ha sido explicado en la sección 3.5.

Así por ejemplo, si el usuario quiere añadir la condición "cuando te toque el hombro derecho", los valores semánticos involucrados quedarían definidos del siguiente modo:

```
@verb = 'touch'
@body = 'Arm'
@axis = 'right'
```

Así, la función de evaluación asociada a la transición sería:

```
FuncionTransicion =
'c = conditions.touch( "Arm", "right" )
 returnAD = c'
```

En las siguientes líneas se envía a la clase encargada de editar la secuencia que añada una condición mediante el comando "`condition`", pasando previamente el valor de la función de evaluación.

Por último, el robot confirma que ha realizado la adición de la transición correctamente mediante el enunciado "`hecho!`".

9. RESULTADOS EXPERIMENTALES. EJEMPLOS DE CONSTRUCCIÓN DE SECUENCIAS MEDIANTE DIÁLOGO.

En este capítulo se presentan algunos ejemplos de diálogos diseñados para la edición verbal de secuencias. Dado que el sistema de diálogo se basa en la interpretación en tiempo real de ficheros `voiceXML`, resulta muy versátil, ya que, es posible implementar diversos diálogos concretos, sin realizar ningún cambio en el gestor del diálogo.

Con el objetivo de mostrar de alguna manera el potencial de este sistema, en lo referente a la edición verbal de secuencias, se presentan dos ejemplos contrapuestos. En el primero, se muestran distintos diálogos que no están necesariamente relacionados de forma directa con la edición de la secuencia, pero que constituyen una parte fundamental de toda interacción natural. Se trata de diálogos protocolarios que ayudan a establecer una relación concreta entre el robot y el usuario, y a conocerse mutuamente. La relación entre interlocutores es de especial importancia en la interacción, entre otros motivos, porque clasifica el aspecto de contenido de su comunicación (sobre la interacción presencial humana véase apéndice A.1.4).

En el segundo ejemplo se muestra una aplicación de diálogos en donde se espera que la iniciativa la tome mayormente el usuario. Se supone que el usuario ya ha utilizado este sistema en alguna ocasión y conoce el léxico, las estructuras gramaticales y las consecuencias de su locución en la edición de la secuencia. El sistema, por su parte, se encarga de ir recopilando toda la información necesaria para ir construyendo la secuencia, realizando las preguntas oportunas en relación a la información que falte, y ayudando así al usuario según el contexto del diálogo en el que se encuentren.

En esta sección se va explicando las partes más importantes del código

voiceXML en el que se implementa el diálogo.

9.1. Ejemplos de diálogos en escenarios protocolarios

A continuación, se presentan algunos ejemplos de diálogos sencillos en los que se podrá observar el uso que se hace de las definiciones de las reglas semánticas de la gramática explicada, así como el funcionamiento de voiceXML.

Los sistemas de diálogo que aquí se presentan son ejemplos de lo que se han venido a denominar escenarios protocolarios: saludos, despedidas, petición de ayuda, y petición de confirmación.

9.1.1. Saludos y comienzo del diálogo

En este ejemplo, primero se declara una variable a la que se le asigna un valor aleatorio entre 0 y 1, mediante el correspondiente código en ECMAScript. Esta variable se utiliza para dar mayor naturalidad al diálogo, de modo que no siempre se repitan las mismas frases:

```
<var name = "r" expr = "Math.random()"/>
```

A continuación comienza el formulario contact, que tiene dos campos: uno implementado en el elemento <initial name = ‘‘saludo’’> y otro mediante el elemento <field name = ‘‘da’’>. Al haber un campo <initial> el diálogo es de iniciativa mixta, con lo que el usuario puede ser el primero que realice el saludo.

```
<form id = "contact">
  <initial name="saludo">
     <prompt cond = "r &gt; .49">Hay alguien por aqui?</prompt>
     <prompt cond = "r &lt; .50">Hola, soy magui</prompt>
  </initial>
 <field name="da">
   <prompt>
      Hola, es lo primero, no?
   </prompt>

   <filled>
     <if cond= "da == 'fp'>
       <prompt cond = "r &gt; .50">\item=Ollah</prompt>
       <prompt>Que tal!</prompt>
       <goto next = "12-contact.vxml"/>
     </if>
     <clear namelist= "da"/>
```

```
    </filled>
  </field>
</form>
```

El campo "`da`" se rellena con un enunciado del usuario, mediante la regla gramatical descrita que define un atributo semántico con el mismo nombre. Una vez completado el campo, se ejecuta el contenido del elemento `<filled>`. En este elemento se comprueba si el usuario ha realizado un saludo, es decir, si el acto de diálogo es "`fp`" (apertura convencional). En caso afirmativo, se devuelve el saludo y se pasa a otro diálogo, en este caso otro fichero externo `<goto next ="12-contact.vxml">`.

Si el usuario no realiza un saludo, entonces se ejecuta el elemento `<clear>` que borra el contenido del campo `da`. De este modo el algoritmo FIA vuelve a entrar en dicho campo y pasa a ejecutar su contenido.

Un ejemplo de un diálogo dentro de esta escena sería el siguiente:

MAGGIE: ¿Hay alguien por aquí?
USUARIO: Quiero jugar contigo
MAGGIE: Hola, es lo primero, ¿no?
USUARIO: ¡Hola Magui!
MAGGIE: ¡Hola!
[*El flujo de diálogo se mueve al primer formulario del fichero 12-contact.vxml*]
...

9.1.2. Despedida y finalización del diálogo

Para un escenario de este tipo, los actos de diálogo involucrados serían `fc` y `e`, es decir, clausura convencional y salida. Un fragmento de código `voiceXML` que permitiría la interpretación de ambos actos de diálogo sería:

```
<field name = "da">
  <filled>
  ...
  <if cond = "da == 'e'">
    <prompt>Saliendo. besos!</prompt>
    <exit/>
  <elseif cond = "da == 'fc'"/>
    <prompt>Adios, un besote!</prompt>
    <exit/>
  ...
  </filled>
  ...
</field>
```

Una vez percibidos, el sistema directamente sintetiza una frase de despedida, mediante el elemento `<prompt>` y sale, mediante el elemento `<exit>`.

Un ejemplo de diálogo podría ser así:

MAGGIE: Dime la acción o condición que quieres añadir ahora.
USUARIO: quiero salir
MAGGIE: Saliendo, ¡besos!
[*El flujo de diálogo termina*]
...

9.1.3. Petición de ayuda

Para un escenario de petición de ayuda una vez detectado el acto de diálogo "`h`" el robot ofrece su ayuda y mueve el flujo del diálogo a otro fichero externo "`ayuda.vxml`":

```
...
  <field name = "da">
...
    <filled>
...
      <elseif cond = "da == 'h'"/>
  <prompt>Perdona!</prompt>
  <prompt>Voy a echarte una mano</prompt>
  <goto next = "ayuda.vxml">
...
    </filled>
...
```

En este fichero se ofrece ayuda en distintos niveles explicando al usuario las opciones que tiene, etc. El ejemplo más sencillo es ofrecer información verbal general:

```
<prompt>Tienes que decirme que accion o condicion quieres
    incorporar en la secuencia.</prompt>
<prompt>Para decirme una accion tienes que nombrar un verbo y
    la parte del cuerpo involucrado. Por ejemplo, levantar
    brazo izquierdo, girar a la derecha.</prompt>
<prompt>Por ejemplo. Si quieres que levante el brazo izquierdo,
     di levanta el brazo izquierdo.</prompt>
<prompt>Tambien puedes decir que haga varias cosas a la vez.</
    prompt>
<prompt>Por ejemplo me dices a la vez haz tres cosas.</prompt>
<prompt>O tambien puedes decir que quieres crear una seleccion
    entre varias condiciones.</prompt>
```

```
6  <prompt>Por ejemplo me dices que considere tres opciones.</
       prompt>
```

O bien hacer la ayuda más interactiva con preguntas de confirmación, etc.

Un ejemplo de diálogo para esta escena podría ser así:

MAGGIE: Dime la acción o condición que quieres añadir ahora.
USUARIO: Échame una mano, porfi.
MAGGIE: ¡Perdona!
MAGGIE: Voy a echarte una mano.
[*El flujo de diálogo se mueve al fichero ayuda.vxml*]
MAGGIE: Para decirme una acción tienes que nombrar un verbo y la parte del cuerpo involucrado. Por ejemplo, levantar brazo izquierdo, girar a la derecha, ...
...

9.1.4. Confirmación de una acción mencionada por el usuario

Esta confirmación básicamente consiste en realizar al usuario una consulta sobre la acción que se va a añadir, dando así la opción al usuario de realizar las correcciones pertinentes, si las hubiere:

```
0  <form id= "confirmAction">
     <field name = "da">
2      <prompt>
         Estas seguro de que quieres incluir esta accion?
4      </prompt>
       <filled>
6        <if cond = "da == 'nn'">
           <prompt>
8            Ah! Vale! Me he colao! Volvemos a empezar!
           </prompt>
10       <elseif cond = "da == 'ky'"/>
          <prompt>Perfecto!</prompt>
12        <prompt>#action</prompt>
          <prompt>ya esta</prompt>
14       <else/>
           <clear namelist = "da"/>
16         <prompt>
             Dices <value expr = "lastresult$.utterance"/>
18         </prompt>
           <prompt>Contesta sii, o no.</prompt>
20         <goto next = "#confirmAction"/>
```

```
        </if>
22      <goto next = "#node"/>
      </filled>
24  </field>
</form>
```

En este diálogo o formulario, el robot realiza el acto de diálogo `qy` (pregunta de sí o no)[1] Luego analiza la respuesta del usuario. Ésta ha de ser un acto de diálogo del tipo `ky` (respuesta afirmativa) o `nn` (respuesta negativa) que son los *pares adyacentes* a `qy` (véase sección 7.1.2)

Si el usuario realiza otro acto de diálogo, el sistema repite lo que ha entendido mediante la variable `lastresult$.utterance`, que se ha explicado en el apéndice D. Después reconduce el diálogo dando la instrucción informativa *"Contesta sii, o no"*, y vuelve a realizar la petición de confirmación. Nótese que el campo `<field name = ‘‘da’’>` ha sido borrado mediante el elemento `<clear namelist = ‘‘da’’>`

La confirmación del usuario puede ser positiva, en cuyo caso se añade la acción, o negativa, en cuyo caso el robot enuncia *Ah! Vale! Me he colao! Volvemos a empezar!* y el flujo del diálogo vuelve al diálogo de partida, en el ejemplo denotado por la etiqueta `node`, mediante el elemento `<goto next = ‘‘#node’’>`

9.2. Ejemplo de edición de una secuencia con ayuda contextual

En esta aplicación, el usuario y el sistema dialogan para editar una secuencia. El diseño se ha realizado considerando que la iniciativa reside principalmente en un usuario que ya ha utilizado la herramienta y que, por tanto, en cierta medida la conoce. No obstante, el sistema es capaz de ofrecer al usuario la ayuda necesaria mediante enunciados que guíen el proceso de edición solicitando la información precisa necesaria.

Así, de cada enunciado del usuario, el sistema trata de recopilar una serie de atributos. Éstos han sido especificados en la gramática del reconocedor automático del habla (véase apartado 4), y son introducidos en el sistema gestor de diálogo mediante la interfaz que construye el resultado en lenguaje natural (véase 7.3.2). Esta interfaz, introduce también datos no verbales como, por ejemplo, los asociados a los eventos de los sensores de tacto.

[1]Nótese, que los actos de diálogo pueden ser ejecutados tanto por el usuario como por el robot, aunque en el presente trabajo se haya desarrollado con mayor extensión la percepción de actos del diálogo del usuario por parte del robot.

El resto de la sección continúa del siguiente modo. Primero se describen los atributos comentados. Después, se describe el proceso de percepción de estos atributos. Se continúa describiendo cómo afecta la percepción de estos atributos a la secuencia y cómo ésta es finalmente ejecutada.

9.2.1. Descripción de los atributos comentados

Los atributos verbales son los siguientes :

- `da`, acto de diálogo asociado al enunciado. En el ejemplo se consideran los siguientes: `ad` o directiva de acción, `e` o salida, `nd` o respuesta general, `fc` o clausura convencional, `h` o duda y petición de ayuda, `ky` respuestas afirmativas, `nn` respuestas negativas y `sd`, declaración sin emisión de juicio.
- `type`, cuando el acto de diálogo es una directiva, este atributo se refiere al tipo de directiva. Ésta puede tomar los siguientes valores: `action`, `condition`, `then`, `atonce`, `selseq`, `loop` y `runseq`.
- `sintaxis`, tipo de sintaxis de cuando el tipo de la directiva es `action` o `condition`.
- `verb`, identifica el verbo de la acción o el asociado a la condición mencionada.
- `body`, identifica la parte del cuerpo involucrada en la acción o condición.
- `axis`, identifica la lateralidad de la acción o de la parte del cuerpo mencionada.
- `limit`, está relacionado con el valor de posición de movimiento asociado a la acción o condición.
- `cardinal`, establece el número de ramas en paralelo que van a ser abiertas.
- `ordinal`, sirve para identificar una rama dentro de una estructura de varias ramas en paralelo.

El único atributo no verbal considerado es `tocado`, que está asociado a la habilidad de tacto.

La aplicación se construye en un único fichero con dos formularios o diálogos[2]: `node` y `confirmAction`. Una vez recopilada toda la información, se edita la parte de la secuencia tal y como se haya establecido mediante diálogo y el flujo de la aplicación vuelve al comienzo.

Antes del primer formulario se coloca la cabecera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" xmlns:xsi="http://www.
    w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
    www.w3.org/2001/vxml http://www.w3.org/TR/voicexml20/vxml.xsd
    " version="2.0">
  <var name = "clase"/>
  <var name = "method"/>
  <var name = "position"/>
  <var name = "nConditions" expr = "0" />
  <var name = "n" expr = "0" />
  <var name = "FuncionActivacion"/>
  <var name = "FuncionTerminacion"/>
  <var name = "FuncionTransicion"/>
  <var name = "confianza" expr = "1"/>
```

En ella puede observarse que el actual fichero realiza una apertura a nivel documento, pero también a nivel aplicación, tal y como se explica en D.2. Por tanto, las variables declaradas en este ámbito son visibles en todos los documentos y diálogos de la aplicación. La aplicación lleva por nombre el de este fichero principal.

Comienza el primer formulario, con una sección que establece la gramática mediante el envío de un comando en el protocolo #$, el comando `setGrammar=sfcPy.gram` es enviado a la habilidad de reconocimiento automático del habla para que cargue el fichero con la gramática semántica.

```
  <form id = "node">
    <block>
      <prompt count="1">#setGrammar$sfcPy.gram</prompt>
      <assign name = "n" expr = "n + 1"/>
    </block>
```

La variable `n` funciona como un contador, dado que la aplicación se ejecuta en bucle, hasta que el usuario solicite salir. La exposición que prosigue se refiere al análisis de un solo enunciado (turno) del usuario.

Como se quiere una iniciativa mixta, y que el usuario mediante un solo enunciado pueda rellenar todos los campos o atributos del sistema, se

[2]Diálogo concebido según la nomenclatura que define el estándar `voiceXML` (véase apéndice D)

comienza con un elemento `<initial>`, tal y como se explica en la sección D.5.2:

```
<initial name = "inicio">
  <prompt cond = "n == 1">\item=Throat_01</prompt>
  <prompt cond = "n == 1">Dime una accion o una condicion</
      prompt>
  <prompt cond = "n == 2">Despues que?</prompt>
  <prompt cond = "n &gt; 2">y ahora?</prompt>
</initial>
```

Mediante distintos valores de la variable n puede cambiarse el mensaje de inicio que da el robot, para seguir editando la secuencia.

9.2.2. Percepción del acto de diálogo

Puesto que la lectura de los distintos campos es jerárquica y ordenada, se comienza por el campo más importante, que es el referido al acto de diálogo asociado al enunciado del usuario:

```
<field name = "da">
  <filled>
    <prompt cond = "(da == 'nd') | (da == 'ad')">dices <value
        expr = "lastresult$.utterance"/>. Confianza = <value
        expr = "lastresult$.confidence"/></prompt>
<if cond = "da == 'e'">
  <prompt>Saliendo. besos!</prompt>
  <exit/>
    <elseif cond = "da == 'nd'"/>
  <prompt>\item=Mhm</prompt>
  <assign name = "type" expr = "true"/-->
<elseif cond = "da == 'fc'"/>
  <prompt>Adios, un besote!</prompt>
  <exit/>
    <elseif cond = "da == 'h'"/>
  <prompt>Perdona!</prompt>
  <prompt>Voy a echarte una mano</prompt>
  <goto next = "ayuda.vxml">
  <assign name = "inicio" expr = "true"/>
<elseif cond = "da == 'ky'"/>
  <prompt>|item=Laugh</prompt>
<elseif cond = "da == 'sd'"/>
  <assign name = "sintaxis" expr = "true"/>
  <assign name = "verb" expr = "true"/>
  <assign name = "body" expr = "true"/>
  <assign name = "axis" expr = "true"/>
```

```
24    <assign name = "limit" expr = "true"/>
  <elseif cond = "da == 'nn'"/>
26          <prompt>|item=Smack Perdon! Me he colado. Rectifico. |
                item=Laugh</prompt>
  </if>
28      </filled>
    </field>
```

Una vez completado el campo, se ejecuta el contenido de la etiqueta `<filled>` Se comienza repitiendo lo que el sistema ha reconocido literalmente, lo cual sirve de realimentación para que el usuario pueda realizar la depuración que considere. Luego se analizan los distintos valores del acto de diálogo del enunciado. En caso de que se trate de un acto de diálogo "afirmación" (de tipo `sd`) se entiende que el usuario está realizando una operación de referencia de alguna de las ramas, tal y como muestra la regla semántica de la gramática:

```
 /* statement-non-opinion */
$sd = ([en (el|la)]($cardinal| $ordinal)secuencia){<@da "sd">};
```

Por tanto, es de suponer que se está editando una estructura secuencial con varias ramas en paralelo, y que el usuario quiere colocar el foco en alguna de ellas. Así que hay diversos campos que no tienen sentido y, por tanto, se autocompletan mediante la etiqueta `<assign>`. En este caso se trata de los campos `verb`, `body`, `axis` y `limit`.

El siguiente campo, analiza el tipo de directiva de acción que ha sido emitida.

```
0     <field name = "type" cond = "da == 'ad'">
        <filled>
2         <if cond = "type == 'action'">
      <prompt>marchando</prompt>
4         <elseif cond = "type == 'condition'"/>
      <prompt>incorporando una condicion</prompt>
6     <assign name = "sintaxis" expr = "true"/>
      <assign name = "limit" expr = "true"/>
8         <elseif cond = "type == 'then'"/>
      <prompt>Despues?</prompt>
10    <assign name = "verb" expr = "true"/>
      <assign name = "body" expr = "true"/>
12    <assign name = "axis" expr = "true"/>
          <elseif cond = "type == 'atonce'"/>
14    <prompt>a la vez</prompt>
      <assign name = "sintaxis" expr = "true"/>
16    <assign name = "verb" expr = "true"/>
      <assign name = "body" expr = "true"/>
```

```
18      <assign name = "axis" expr = "true"/>
        <assign name = "limit" expr = "true"/>
20    <elseif cond = "type == 'selseq'"/>
        <prompt>abrimos opciones</prompt>
22      <assign name = "sintaxis" expr = "true"/>
        <assign name = "verb" expr = "true"/>
24      <assign name = "body" expr = "true"/>
        <assign name = "axis" expr = "true"/>
26      <assign name = "limit" expr = "true"/>
    <elseif cond = "type == 'loop'"/>
28      <assign name = "sintaxis" expr = "true"/>
        <assign name = "verb" expr = "true"/>
30      <assign name = "body" expr = "true"/>
        <assign name = "axis" expr = "true"/>
32      <assign name = "limit" expr = "true"/>
  <elseif cond = "type == 'runseq'"/>
34      <prompt>Ejecutamos la secuencia</prompt>
        <prompt>#runseq</prompt>
36      <assign name = "sintaxis" expr = "true"/>
        <assign name = "verb" expr = "true"/>
38      <assign name = "body" expr = "true"/>
        <assign name = "axis" expr = "true"/>
40      <assign name = "limit" expr = "true"/>
          <elseif cond = "type == true"/>
42      <prompt>\item=Oh</prompt>
    </if>
44  <if cond = "type != 'action'">
      <assign name = "sintaxis" expr = "true"/>
46  </if>
       </filled>
48    </field>
```

Para cada tipo de directiva se autocompletan los campos posteriores cuya información no es requerida. Así, el algoritmo `FIA` entre en dichos campos. Así por ejemplo, si el usuario ha realizado una directiva en el que ordena al robot ejecutar la secuencia, el tipo involucrado sería `type = 'runseq'` y, por tanto, el resto de campos (`sintaxis`, `verb`, `body`, `axis` y `limit`) no tienen sentido para esta orden. El comando "ejecutar secuencia" se realiza mediante la línea

```
0       <prompt>#runseq</prompt>
```

que al comenzar con el carácter #, será interpretado por el intérprete-#$.

De este modo, mediante la percepción del acto de diálogo se logra establecer el contexto de edición: si se trata de añadir un nodo, varias ramas en paralelo, un bucle, un comando referencial o ejecutando la secuencia.

9.2.3. Percepción del resto de atributos

La recopilación del resto de atributos se realiza como se ha explicado en la sección 8, mediante el siguiente código:

```
0
    <field name = "sintaxis">
2     <prompt>No acabo de entender la estructura de la frase</
          prompt>
      <filled>
4       <log>sintaxis = <value expr = "sintaxis"/></log>
        <if cond = "sintaxis == 2">
6         <assign name = "limit" expr = "true"/>
        <elseif cond = "sintaxis == 3"/>
8         <assign name = "axis" expr = "true"/>
        </if>
10    </filled>
    </field>
12
   <field name="verb">
14      <prompt>Me falta el verbo.</prompt>
        <prompt cond = "type== 'action'">Dime el nombre de la
            accion</prompt>
16      <prompt cond = "type == 'condition'">Dime que me vas a
            hacer</prompt>
     <filled>
18     <log>verb = <value expr = "verb"/></log>
     </filled>
20   </field>

22   <field name="body">
      <prompt>Dime la parte del cuerpo involucrada</prompt>
24    <filled>
        <log>body = <value expr = "body"/></log>
26 <if cond = "(body == 'Neck')&amp;amp;(sintaxis == 1)">
   <assign name = "axis" expr="true"/>
28 </if>
      </filled>
30   </field>

32   <field name="axis">
       <prompt>Izquierda o derecha?
34     </prompt>
       <filled>
36       <log>axis = <value expr = "axis"/></log>
       </filled>
38   </field>
```

```
40    <field name="limit">
        <prompt>Hasta donde?</prompt>
42      <filled>
          <log>limit = <value expr = "limit"/></log>
44      </filled>
      </field>
46
      <field name="cardinal" cond = "type == 'atonce'">
48      <prompt>Cuantas ramas en paralelo quieres que abra?</prompt
          >
        <filled>
50        <log>cardinal = <value expr = "cardinal"/></log>
  <prompt><value expr="cardinal"/></prompt>
52  <assign name = "ordinal" expr = "cardinal"/>
        </filled>
54    </field>

56    <field name="ordinal" cond = "type == 'atonce'">
        <prompt>A que rama te refieres?</prompt>
58      <prompt>Primera, segunda, tercera...</prompt>
        <filled>
60        <log>ordinal = <value expr = "ordinal"/></log>
  <prompt><value expr="ordinal"/></prompt>
62      </filled>
      </field>
```

En todos los campos, se incluyen preguntas al usuario, en caso de que el enunciado no contemple información que haya podido completar estos campos, o bien, que no hayan sido *autocompletados* por campos anteriores. En este caso, el robot tomaría el turno para realizar estas preguntas. El algoritmo `FIA` se quedaría detenido en el primer campo, en orden de aparición, que no esté completado, hasta que lo esté, o bien, se produzca algún evento en el diálogo (como `nomatch` (fallo en el reconocedor) o `noinput` (tiempo de silencio del usuario sobrepasado))

El campo `tocado` se completa ante eventos de tacto en el robot. Es un campo ligado a información no verbal y por tanto, no tiene asociado un acto de diálogo concreto. Esto se especifica mediante una condición de entrada en este campo: que el campo anterior `da` no esté definido (o completado).

```
0
      <field name="tocado" cond = "da == undefined">
2       <filled>
  <prompt>\item=Laugh_01</prompt>
4       </filled>
      </field>
```

Cuando el usuario toca al robot, se produce este evento, el `FIA` entra en este campo y lo que ejecuta es la emisión de una pequeña risa.

9.2.4. Construcción de la secuencia

Una vez completados todos los campos, lo cual puede haberse hecho en turnos sucesivos en los que el robot haya preguntado al usuario sobre la información concreta que requiere, se ejecuta el contenido de la siguiente etiqueta de tipo `<block>`, muy semejante al contenido explicado en la sección 8.4.2.

Adición de una acción

Por un lado está la asignación de las funciones asociadas a una etapa, en caso de que se esté añadiendo un acción:

```
<block>
 <if cond = "type == 'action'">
   <if cond = "sintaxis == 1">
      <if cond = "body == 'Arm'">
        <assign name = "clase" expr = "axis + body"/>
      <else/>
        <assign name = "clase" expr = "body"/>
</if>
      <assign name = "method" expr = "verb"/>
      <assign name = "position" expr = "0.7 + '*' + body +
         limit"/>
    <elseif cond = "sintaxis == 2"/>
      <assign name = "clase" expr = "body"/>
      <assign name = "method" expr = "verb"/>
      <assign name = "position" expr = "body + axis"/>
    <elseif cond = "sintaxis == 3"/>
      <assign name = "clase" expr = "'Base'"/>
      <assign name = "method" expr = "verb"/>
      <assign name = "position" expr = "'Base' + limit"/>
   </if>
  <assign name = "FuncionActivacion" expr = "clase + '.' +
     method + '(' + position + ')\n'"/>
  <assign name = "FuncionActivacion" expr = "
     FuncionActivacion + 'returnAD = 1\n'"/>
  <assign name = "FuncionTerminacion" expr = "'p = '+ clase +
      '.' + 'get' + verb + 'Pos()\n'"/>
  <!--assign name = "FuncionTerminacion" expr = "
     FuncionTerminacion + 'if( abs( p ) &gt;= abs( ' +
     position +' ) ) :\n'"/-->
```

```
    <assign name = "FuncionTerminacion" expr = "
        FuncionTerminacion + 'if( abs( p -' + position + ')/('
        + position + ')&gt;= 0.10 ) :\n'"/>
    <assign name = "FuncionTerminacion" expr = "
        FuncionTerminacion + '\\ returnAD = 0\n'"/>
    <assign name = "FuncionTerminacion" expr = "
        FuncionTerminacion + 'else :\n'"/>
    <assign name = "FuncionTerminacion" expr = "
        FuncionTerminacion + '\\ returnAD = 1\n'"/>

     <log>funcActivacion =<value expr = "'\n' +
         FuncionActivacion"/></log>
     <log>funcTerminacion =<value expr = "'\n' +
         FuncionActivacion"/></log>
     <prompt>#funcionActivacion$<value expr = "
         FuncionActivacion"/></prompt>
     <prompt>#funcionTerminacion$<value expr = "
         FuncionTerminacion"/></prompt>
     <if cond = "lastresult$.confidence &lt; 0.5">
       <goto next = "#confirmAction"/>
     <else/>
       <prompt>#action</prompt>
       <prompt>ya esta</prompt>
     </if>
```

Adición de una condición

Por otro lado, tenemos la asignación a las funciones asociadas a una transición, en caso de que se esté añadiendo una condición. Se muestra, tanto la adición de una condición explícita, donde `type == 'condition'`, como el caso de la adición de una condición implícita, o *nodo then*, donde `type == 'then'`:

```

    <elseif cond = "type == 'condition'"/>
      <assign name = "FuncionTransicion" expr = "'c = conditions
          .' + verb + '(&quot;' + body + '&quot;,&quot;' + axis
          + '&quot;)\n'"/>
      <assign name = "FuncionTransicion" expr = "
          FuncionTransicion + 'returnAD = c\n'"/>
      <log>funcTransicion= <value expr = "'\n' +
          FuncionTransicion"/></log>
      <prompt>#funcionTransicion$<value expr = "
          FuncionTransicion"/></prompt>
      <prompt>#condition</prompt>
      <assign name = "nConditions" expr = "nConditions + 1"/>
```

```
8         <prompt>hecho!</prompt>
      <elseif cond = "type == 'then'"/>
10        <prompt>#then</prompt>
          <prompt>sigamos!</prompt>
12        <assign name = "nConditions" expr = "nConditions + 1"/>
          <log>nConditions = <value expr = "nConditions"/></log>
14    </if>
```

Adición de varias ramas secuenciales en paralelo

En este bloque también se analiza si se están añadiendo varias secuencias en paralelo. Estas, como se sabe, pueden ser de dos tipos. O bien secuencias simultáneas:

```
0     <if cond = "type == 'atonce'">
        <prompt>Hago <value expr = "cardinal"/> ramas a la vez.</
            prompt>
2       <prompt>#atonce$<value expr = "cardinal"/></prompt>
      </if>
```

O bien, una selección de secuencias:

```
0     <if cond = "type == 'selseq'">
        <prompt>Hago una seleccion de <value expr = "cardinal"/>
            ramas.</prompt>
2       <prompt>#selseq$<value expr = "cardinal"/></prompt>
      </if>
```

En ambos casos, el valor del cardinal establece el número de ramas secuenciales que se van a abrir. Estas ramas pueden cerrarse explícitamente, ante el adecuado acto de diálogo del usuario.

Creación de un bucle

Para crear una estructura de bucle:

```
0     <if cond = "type == 'loop'">
        <prompt>Volvemos al principio</prompt>
2       <prompt>#loop</prompt>
      </if>
```

En caso de existir varias ramas en paralelo abiertas, el comando `#loop` primero cierra la clausura de dichas ramas, mediante una etapa trivial. Luego une esta etapa trivial con la primera de la secuencia.

Cambio del foco de atención

Cuando se están editando varias ramas en paralelo, el usuario puede referirse a una u otra mediante una operación referencial:

```
0     <if cond = "da == 'sd'">
        <prompt>En la rama <value expr = "ordinal"/></prompt>
2       <prompt>#branchFocus$<value expr = "ordinal"/></prompt>
      </if>
```

Con ello se realiza la asignación del foco de atención a una rama concreta, que viene dada por el valor asociado al campo `ordinal`.

9.2.5. Ejecución de la secuencia

La secuencia creada hasta el momento, puede ejecutarse mediante el comando explícito:

```
0     <if cond = "type == 'runseq'">
        <prompt>que voy!</prompt>
2       <prompt>#runseq</prompt>
        <assign name = "inicio" expr = "true"/>
4       <goto next = "#runseq" />
      </if>
```

La secuencia es cargada en el secuenciador y este comienza a interpretarla. El flujo del diálogo, entonces se mueve al formulario `runseq`, que será explicado más adelante.

Por último, el formulario `node`, termina borrando todos los campos para volver a comenzar. Se vuelve así, al principio del diálogo.

```
0     <clear namelist = "da type sintaxis verb body axis limit
          cardinal ordinal"/>
      <goto next = "#node" />
2   </block>
  </form>
```

Como se ha comentado anteriormente, cuando se está ejecutando una secuencia, se abandona el diálogo asociado a la edición y se comienza un diálogo nuevo, dentro de la misma aplicación, denominado `runseq`. De este diálogo, se puede salir simplemente mediante el acto de diálogo `[e]`, esto es, *salida*, o mediante el comando "parar secuencia." Al final de este formulario el flujo vuelve al diálogo `node`.

```
0   <form id = "runseq" >
      <field name = "da">
2       <prompt>\item=Ehi</prompt>
```

```
      <prompt>Mira que bien lo hago!</prompt>
      <prompt>Cuando quieras que pare dimelo</prompt>
      <filled>
  <if cond = "da == 'e'">
    <prompt>Saliendo. besos!</prompt>
  <elseif cond = "da == 'fc'"/>
    <prompt>Adios, un besote!</prompt>
        </if>
  <exit/>
      </filled>
    </field>

    <field name = "type">
      <filled>
  <if cond = "type == 'stopseq'">
    <prompt>Paramos la secuencia</prompt>
    <prompt>\item=Laugh</prompt>
    <prompt>#stopseq</prompt>
  <elseif cond = "type == 'rmseq'"/>
    <prompt>Paramos y borramos la secuencia</prompt>
    <prompt>#stopseq</prompt>
    <exit/>
  </if>
  <assign name = "n" expr = "0"/>
  <goto next = "#node"/>
      </filled>
    </field>
  </form>

</vxml>
```

10. CONCLUSIONES Y TRABAJOS FUTUROS

10.1. Aportaciones de la tesis

El trabajo desarrollado en esta tesis realiza aportaciones en tres líneas principales, estrechamente relacionadas entre sí.

Por un lado, se han expuesto las características más importantes de los **robot sociales**, los retos que plantean y las bases de diseño que han servido para la creación de *Maggie*, robot social del RoboticsLab, y plataforma de investigación en interacción humano robot utilizada en el desarrollo de esta tesis. Por otro lado, se ha realizado una propuesta novedosa dentro del campo de la incipiente **programación natural por un usuario final**, al combinar un sistema de diálogo con un sistema de creación de diagramas funcionales.

Se ha desarrollado un sistema de acceso mediante voz a la edición de un diagrama funcional o secuencia en el estándar `SFC`, el **editor verbal de secuencias.** Esto ha implicado el diseño de unas gramáticas semánticas que relacionan los enunciados del usuario con las funcionalidades del robot.

Se ha desarrollado también, un **sistema gestor de diálogo** de iniciativa mixta, basado en huecos de información, que permite interpretar cualquier diálogo escrito en el estándar `voiceXML`. A través de este sistema se mejora el proceso de construcción de una secuencia en el robot, al incluir interacción conversacional.

El trabajo previo que lleva al diseño e implementación del sistema gestor de diálogo, y que parte del estudio de la **interacción presencial entre humanos** incluye un importante compendio de referencias y conceptos que se proponen como esenciales a la hora de diseñar e implementar un sistema que gestione la interacción en sentido amplio, esto es, incluyendo multimodalidad y autonomía (véase apéndice A).

Estas aportaciones se describen con mayor detalle a continuación.

10.1.1. Robots sociales

Uno de los objetivos planteados es el de realizar una reflexión, que permitiera comprender y ofrecer ciertas claves sobre el deseo de crear robots sociales. Atendiendo a las cifras de producción y económicas, a la cantidad de publicaciones y congresos científicos relacionados, así como a las referencias a los robots sociales desde cualquier ámbito del arte y de la cultura, podemos afirmar que existe un gran interés en el ser humano en los robots personales.

Así que se han analizado distintos aspectos que necesariamente debe contar un robot social. En el diseño externo, se debe atender al denominado *valle siniestro*, donde el usuario muestra un rechazo natural a una creación con un aspecto *demasiado* humano. En el diseño interno, el robot social plantea una serie de requisitos y problemas de los cuales se da una descripción concreta, que sirve como punto de partida hacia su resolución. Estos requisitos se relacionan con el cómo programar un robot social de modo natural por el usuario final, lo cual resulta de especial interés para una utilización más fácil, intuitiva y cercana de los robots sociales.

Finalmente se presenta a Maggie, que ha sido fruto de un trabajo en equipo, como una plataforma de investigación en la resolución de los requisitos que plantean los robots sociales.

10.1.2. Sistema de gestión de diálogo

Una aportación a destacar que proporciona la presente tesis es el sistema de gestión de diálogo desarrollado. Este sistema está basado en la definición de un estándar que realiza el `W3C` (Word Wide Web Consortium). En la actualidad existen tres paradigmas principales para la gestión de diálogos según esté basado en huecos de información, en un planificador, o en la correspondencia entre patrones. Se escoge el primer paradigma porque es el que mejor se adapta a las exigencias del sistema de programación natural y no requiere de terceras partes que se encarguen de crear modelos del mundo difíciles de manejar y con múltiples limitaciones.

El sistema de diálogo implementado cuenta además con algunas ventajas respecto a otros sistemas de diálogo desarrollados en otros robots:

- Es de iniciativa mixta, con lo que tanto el usuario como el robot pueden tomar la iniciativa en cualquier parte del diálogo.
- Atiende a otros aspectos temporales del diálogo, como la gestión del silencio, las irrupciones y la entonación en la expresión hablada.

- Se acerca a la interpretación del lenguaje natural al incluir un sistema de reconocimiento de voz utilizando una gramática de contexto libre, que incluye asignaciones semánticas.

- Permite percepción multimodal, al incluir un módulo capaz de representar cualquier percepción del robot, en el estándar `NLSML` e incorporar esta representación como entrada en el sistema de diálogo.

- Permite expresión multimodal, al incluir un protocolo no contemplado en la definición del estándar `voiceXML` que habilita el envío de órdenes a otras partes de la arquitectura, por ejemplo, a habilidades de expresión multimodal.

- Realiza una asociación directa entre los datos que se quieren percibir mediante diálogo, los atributos semánticos de la gramática del reconocedor y los campos que se incluyen en el sistema de diálogo.

- El sistema de diálogo se interpreta en tiempo de ejecución, con lo que permite ser modificado mientras el propio diálogo se está realizando. Esta modificación en tiempo real o en tiempo de ejecución afecta tanto al diálogo en sí, como a las reglas semánticas de la gramática, que también pueden modificarse dinámicamente.

10.1.3. Edición y ejecución de secuencias `SFC`

Otra aportación importante de esta tesis es el sistema de edición verbal de diagramas funcionales, o secuencias. Este editor trata de resolver cómo describir verbalmente una secuencia, que es una estructura gráfica. Para ello se han realizado estudios previos con personas describiendo secuencias, de los cuales se deduce que el usuario realiza dos operaciones verbales: creación de nodos y de estructuras secuenciales, y operaciones referenciales a partir de la secuencia creada.

Se ha implementado un repertorio inicial de acciones y condiciones en Maggie. En el dominio de acciones se incluyen movimientos que involucran todos sus grados de libertad: cabeza, brazos y base. En el dominio de condiciones se atienden a eventos que puedan llegar de los sensores de tacto. Los dominios implementados pueden ser ampliados *ad líbitum*.

El sistema de edición de secuencias permite construir las tres estructuras elementales que define el estándar de `SFC`: secuencia simple, selección de secuencias y secuencias simultáneas. Además se incluye la posibilidad de

realizar un bucle. Considerando que cualquier secuencia puede ser descrita en términos de estas estructuras básicas, el editor así construido, permitiría entonces editar cualquier secuencia que el usuario desee.

La edición de secuencias incluye además la posibilidad de realizar operaciones referenciales, es decir, realizar una mención a una parte de la secuencia mediante voz. Así, al tener varias ramas secuenciales en paralelo abiertas el sistema permite colocar el foco de edición en cualquiera de ellas, según mencione el usuario.

Como ejemplo, se han presentado varios sistemas concretos de diálogo para la edición verbal interactiva de secuencias. En uno de ellos, la iniciativa en el sistema es llevada principalmente por un usuario que ya conoce la herramienta. El robot, por su parte, es capaz de presentar una ayuda contextual, es decir, a preguntar por la información que le resulta esencial para ir construyendo la secuencia, y de este modo solucionar problemas que puedan surgir por errores en el reconocimiento del habla o errores por pérdida de la coherencia en la interacción. El robot hace así las veces de tutor, guiando al usuario paso por paso, en el proceso de crear la secuencia.

10.2. Limitaciones y trabajos futuros

Esta tesis es, ante todo, un comienzo. El objetivo principal era el de comenzar ciertas líneas de investigación en la interacción entre un humano y un robot social. El sistema implementado ha permitido realizar algunas aportaciones en el tema y obtener algunas conclusiones, pero no está exento de ciertas limitaciones que pasamos a describir. En esta sección se exponen los trabajos futuros más relevantes que pueden surgir de las contribuciones de esta tesis.

10.2.1. Mejoras en el editor verbal de secuencias

Por un lado, se proponen mejoras que involucran la comunicación entre los distintos módulos del sistema global: sistema de diálogo, editor de secuencias y secuenciador. Por otro lado, se proponen mejoras más generales que involucran al editor en sí.

Ampliación de las lineas comunicativas en el sistema

En la figura 10.1 se ha representado el sistema de edición de secuencias mediante interacción. Respecto al diagrama representado en la figura 8.1,

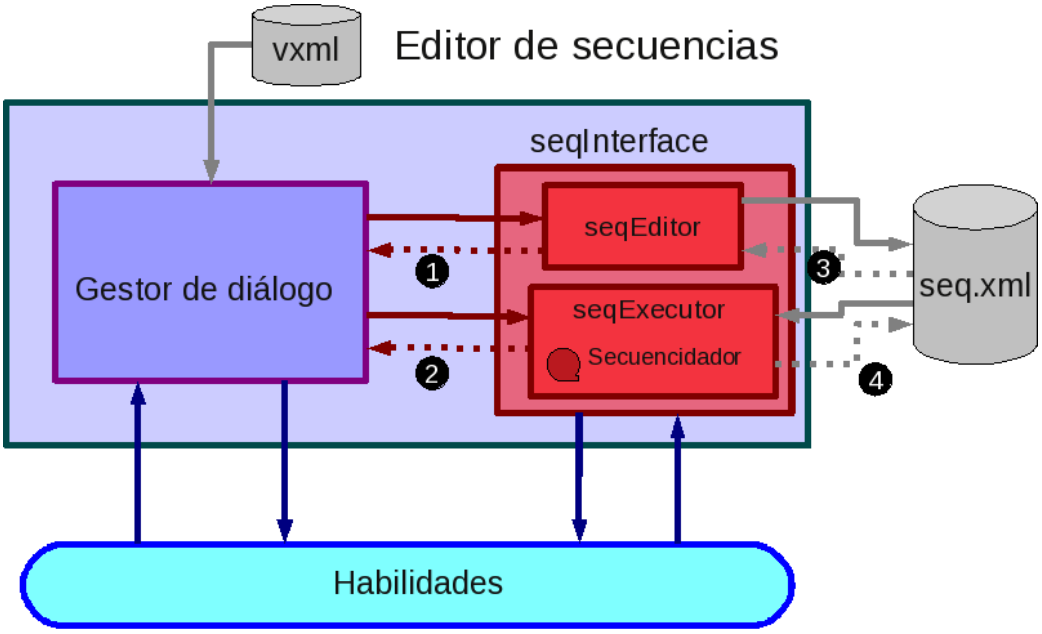


Fig. 10.1: *Conexiones nuevas para el actual sistema de edición verbal interactivo de secuencias.*

donde se realizaba la descripción de dicho sistema, se han incorporado cuatro conexiones adicionales que ampliarían el proceso de edición de un modo muy sencillo. Estas mejoras son las siguientes:

1. **Conexión del editor de secuencias al sistema de diálogo.** Actualmente, el sistema de diálogo directamente envía órdenes al editor de secuencias para crear una secuencia, sin embargo, el sistema de diálogo no puede realizar consultas directas sobre las características de la secuencia que se está creando. Mediante esta conexión el sistema de diálogo puede conocer directamente datos sobre la secuencia. De este modo, el robot podría responder ante cuestiones que el usuario plantee sobre la secuencia creada: cuántas etapas tiene, qué transición precede a qué etapa, etc.

2. **Conexión del ejecutor de la secuencia o secuenciador, al sistema de diálogo.** Actualmente, el sistema de diálogo es capaz de enviar un comando de ejecución o pausa al secuenciador, sin embargo, no tiene acceso directo al estado de ejecución de la secuencias. Por ejemplo, no puede consultar qué etapas están activas en un momento dado, qué transiciones son ciertas, etc.

3. **Lectura del editor de la secuencia de una secuencia ya creada.** De este modo, el editor no tendría que realizar la edición de una secuencia desde cero, sino que podría partir de una secuencia ya creada y reeditarla.

4. **Escritura del estado de ejecución de una secuencia ya creada.** Actualmente el secuenciador se limita a leer y ejecutar una secuencia representada mediante un fichero `XML`. Se propone que el secuenciador pueda guardar el estado de ejecución de una secuencia, por ejemplo si se interrumpe su ejecución. Esto se realizaría reescribiendo el vector de marcas en el fichero que representa la secuencia.

Secuencias reutilizables

Se propone que una vez construida una secuencia, ésta pueda reutilizarse como parte de otra secuencia que la incluya. En la definición del estándar de SFC, esta secuencia reutilizable se denomina *macro*. Una macro se incorpora dentro de una secuencia como una etapa más, resultando transparente para la nueva secuencia su contenido.

Cada secuencia nueva puede guardarse con un identificador, un nombre definido por el usuario, el cual incorporaría esta nueva secuencia como una macro simplemente nombrando el nombre establecido, al igual que se nombra una acción dentro del dominio de acciones. Por tanto, la incorporación de macros implica cambios en la gramática del sistema gestor de diálogos, que debe incorporar el nuevo nombre de la nueva acción-macro. Esto es factible fácilmente, dado que la gramática se representa como un fichero de texto, y es interpretada de manera dinámica en tiempo de ejecución.

Editor inteligente

Hasta ahora, la edición de la secuencia se basaba en una mención explícita, por parte del usuario, de las acciones y condiciones del dominio de primitivas. Se propone también una edición más inteligente, en la que el usuario pueda subir de nivel en su edición y hacer mención a acciones o condiciones más complejas que impliquen una combinación secuencial de acciones o condiciones primitivas. Por ejemplo, que el usuario pueda añadir una acción compleja como "ir al despacho de Ángela" que no está explícitamente considerada dentro del dominio de acciones primitivas y que involucra una resolución, en forma de una secuencia de acciones o plan que lleva al robot hasta el despacho de Ángela.

10.2.2. Mejoras en el sistema de diálogo

Generación automática de gramáticas

El sistema de diálogo implementado permite una recuperación de errores del reconocedor automático del habla mediante interacción conversacional, haciendo así el sistema bastante robusto para la percepción del mensaje del usuario. No obstante, el reconocedor de voz presenta ciertas limitaciones importantes. El hecho de que el usuario tenga que terminar un enunciado para que el reconocedor devuelva un resultado es una limitación importante a nivel temporal. O a nivel de contenido, se propone un sistema que incremente dinámicamente las gramáticas: su léxico, sus reglas sintácticas, e incluso sus asignaciones semánticas.

Esto puede realizarse incorporando una herramienta de dictado en paralelo al reconocedor de voz. Ambas herramientas de reconocimiento de habla pueden trabajar coordinadamente. La herramienta de dictado no utiliza gramáticas para el reconocimiento y se limita a "traducir" la información acústica en texto literal. Este texto puede ser utilizado para construir dinámicamente gramáticas para el reconocedor automático del habla.

Incorporación de un modelo del mundo y del "sí mismo"

Un modelo del mundo implica incorporar en el robot un modelo de los objetos y de los usuarios con los que el robot va a interactuar. Los objetos pueden involucrar desde el espacio en el que el robot se mueve: habitaciones, puertas, pasillos, etc, hasta objetos típicos que pueda manejar el usuario: juguetes, medicinas, etc. También implica un modelo del usuario: datos personales, posición, posible estado emotivo, modelo de los conocimientos o creencias del usuario, qué es capaz de ver, etc.

Del mismo modo, el modelo del mundo también podría incorporar un modelo sobre el robot mismo: estado de las baterías, posición gestual y respecto a otros objetos del entorno, estado de ejecución como qué habilidades están activas, en qué estado de ejecución están, etc. En este sentido, el modelo que tenga el robot sobre sí mismo puede ser considerado como un caso particular del modelo del mundo.

Un modelo de cada objeto implica modelar las operaciones posibles con dicho objeto. Por ejemplo, saber que una puerta se puede cruzar, un pasillo se puede recorrer, una medicina puede ser portada por el usuario, etc.

El modelo del mundo debe permitir realizar operaciones internas de razonamiento dentro del modelo. Por ejemplo, saber que cuando se cruza una

puerta se deja de estar en la habitación actual, o si un usuario está cerca del robot quiere decir que el usuario está en la misma habitación que el robot, etc.

Por último, el modelo del mundo debe poder ser accesible en los dos sentidos, para entrada y salida. Como entrada, para la realización de consultas por parte del usuario. Como salida, como materia prima de generación del lenguaje por parte del propio robot. Por ejemplo, el usuario puede preguntar al robot sobre su nivel de las baterías, sobre qué está ejecutando en ese momento, o, si está ejecutando una secuencia, que describa en lenguaje natural cómo es la secuencia, etc.

Generación de lenguaje natural

En el apartado 5.1.1 se introdujeron algunos procesos involucrados en la generación de lenguaje natural. Básicamente el punto de partida de la generación del lenguaje es doble: por un lado, el contenido que se quiere comunicar, y por otro, la relación que se quiere establecer con el usuario.

El contenido objeto de la generación está relacionado con la cooperación del robot para mantener una coherencia con el usuario, lo cual puede involucrar algún tipo de deliberación que establezca un objetivo comunicativo concreto. Para generar dicho contenido el robot utiliza su modelo del mundo descrito anteriormente.

La generación de lenguaje natural además implica un modelo morfosintáctico de la lengua, que permita realizar las inflexiones necesarias de género, número, tiempo, modo, etc.

Desarrollo de percepción y acción multimodal

La interacción natural es multimodal. El sistema de diálogo implementado permite incorporar fácilmente información multimodal tanto en expresión como en percepción, gracias a los sistemas descritos en la sección 7.3.

En expresión, se proponen las siguientes vías de desarrollo:

- **Control prosódico.** Aumentar la expresividad acústica del habla. Además de la expresividad en el habla que se explica en la sección 5, también se han realizado algunos trabajos previos con la herramienta `MBROLA`. En ellos, se generan curvas envolventes del tono a partir de una base de datos que se capta del usuario en tiempo real. Este trabajo ha sido publicado en [Gorostiza and Salichs, 2008] y [Gorostiza and Salichs, 2006].

- **Expresión de gestos emotivos.** Es decir gestos que muestren tristeza, alegría, sorpresa (incluso susto), enfado, etc. También se ha comenzado a diseñar un método de generación de estos gestos a partir de un conjunto discreto de parámetros.

- **Expresión de gestos discursivos**, donde se incluyen también señales anafóricas. Son gestos como afirmar, negar, expresar que se está prestando atención o que se está en un estado pensativo, etc.

- **Expresión audiovisual.** Se propone diseñar un modelo donde se establezcan unos parámetros y unas reglas entre estos parámetros para articular mediante imágenes y sonido un discurso emotivo, afectivo e incluso conceptual. Puede realizarse un discurso más figurativo o bien seguir las líneas teóricas más abstractas descritas en [Kandinsky, 1912b] y [Kandinsky, 1912a], así como trabajos del mundo del arte visual como Oskar Fischinger[1], Hans Richter[2] o Chris Cunningham [3].

La expresión multimodal no deja de ser una expresión de un lenguaje, con lo que la generación del mensaje multimodal puede seguir procesos análogos a los descritos anteriormente en la generación de lenguaje natural.

Análogamente para la percepción de gestos, se proponen las siguientes vías de desarrollo:

- **Percepción de características sonoras de la voz del usuario.** Para así identificar su edad, su sexo, su estado anímico, su intencionalidad en sus actos comunicativos, etc.

- **Percepción de características sonoras del entorno.** Por ejemplo, si hay ruido en una sala. También percepción de características sonoras de la música. Por ejemplo identificación del ritmo de la música, para desarrollar una habilidad de baile.

- **Percepción del ritmo de la expresión del usuario.** Mediante la percepción del ritmo en sus movimientos al hablar unido a una percepción de las pausas y la articulación de énfasis, etc. La percepción del ritmo entre interlocutores resulta de vital importancia dado que en la interacción presencial estos ritmos tienden a influenciarse.

[1] véase por ejemplo *Early Abstarctions* http://www.youtube.com/watch?v=RrZxw1Jb9vA
[2] véase por ejemplo *Rythm.21* http://www.youtube.com/watch?v=QEgULqLn5iU
[3] véase por ejemplo *gantz graf* http://www.youtube.com/watch?v=s4ZwTUUue1w&feature=related

- **Percepción de gestos deícticos**, que ayuden a resolver referencias anafóricas como "aquí", "este", etc. O que ayuden a identificar magnitudes expresadas mediante gestos.

- **Percepción de gestos táctiles en pantalla.** Dado que Maggie cuenta con una pantalla táctil en su pecho, el usuario puede expresar alguna opción pulsando cierta información mostrada en una parte de la pantalla, arrastrando iconos, etc.

10.2.3. Métodos de evaluación

Se propone, por último, crear un método de evaluación que permita informar de la eficiencia comunicativa entre el usuario y el sistema de interacción conversacional. El sistema debería incorporar al menos las siguientes características:

- **Criterios de evaluación.** Donde se establecen unas métricas que indiquen qué es lo que se quiere medir: como es la amigabilidad del sistema, la cercanía con el usuario, su facilidad de uso, etc. Medidas también de la sensación de coherencia por parte del usuario, de la sensación de empatía, etc.

- **Métodos de evaluación.** Diseñar cómo se va a realizar la evaluación. Por ejemplo, la realización de una serie de *test* antes y/o después del experimento. Establecer qué variables son significativas en la evaluación. Por ejemplo, el tiempo de interacción, los errores en la actuación del robot, errores de entendimiento del usuario, ..., y relación de estas variables con los parámetros que se quieren medir, etc.

- **Resultados de las evaluaciones.** Es decir, diseñar cómo se va a presentar esta evaluación. Entre otras cosas, se deberá incluir también una descripción de la población de estudio: niños, adolescentes, personas mayores, etc.

10.3. Algunas palabras finales

A lo largo de la historia, encontramos muchas referencias en el arte y en los mitos a una especie de necesidad del ser humano a tener un compañero igual[4] a él (/ella?), con el que compartir la existencia. Así, en los mitos sobre *Pigmalión y Galatea*, sobre *Prometeo*, en el *Golem* del rabino Loew, en la creación de *Talos* por Dédalo, en las mujeres doradas con movimiento propio de *Hefestos*, en el Génesis judío, etc. En todos estos mitos se habla de una criatura que no es animal, ni humana. Es creada por el propio ser humano y muy semejante a él. Le sirve de compañero, de esclavo, de ayudante, etc.

También en la literatura encontramos referencias a esa criatura *igual, pero distinta* como en *Frankestein* de Mary Shelly, en *El hombre de arena* y en *Los autómatas* de E.T.A. Hoffmann, y más cerca de la actualidad toda la literatura sobre los robots de Isaac Asimov o en la obra *¿Sueñan los androides con ovejas eléctricas?* de Phillip Dick, etc. Obras donde la criatura que se muestra resulta un tanto siniestra, y menos amigable en la medida en que es más semejante a su propio creador con el que se llega a confundir.

Es difícil definir qué es un robot[5]. El origen del vocablo no es científico, sino que surge por primera vez en una obra de teatro: *R.U.R. (Rossum's Universal Robots)* de Karel Capek (1921). El término viene del polaco *robota*, y significa *trabajo*, prestación personal, etc, a veces con cierto sentido peyorativo: trabajo desagradable, difícil, duro o inmoral. Y este origen del término en un escenario de teatro ya indica una estrecha relación existente entre la ficción y la realidad de los robots sociales.

Finalmente concluimos que cierta idea de robot personal o robot social ha estado presente en el ser humano desde el origen mismo de su cultura. Y no solo eso, sino que la idea de robot personal también tiene aspectos en común con todas las artes, en la medida en que producen representaciones de la realidad. No hay que olvidar que arte y técnica nacieron de la mano[6]. Pero además, en todo arte está el proceso de creación. Este concepto está muy ligado de nuevo al mito de Prometeo, que crea a los hombres; a Hefestos, dios técnico por antonomasia, que crea a la primera mujer, Pandora; a Pigmalión que crea a una mujer ideal Galatea, a Victor Frankestein que crea su criatura sin nombre, ... Todos estos mitos responden a un vacío del ser humano. Los robots sociales podrían cubrir de manera real ese vacío.

[4] ...pero distinto

[5] La R.A.E define el término como: *Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas.*

[6] El término de origen griego *tecné* significa tanto arte como técnica.

Surge entonces la siguiente cuestión: *¿qué modificaciones sufrirá el ser humano al compartir la existencia con una criatura tan semejante?*

APÉNDICE

A. INTERACCIÓN PRESENCIAL ENTRE SERES HUMANOS

A.1. Modelos de la comunicación humana

Se ha considerado importante incluir una breve referencia a algunos de estos modelos por varias razones. La primera, porque sirven de base para el diseño de un sistema de comunicación como es el sistema de diálogo que se presenta más adelante en este trabajo. La segunda, porque, si bien, por límite de tiempo y de contenido, no se llega a implementar gran parte de los conceptos que en esta sección se van a presentar, queremos dejar constancia de cuáles de estos conceptos nos resultan esenciales a la hora de diseñar cualquier sistema de interacción con el ser humano, y de ese modo guiar los trabajos futuros que abre esta tesis. Además, queremos hacer constancia de la importancia de mantener una investigación pluridisciplinar en el campo de la interacción humano robot, que tenga en cuenta el estudio que sobre la comunicación humana se realiza fuera del ámbito ingenieril[1] [Ansina, 1989] [Mortensen, 1972].

A.1.1. Modelo de Shannon-Weaver

La aparición a partir de 1945 de los trabajos de Wiener, Shannon y Weaver, entre otros, sobre los sistemas de comunicación, que dieron origen a lo que luego se denominará como *cibernética*, supone el comienzo de un desplazamiento del énfasis de las *intenciones* a los *efectos* en el estudio de la conducta comunicativa humana. Se pasa de una perspectiva fundamentalmente introspectiva a otra predominantemente predictiva, centrada en los fenómenos de *interacción*, y por tanto, más observable, rigurosa y científica.

[1] Asimismo, no nos resulta casual el hecho de que los conceptos que se exponen en esta sección más cercanos al campo de la lingüística, la psicología o la sociología hayan dado, por otro lado, origen al campo de conocimiento en el que se encuadra el presente trabajo: la Automática.

El modelo de Shannon-Weaver, se centra en la eficiencia en la *transmisión* de la información, que es codificada mediante un *código* en una serie de *símbolos* o *mensaje*, y transmitido mediante una *señal* por un *canal* con *ruido*, desde un *emisor* a un *receptor*. Este modelo es quizás el más conocido debido a sus repercusiones actuales en la sociedad tecnológica de la información, lo cual ha sido posible por su gran precisión matemática en la descripción del modelo. Además de los conceptos de emisor, receptor, canal, etc, el modelo realiza una nueva e interesante contribución a la concepción de *información*.

Por un lado, el modelo incorpora una necesidad de *interacción* entre los dos sistemas que se comunican para poder definir el concepto de información. La información es una magnitud que comparten dos conjuntos. Es decir, que un ente se puede considerar información cuando contiene algo que es común al receptor. Una cadena de símbolos produce mensajes, pero en sí mismos no constituyen información. Solamente después de que dicha cadena sea correlacionada con cierto mensaje que el receptor conoce, la cadena refleja información.

Por otro lado, define una medida matemática de la información. Para ello, se hace necesario definir una medida de la incertidumbre del mensaje asociada al hecho de que el mensaje es codificado mediante símbolos pertenecientes a un alfabeto finito y transmitido por un canal cuya capacidad también es finita. Además, el modelo presupone que el receptor no tiene ningún modo de predecir qué símbolo le va a llegar en cada instante.

A esta incertidumbre se la denomina *entropía de Shannon*. Información se define como la entropía de correlación o entropía mutua entre dos variables aleatorias. Dichas variables se refieren al símbolo del mensaje en el sistema emisor y al del mensaje en el sistema receptor. Así, la información mide la correlación entre dos conjuntos. Dicho de otro modo, emisor y receptor "se están entendiendo" en un instante dado cuando su correlación es máxima, es decir, cuando ambos manejan el mismo símbolo (véase [Shannon, 1948]).

A.1.2. Modelo de Schramm.

Este modelo da más énfasis, respecto al modelo de Shannon-Weaver, al concepto de interacción entre los interlocutores, en el proceso de comunicación presencial. Codificación y decodificación son procesos que ambos, emisor y receptor realizan simultáneamente. Por tanto, la comunicación entre los interlocutores, en este modelo es bidireccional.

Para Schramm lo más importante en la comunicación es el hecho de que la fuente y el destino estén *sintonizados*, es decir, que la experiencia acu-

mulada en ambos tenga cada vez más elementos en común. A este nuevo concepto lo denomina *campo de experiencia* que representa el conocimiento de cada individuo y que en la medida en que se solape más fácil resultará la comunicación. Así, el modelo da importancia también al *contexto* de cada uno de los interlocutores. El mensaje tiene muchas más probabilidades de éxito si guarda consonancia con las actitudes, valores y metas del receptor. De alguna manera, estas ideas están ligadas con un nuevo concepto que va a ser desarrollado ampliamente por los cibernéticos y que abrirá un nuevo campo de conocimiento: *la realimentación*.

Otra contribución importante que recoge el modelo de Schramm, es la característica plural del canal de comunicación, y, por tanto, de los distintos *modos* existentes en la comunicación interpersonal. Los seres humanos, cuando se comunican presencialmente, envían y reciben informaciones por varios canales, y no solo por el canal verbal. Estos nuevos canales, se refieren a los modos no verbales: gestos, relación espacial, etc, y que, como se verá más adelante, resultan de gran importancia en la interacción interpersonal. En la interacción humano robot, esta pluralidad coincide con el concepto, presentado anteriormente, de *multimodalidad* (véase 3.1.1).

A.1.3. Funcionalismo lingüístico de Jackobson

La lingüística estructural comienza a partir de la obra de Saussure al desentenderse de la diacronía en el hablante, es decir, del aspecto temporal del lenguaje ([de Saussure, 1916]) Surgen así conceptos como el de *significante* y *significado*[2], y de ahí la *semántica*, entendida como *la ciencia encargada del estudio del significado de los signos lingüísticos*[3]. Al estudiar el lenguaje como una estructura, se produce un giro hacia la importancia de las relaciones de sus constituyentes en detrimento del análisis de sus elementos.

Por otro lado, entendiendo el lenguaje como producto de la actividad humana, surge así el concepto de *acto de comunicación verbal*, que en el presente trabajo va a ser recuperado en los actos del diálogo (véase 7.1.2). Como acto humano el lenguaje adquiere un carácter teleológico o de finalidad: el individuo, siguiendo una intención propia, utiliza el lenguaje como acto para llevarla a cabo.

El análisis del lenguaje, desde este punto funcional y estructural (relacio-

[2]El significante podemos relacionarlo con la parte literal de las gramáticas utilizadas en nuestro sistema de reconocimiento automático del habla (véase 4.1.3), y el significado con su parte "semántica."

[3]Definición obtenida de la RAE

nando sus elementos) llevará a Jackobson a realizar un esquema de *funciones del lenguaje* [Sebeok, 1966] basado en el modelo de Shanon-Weaver y en las funciones que ya distinguía el lingüista Karl Bühler (1879-1963):

- **Función emotiva o expresiva**, asociada al destinador, hablante o emisor y a su relación con el mensaje. Revela el aspecto afectivo y subjetivo de la comunicación y da cuenta de la sinceridad del hablante.
- **Función referencial**, asociada al contexto, a la información en sentido estricto, a su aspecto denotativo y da cuenta del carácter verdadero del mensaje.
- **Función poética**, orientada hacia al mensaje y al aspecto estético de la comunicación, donde reside su belleza.
- **Función fática**, asociada al contacto que se produce entre los interlocutores. Son mensajes cuya finalidad es la de establecer, prolongar o interrumpir la comunicación. En ese sentido, está relacionada con las normas de educación y convenios sociales.
- **Función metalingüística**, asociada al código de la comunicación. Suele ir implícita en la articulación del propio mensaje, y establece, por parte del destinador qué código resulta correcto para una comunicación efectiva.
- **Función conativa**, define las relaciones entre el mensaje y el destinatario, del cual el destinador espera una reacción. Esta función está relacionada con la legitimidad del hablante sobre la actitud que quiere provocar en el destinatario.

Así, cada función del lenguaje está asociada a un elemento dentro del sistema de comunicación: al hablante, al destinatario, al contexto, etc. Estas funciones, por tanto, se dan concurrentemente en un mismo acto verbal, aunque en cierta jerarquía, existiendo así una función primaria que adquiere mayor importancia respecto al resto.

A.1.4. Teoría de la Comunicación Humana de la escuela de Palo Alto

En la paráfrasis realizada en los distintos modelos anteriores puede observarse cierta importancia progresiva que se va dando a la interacción entre los interlocutores (modelo de Schramm) así como una progresiva complejidad en varios niveles en el acto comunicativo (modelo de Jackobson) Esta progresión culmina en el modelo sistémico que proponen los trabajos de los discípulos de Gregory Bateson (uno de los creadores de la cibernética) en la escuela de Palo Alto [Paul Watzlawick, 1967]

Este modelo surge de investigaciones sobre las *paradojas* en el lenguaje y sobre ciertas patologías psíquicas y su relación con el modo en que se comunican los seres humanos. Dichas patologías mostraban cierto *equilibrio* o *estabilidad* ante cualesquiera fueran las entradas al sistema. Para explicar esta estabilidad, se construye un nuevo modelo basado en la *retroalimentación* que supone el medio, el contexto, etc, en la actitud del individuo, al contrario que el psicoanálisis que interpretaba la psique humana como un sistema determinista, o la psicología analítica que utilizaba modelos teleológicos (orientados a un fin) Los sistemas retroalimentados o realimentados, se caracterizan precisamente, por su capacidad para mostrar estabilidad, *equifinalidad* u homeostásis en alguna de sus variables, es decir, cierta independencia del sistema ante las condiciones iniciales. La realimentación va a ser retomada para el control de sistemas donde resulta esencial ([Ogata, 1993])

En la visión clásica, que surge del modelo de Weaver-Shannon, se consideraba como ítem de entrada el acto comunicativo (la emisión de un mensaje) que funcionaba como *estímulo* por parte del emisor hacia el receptor, el cual *responde* con otro ítem provocando una nueva reacción en el emisor que denominamos *refuerzo.* En el actual modelo, basado en la idea de retroalimentación, el ítem considerado, al mismo tiempo, presenta la característica de estímulo, respuesta y refuerzo, dado que no se entiende un acto comunicativo sin sus causas, y, por tanto, no existe un acto comunicativo inicial.

Se abandona así la noción causal de que un hecho comunicativo ocurre primero y determina un hecho siguiente, sino que se adopta una visión circular. Esta visión en el modelo no tiene porqué ser compartida por los interlocutores[4] cuya acción la ven como una reacción ante la acción del otro y no como, simultáneamente, una causa de la acción del otro.

El modelo acentúa el carácter holístico de la comunicación humana: cada

[4]De hecho esta es una de las causas principales de patologías psicológicas debidas a la comunicación

parte está relacionada de un modo tan sensible con las otras partes, que un cambio en una de ellas provoca un cambio en las demás y en el sistema total. Por tanto, al contrario que en los modelos anteriores, éste establece la imposibilidad de desmenuzar en unidades independientes linealmente causales la dinámica de la interacción. Describe características globales de la comunicación humana. Estas caracterísiticas quedan definidas en cinco "axiomas exploratorios de la comunicación" de un modo bastante informal más que exhaustivo, y que pasamos a enumerar y a desarrollar a continuación:

1. Toda conducta es comunicación.
2. Toda comunicación tiene dos aspectos: contenido y relacional.
3. La naturaleza de una relación (entre interlocutores) es función de la puntuación dinámica de secuencias de hechos.
4. El contenido de toda comunicación tiene dos aspectos: digital y analógico.
5. La interacción entre los interlocutores puede ser de dos tipos: simétrica o complementaria.

Imposibilidad de no comunicar.

Efectivamente, consideremos el caso más singular: el silencio (verbal y no verbal) o, lo que en teatro ha venido a denominarse "máscara neutra". Bueno, pues aun en este caso extremo el individuo está comunicando: su aspecto físico (sexo, altura, edad, ropa (si la lleva), etc), su postura, etc. E incluso, dependiendo del contexto, un individuo en silencio puede comunicar precisamente que no quiere comunicar nada.

Además, el axioma defiende cierta "inercia" en el proceso comunicativo. Es decir, una vez comenzado, el proceso de comunicación tiende a mantenerse en el tiempo. Así, por ejemplo, si un individuo se dirige a otro, indefectiblemente éste se verá en la *obligación* de estar ya metido, sin ninguna escapatoria, en el proceso de comunicación y, por tanto en la obligación de manifestar una reacción limitada por el contexto. Dicha reacción puede ser de cuatro tipos:

- Rechazo. Es una forma descortés y, a pesar de todo, mantiene ya un vínculo entre los interlocutores (en este caso, un vínculo "negativo")
- Aceptación. Esta reacción provoca que la inercia de la conversación se mantenga.

- Descalificación del proceso, es decir, autocontradicciones, incongruencias, cambios de tema, tangentes, oscuridad, metáforas mal interpretadas, etc. Reacciones que provocan la pérdida de sintonía entre lo interlocutores.

- Síntoma como comunicación. En este caso, el individuo reacciona desplazando la responsabilidad de comunicar a una causa mayor externa: "A mí no me molestaría hablarte pero algo ajeno a mi voluntad me lo impide."

El hecho de que el robot vaya a estar siempre comunicando y que sus movimientos vayan a entrar en uno de los cuatro tipos de reacciones al proceso de comunicación tiene sus implicaciones tanto en el diseño externo como en el diseño de sus movimientos comunicativos. El tipo de reacción a escoger, en la comunicación por parte del robot, va a depender del rol con el que queramos que se vaya a presentar de cara a su relación con el usuario.

Comunicación en dos aspectos: contenido y relación.

El primero de los aspectos hacer referencia a la función referencial, relacionada con el contexto, de Jakobson. El aspecto relacional hace referencia a la función conativa, asociada a la relación que tiene el destinatario con el mensaje recibido. Es decir, que en el proceso de comunicación aparecen dos planos paralelos: en uno de ellos discurre el contenido, el tema, las relaciones al contexto, la información en términos absolutos que se está compartiendo; en el otro plano discurre un establecimiento de roles jerárquicos entre los interlocutores: el emisor ofrece un rol que el receptor admite o no.

El aspecto relacional clasifica el aspecto de contenido y es, por ende, una *metacomunicación:* una misma frase puede tener significados totalmente distintos según el receptor acepte o no el rol que está ofreciendo el emisor. Cabe destacar que esta sensación de la relación es subjetiva y, por tanto, hipotética.

Ocurre así, que en la interacción presencial los individuos definen una *mismidad* o autodefinición: "así es como me veo en relación con el *otro* en esta *situación.*"

El otro, o alter, a su vez, en su respuesta ante la autodefinición del emisor, le comunica cómo le concibe: "así es como yo te veo." Esta respuesta puede seguir alguno de los siguientes patrones:

- Confirmación. Esta reacción, crea una base cargada de motivos para que los interlocutores vuelvan a comunicarse por la comunicación misma.

- Rechazo. Éste no necesariamente debe interpretarse con carga negativa en la relación de los interlocutores. El rechazo puede ser constructivo, en la medida en que el receptor comunica y explica el rol que está dispuesto a aceptar del emisor.

- Desconfirmación. Provoca una especie de enajenación o pérdida de la mismidad. En este caso, el emisor recibe una falta total de aprecio por parte del receptor que reacciona como si el otro no existiese.

La relación entre dos participantes, una vez expuesta por ambos, puede ser complementaria o simétrica. En el primer caso, uno de ellos está en cierto modo "subordinado" al otro. En el segundo, ambos se mantienen en un mismo nivel jerárquico en su relación.

Por tanto, la sensación de "ser entendido" por parte de un individuo no solo reside en el hecho de que la información que este comparte (nivel de contenido) se comprendida por el otro, sino que también comprendida la identidad, rol o mismidad que el individuo ofrece al otro, es decir, que haya entendimiento a nivel relacional.

Puntuación de secuencias de hechos.

Si bien, la interacción entre comunicantes, esto es, su intercambio de mensajes, no se modela como "una secuencia ininterrumpida de intercambios", si que existe una organización de los hechos de la conducta comunicativa, que Bateson y Jackson denominan "puntualización de secuencias de hechos." En todo proceso de interacción existen unos patrones de intercambio que, en una comunicación eficiente y no patológica, son acordados por ambos interlocutores, consciente, inconsciente o extraconscientemente. Esta puntualización no es tanto una señalización de cuándo tomar el turno en la dinámica de la interacción, sino que señala la visión subjetiva en el participante del aspecto causal del acto comunicativo, esto es, señala al interlocutor si un acto comunicativo de él mismo o del otro es causa o consecuencia de otro acto comunicativo del otro o de él mismo.

Así se expone un ejemplo de una conflicto matrimonial debido a una patología en la comunicación, en concreto, en el establecimiento no acordado de las pautas de puntuación de secuencias. El esposo afirma que en defensa contra los constantes regaños de su mujer, reacciona con un retreaimiento pasivo, mientras que su mujer afirma que las críticas vienen por la actividad pasiva de su marido. De modo que constantemente están en conflicto pues

cada uno interpreta la actitud del otro como causa de su propia actitud y ninguno rompe esta *realimentación positiva*, que hace al sistema inestable.

Comunicación analógica y comunicación digital.

Esta dicotomía entre los dos tipos de comunicación equivale a la consideración de una comunicación verbal y otra no verbal. De este modo la comunicación no verbal, en la medida en que realiza las representaciones mediante semejanza autoexplicativa, es analógica. La comunicación verbal es no analógica, dado que, salvo en ciertos casos como la onomatopeya, no existe ninguna correspondencia entre la palabra y el hecho o cosa que se designa.

La comunicación digital tiene una característica de continuidad temporal inherente que la hace muy adecuada para compartir información acerca de objetos del entorno así como para la transmisión de conocimiento a lo largo del tiempo. Además en su cercanía a la representación matemática, el lenguaje digital cuenta con una sintaxis lógica poderosa y compleja. Todo esto hace que el lenguaje digital (o verbal) sea la herramienta más adecuada para la transmisión de todo lo que tiene que ver con el aspecto de *contenido* de la comunicación. Sin embargo carece de la semántica adecuada en el campo de la relación.

El lenguaje analógico posee la semántica pero no una sintaxis adecuada para la definición inequívoca de la naturaleza de las relaciones. Por tanto el aspecto de *relación* en la comunicación está predominantemente marcado por la comunicación analógica. Sin embargo esta carece de la potente sintaxis lógica del lenguaje digital: carece de la negación o de estructuras como "si luego...", "o...o..", etc. De hecho el lenguaje analógico cuenta con cierta cualidad de ambigüedad inherente. Así por ejemplo, un individuo puede llorar de alegría, pero también de tristeza; un puño cerrado puede mostrar simpatía o desprecio, etc.

El ser humano, en su comunicación interpersonal, va traduciendo de un modo a otro. Además de la pérdida de información, esta traducción no es inmediata y está sujeta a una significativa ambigüedad, lo cual es fruto de múltiples conflictos y faltas de entendimiento entre los interlocutores.

Interacción simétrica y complementaria.

A largo plazo, las relaciones pueden estar basadas en dos tipos de *gestald*: igualdad o diferencia. En el primer caso, los participantes tienden a igualar su conducta y por tanto se habla de una interacción *simétrica.* En el segundo caso, aparece un fenómeno que Bateson ha venido a denominar *cismogénesis*,

o generación de una escisión o división que caracteriza la interacción complementaria. En esta división, uno de los participantes adquiere un rol de superioridad respecto al resto.

En ambos casos, el rol adquirido por los participantes es estable y cabe destacar el "carácter de mutuo encaje" en el que ambas conductas interrelacionadas tienden a favorecer la una a la otra (realimentación)

Sirva como resumen las palabras de uno de los investigadores más importantes en el desarrollo de modelos de comunicación no verbal, Ray L. Birdwhistell, en relación a la comunicación humana interpersonal: *Un individuo no comunica; participa en una comunicación o se convierte en parte de ella.* [Birdwhistell, 1970] Esto último coincide con el concepto que el biólogo colaborador con la escuela de Palo Alto Humberto Maturama ha pasado a denominar *lenguajear*, que defiende el hecho de que solamente hay comunicación cuando hay coordinación consensuada de acciones y conductas entre los interlocutores.

A.2. Sobre el lenguaje natural

Al contrario de cualquier lenguaje *artificial* o como ha venido a denominarse, lenguaje *formal*, el lenguaje natural se define, evoluciona y se manifiesta como cualquier otra cualidad natural del ser humano, es decir, desde su interior. Su construcción no ocurre bajo un control central y es a posteriori, cuando surgen reglas, normas y descripciones que intentan formalizarlo para su estudio. La descripción más generalizada de un lenguaje natural pasa por su definición en términos de fonética, morfología, sintaxis, semántica y pragmática, según la clasificación realizada en [Morris, 1938].

La Fonética estudia el aspecto más físico de todo lenguaje natural: su acústica. Se basa en un elemento fundamental, el fonema: *cada una de las unidades fonológicas mínimas que en el sistema de una lengua pueden oponerse a otras en contraste significativo.*[5] Así, el conjunto de sonidos de un idioma queda representado mediante sus fonemas. El fonema no es indivisible y está formado por *moras* Dentro de la fonética distinguimos la prosodia, *como el estudio de los rasgos fónicos que afectan a unidades inferiores al fonema, como las moras, o superiores a él, como las sílabas u otras secuencias de la palabra u oración*, que se refieren a la métrica, a los acentos y a su cantidad. El estudio prosódico del lenguaje en la interacción entre humanos es de suma importancia, dado que es en este nivel de segmentación donde

[5]Definición obtenida de la RAE

reside gran parte de la información asociada a la intención del hablante y la focalización del tema del que habla.

La Morfología analiza la estructura de las palabras, esto es, sus partes o morfemas: afijos (prefijos, sufijos e interfijos según su posición dentro de la palabra) y lexema o raíz de la palabra. Es en el lexema donde suele residir el campo semántico de la palabra. El resto de morfemas establecen mediante flexión el género, el número, la persona, el tiempo, el modo o el aspecto de una palabra, según su categoría gramatical (si es sustantivo, verbo, adjetivo, etc...) Estas flexiones resultan de gran importancia a la hora de relacionar unas palabras con otras y establecer enlaces semánticos entre ellas.

La Sintaxis establece en categorías gramaticales una clasificación de todas las palabras de un lenguaje natural. También estudia los sintagmas y su ordenación en oraciones. En el lenguaje natural, por su aspecto oral, definimos *enunciado*, como sintagmas que no tienen necesariamente que estar sintácticamente completos y que pueden contener elementos no verbales, pero que forman sentido. La sintáctica estudia la estructura temática (sintagma nominal y sintagma verbal) la jerarquización y dependencia (agrupación de palabras en torno a la relación de atribución predicativa), la concordancia entre sintagmas y las agrupaciones de oraciones, así como, las modalidades oracionales: declarativas, interrogativas, exclamativas, imperativas, asertivas, afirmativas y negativas.

La Semántica se ocupa del significado en el lenguaje. Está más cerca de su parte más interna y, por tanto, más oscura: las ideas. El *sema*, o unidad mínima de significación dentro de una locución, comparte a la vez un aspecto semántico puramente denotativo con otro aspecto connotativo. Ambos son de difícil análisis. En los diccionarios, así como en otras estructuras ontológicas tiende a definir semánticamente una palabra en su aspecto más denotativo. Pero probablemente es el aspecto connotativo de una palabra el responsable de que los seres humanos tengamos la impresión de que nos entendemos.

La Pragmática se refiere a los efectos que tiene en la conducta de los interlocutores su intercambio de signos. Se encarga, por tanto, del estudio de la parte más externa del lenguaje natural: las relaciones formales entre comunicación y conducta. En este sentido, la pragmática defiende que el proceso de comunicación en lenguaje natural entre individuos en interactivo, frente a un modelo unidireccional.

Por tanto, cada una de estas disciplinas se centra en el estudio de una característica distinta del lenguaje natural. Ahora bien, resulta inevitable, que existan elementos acoplados entre las distintas disciplinas. Ocurre así, por ejemplo, que la prosodia está íntimamente ligada a la pragmática, en la

medida en que es la parte acústica de la locución la que expresa gran parte de la intención del hablante para con su interlocutor. Así también, no resulta casual que sea en el lexema, la parte más invariable de la palabra, que es estudiado por la morfología, donde resida la raíz del campo semántico de la palabra, que es estudiado por la semántica. Por otro lado, si bien, la sintáxis se encarga de estudiar las reglas remanentes en las locuciones de un lenguaje natural, sin entrar en su contenido semántico, solamente podrá la semántica estudiar el significado de una locución, sintagma u oración si esta es necesariamente sintácticamente correcta; o como en el principio composicional de muchos lenguajes, el significado de un enunciado está en la estructura de su sintáxis. Y así, podemos establecer más elementos de relación entre unas disciplinas con otras.

En la comunicación humana, la tendencia natural es la de establecer vínculos y relaciones tales que la comunicación entre individuos sea cada vez más efectiva, en el sentido de maximizar el intercambio de información, minimizando el intercambio de energía. Por tanto, si algo caracteriza al lenguaje natural es su tendencia a llevar a los usuarios a ser cada vez más ahorrativos en todos los niveles. Fonéticamente la tendencia del hablante es siempre la de omitir ciertos fonemas. Sintácticamente, la tendencia natural puede llevar al hablante no solo a omitir el sujeto de la acción (algo muy común en el castellano, por ejemplo) sino sintagmas enteros que se dan por referidos. Semánticamente también hay omisiones claras, como lo muestran el uso de pronombres, la deixis o la anáfora, donde además el habla natural va acompañado de información no verbal que resulta tan crucial en la articulación del mensaje que tomar por separado la información verbal y la no verbal puede cambiar radicalmente el significado percibido.

A.3. Comunicación no verbal

Puesto que la interacción natural es multimodal, la comunicación no verbal juega un papel importante en la interacción humano robot. La función emotiva según el modelo funcional de Jackobson está asociada a la articulación no verbal de la comunicación humana. Asimismo, el aspecto relacional de la comunicación propuesto por Watzlawick contiene gran caudal de comunicación analógica o, lo que viene a ser lo mismo, comunicación no verbal que acompaña o no al habla.

En la sección 3.1.1, se describían varios requisitos que deben cumplir los robots sociales. El de la multimodalidad está íntimamente relacionado con la comunicación no verbal. El estudio de la comunicación no verbal humana es

por tanto necesario para su aplicación a la interacción humano robot.

Se propone un estudio orientado en dos aspectos: uno en la morfosintáxis de los movimientos corporales, y el otro en su semántica. El primero ha de tratar de definir un "lemario", dominio, repertorio y descripción de gestos y su combinación a lo largo del proceso interactivo. El otro ha de ofrecer una relación entre cada gesto y su significado comunicativo.

Podemos estudiar al cuerpo como elemento expresivo desde dos puntos de vista:

- **Kinésica**, o lenguaje corporal que estudia los gestos corporales cuando los interlocutores están dentro del área próxima de interacción.

- **Proxémica**, estudia el lenguaje corporal según la distancia y posición relativa entre los interlocutores, distinguiendo varias áreas de interacción.

Tanto el estudio de gestos como el de la posición se realiza en los distintos niveles lingüísticos: a nivel *sintáctico*, qué movimientos corporales involucra, a nivel *semántico*, que significado comunicativo tienen esos movimientos, y a nivel *pragmático*, su efecto sobre el comportamiento del interlocutor. Por ejemplo, cuando el hablante mira a los ojos del oyente (nivel sintáctico), en el contexto adecuado significa que está preguntando si el oyente le está prestando atención (nivel semántico), el cual puede reaccionar con un acto comunicativo de afirmación o confirmación (nivel pragmático).

En la recopilación de estudios sobre el tema que se realiza en [Davis, 1971], se tratan los distintos aspectos de la comunicación no verbal de una manera bastante amplia. El estudio del lenguaje corporal puede realizarse separando distintos canales o modos: modo visual, auditivo, táctil y olfativo, o también teniendo en cuenta las distintas partes del cuerpo involucradas, teniendo así gestos faciales, realizados con las manos, posturales, etc. También se analizan escenarios concretos de interacción, como es el saludo o el galanteo. Resulta de especial interés el tratado que se realiza sobre el *ritmo* de la interacción, pues el ritmo juega un papel muy importante en la sensación de empatía o unión entre los interlocutores. Por ejemplo, en el intercambio del turno de la interacción.

En el intercambio del turno entre dos interlocutores se ha observado la siguiente secuencia de escenas:

1. El ritmo del hablante es percibido por el oyente que comienza a moverse en sincronía comunicando así su deseo de tomar el turno.

2. El hablante, por su parte, percibe tal deseo y ambos interlocutores se mueven en sincronía durante un breve periodo de tiempo.

3. El oyente realiza una pequeña variación del ritmo, estableciendo un *tempo* personal.

4. Si el hablante acepta el cambio de turno comienza a moverse con el ritmo establecido por el oyente.

5. El oyente percibe que el cambio de turno ha sido aceptado e irrumpe la conversación.

Se podrían distinguir los gestos en dos tipos:

- **Emotivos.** Se refieren a la expresión de emociones, sentimientos, afecto, estado de ánimo, etc. En [Ekman, 1999] por ejemplo, se muestran las relaciones entre gestos y emociones percibidas, distinguiendo hasta quince emociones básicas: tristeza, miedo, ira, sorpresa, felicidad, diversión, etc.

- **Discursivos.** Es decir, gestos que no son esencialmente emotivos, que acompañan o no a la información verbal y que tienen fuerte carga semántica, comunicacional o, simplemente, de contenido. Una clasificación categórica de estos gestos en emblemáticos, ilustrativos, reguladores, sustitutivos, etc ha sido propuesta en [Ekman and Friesen, 1969].

Estas clasificaciones resultan esenciales para formalizar la expresión y percepción de actos comunicativos no verbales por parte de un robot social.

B. INTRODUCCIÓN AL ESTÁNDAR SOBRE DIAGRAMAS FUNCIONALES SFC

B.1. Introducción

Después de haber realizado un compendio de los principales robots personales o robots sociales que existen en la actualidad, sean prototipos de investigación en laboratorios, como productos comerciales, podemos distinguir, según su arquitectura de control, dos grupos: arquitecturas donde se representa explícitamente un plan o una secuencia de interacción con el entorno, y arquitecturas donde la interacción con el entorno no está representada de un modo explícito. En el primer grupo estarían las arquitecturas de control que incluyen un planificador, en el segundo estarían las arquitecturas basadas en un modelo de las emociones.

En el presente trabajo consideramos la posibilidad de que el robot interactúe con el entorno siguiendo una *secuencia*. Se define secuencia *como una red de acciones y condiciones alternadas*. Entendemos por acción como *cualquier proceso o efecto de la actividad del robot con su entorno* y por condición como *una entidad funcional que puede tomar dos valores, verdadero o falso, según ciertas variables del robot o del entorno, donde se incluye al/los usuario/s*. Cabe destacar, por tanto, que el concepto de secuencia aquí manejado, nno solamente establece una serie de acciones a ejecutar siguiendo cierto orden jerárquico, sino que, en la secuencia también se incluyen condiciones que tienen que ver con el entorno, por tanto, la secuencia no establece simplemente un modo de acción en el entorno, si no un modo de *interacción* con el entorno.

La secuencia se representa como un diagrama secuencial o *SFC (Sequential Function Chart)* estándar definido en `IEC 61131-3`[I.E.C., 1993] donde se establece un lenguaje de representación para *PLC's (Programmable Logic Controllers)* basado en `GRAFCET` *(GRAphe de Commande Etape Transition)*

y en redes de Petri.

La sección comienza realizando una breve revisión de algunas características de las redes de Petri, así como de su representación algebraica dado que dicha representación ha sido utilizada también a la hora de modelar las secuencias SFC.

B.2. Origen del diagrama funcional: redes de Petri

Fig. B.1: *Foto reciente de Carl Adam Petri*

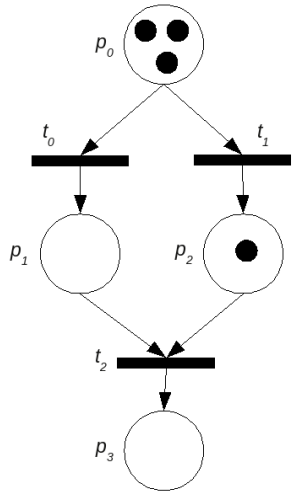
En 1962 Carl Adam Petri publica su tesis doctoral titulada "Comunicación con autómatas", donde propone un nuevo tipo de *red* para representar el funcionamiento sistemas distribuidos. Esta nueva estructura será utilizada hasta la fecha por miles de trabajos en automática, informática e incluso química, y en seguida tomará el nombre de *red de Petri.*

Una red de Petri es un grafo bipartito, que consta de un conjunto de *lugares*, representados mediante un círculo, y un conjunto de *transiciones*, representados por un segmento. Cada lugar puede unirse a una o más transiciones y vicebersa, cada transición puede unirse a uno o varios lugares. Cada lugar, además, puede contener una o más marcas, que indican si el lugar está activo o no. Las marcas pueden moverse a través de las transiciones de un lugar a otro, de acuerdo a unas *reglas de disparo.*

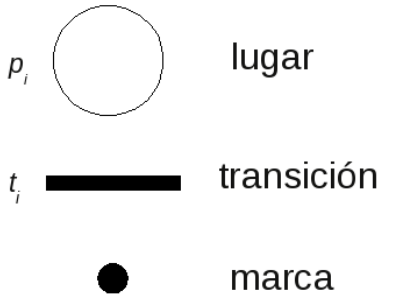
B.2.1. Definición formal de una red de Petri

Si bien existen múltiples definiciones formales de una red de Petri, podemos decir que una red de Petri PN está formada por la *quíntupla* $PN = (P, T, W^-, W^+, M_0)$ donde:

- $P = \{p_0,\ p_1,\ ..., p_n\}$, es el conjunto finito de lugares de la red.



(a) Gráfico de la red de Petri



(b) Elementos de una red de Petri

Fig. B.2: *Ejemplo de una red de Petri*

- $T = \{t_0,\ t_1,\ ..., t_n\}$, es el conjunto finito de transiciones de la red.
- $P \bigcap T = 0$, es decir que ambos conjuntos son disjuntos, o dicho de otra manera, que una red de Petri es un sistema bipartito.
- $W^- : P \times T \mapsto \mathbb{N}_0$, es el conjunto de conexiones de pre-condiciones.
- $W^+ : P \times T \mapsto \mathbb{N}_0$, es el conjunto de conexiones de post-condiciones.
- $M_0 : P \mapsto \mathbb{N}_0$, en el vector de marcas iniciales.

Las matrices W^- y W^+ representan las conexiones entre lugares y transiciones.

La matriz de pre-condiciones W^- representa las conexiones desde los lugares a las transiciones, es decir, los arcos de salida de los lugares. El número natural en la matriz marca un valor de *peso* de salida del lugar. Este valor indica el número de marcas que debe haber en el lugar anterior a la transición para que se ejecute su *función de transición*.

La matriz de post-condiciones W^+ representa las conexiones desde las transiciones a los lugares, es decir, los arcos de entrada a los lugares. El número natura en la matriz indica el número de marcas que se colocarán en el lugar receptor si la condición asociada a la transición es verdadera y hay movimiento de marcas.

Asímismo, asociada a cada transición existe una condición denominada *función de transición.* Esta función deberá ser cierta para que se produzca un movimiento de marcas, como se verá más adelante.

Una de las características más interesantes de una red de Petri es que permite tener varios lugares activos simultáneamente. Este paralelismo es lo que la hace ser una potente herramienta de representación para sistemas distribuídos en general.

De la descripción general definida en este apartado han surgido numerosas variaciones con distintas aplicaciones. Estas variaciones incluyen ciertas limitaciones en cuanto a las conexiones posibles entre lugares y transiciones respecto la definición general. Existen así *máquina de estados, grafos de marcas, redes de libre elección (FC-nets), redes extendidas de libre elección (EFC-nets), red simple (SPL-net) y red simple extendida (ESPL-net).*

Por ejemplo. En la figura B.2 puede verse una red de Petri de 4 lugares y 3 transiciones. Su representación formal sería:

- $P = \{p_0,\ p_1,\ p_2\ p_3\}$, conjunto de lugares de la red.
- $T = \{t_0,\ t_1,\ t_2\}$, conjunto de transiciones de la red.
- $W^-(p_0, t_0) = 2, W^-(p_0, t_1) = 1, W^-(p_1, t_2) = 1, W^-(p_2, t_2) = 3$, es el conjunto de conexiones de pre-condiciones.
- $W^+(p_1, t_0) = 1, W^+(p_2, t_1) = 2, W^+(p_3, t_2) = 1$, es el conjunto de conexiones de post-condiciones.
- $M_0 = (3, 0, 1, 0)$, en el vector de marcas iniciales.

B.2.2. Reglas de disparo en una red de Petri

El movimiento de las marcas en los lugares se realiza de modo discreto en "ciclos". Para que en una transición concreta dicho movimiento se realice, tienen que cumplirse dos condiciones en el mismo ciclo:

1. Todos los lugares previos a la transición deben estar activos, y tener cada uno un número de marcas igual o superior al peso asociado a las respectivas conexiones de entrada a la transición. En este caso se dice que *la transición está activa*
2. La condición asociada a la transición debe ser cierta durante todo el ciclo.

Estas "reglas de disparo" pueden formalizarse con las siguientes definiciones:

1. Sea $M : P \mapsto \mathbb{N}_0$, función de marcas que devuelve el número de marcas del lugar p.

2. La transición $t \in T$ está activa para unas marcas M, $M[t >$ sii:

$$\forall p \in P, M(p) \geq I^-(p,t) \tag{B.1}$$

3. Cuando la transición $t \in T$ se dispara cambia el vector de marcas de M a M', $M[t > M'$, de modo que $\forall p \in P$:

$$M'(p) = M(p) - I^-(p,t) + I^+(p,t) \tag{B.2}$$

4. Definimos $\sigma = t_0, t_1, ..., t_n \backslash n \in \mathbb{N} \wedge n \geq 0$ como secuencia de disparos de modo que existen las marcas $M_0, M_1, ..., M_{n+1}$ tal que:

$$M_i[t_i > M_{i+1},\ \forall i = 0, ..., n \tag{B.3}$$

Por ejemplo Supongamos que en la red de la figura B.2 en un ciclo dado, la condición asociada a la transición t_0 es verdadera, entonces vemamos si se cumple la ecuación B.1, esto es, si la transición está activa. $M(p_0) = 3$ y como $I^-(3,0) = 2$, entonces $M(p_0) > I^-(3,0)$, luego la transición está activa. Así, el disparo $M[t_0 > M'$, siendo $M'(p_0) = M(p_0) - I^-(3,0) + I^+(3,0)$, y como $I^+(3,0) = 0$, entonces $M'(p_0) = 1$ son las marcas con las que se queda el lugar p_0 después de la transición.

B.3. Elementos de un diagrama funcional

Todo diagrama funcional viene definido por tres conjuntos de símbolos:

- **Etapas**, análogas a los lugares de una red de Petri.
- **Transiciones**, análogas a las transiciones de una red de Petri.
- **Arcos** o enlaces entre etapas y transiciones.

El estándar define la secuencia en su aspecto gráfico, de ahí el término de *diagrama*. Por otro lado, cada etapa está asociada a una *acción* y una *condición* a cada transición, de ahí el aspecto *funcional* del diagrama.

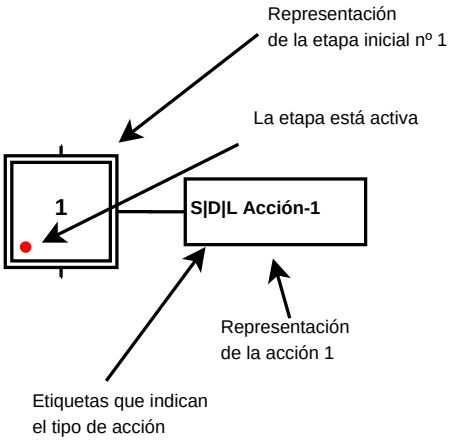


Fig. B.3: *Representación de una etapa en una SFC*

B.3.1. Etapas y acciones

Una etapa caracteriza el comportamiento invariante del sistema en cuestión. Toda etapa puede estar en dos estados exclusivos: activo o inactivo. Cuando una etapa en estado inactivo pasa a estado activo, la acción asociada a la etapa comienza a ejecutarse. Este comienzo está asociado a una *función de activación*. Cuando la etapa pasa de estado activo a estado inactivo, la función asociada se denomina *función de desactivación.*

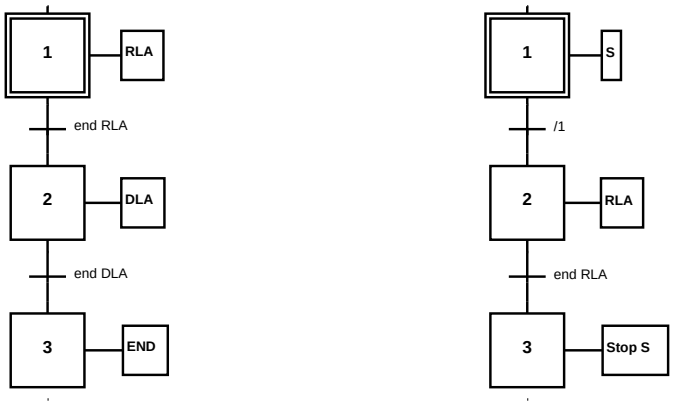
La representación de una etapa es mediante un cuadrado. Si la etapa es una etapa de comienzo, el estándar define su representación mediante un cuadrado con doble línea. Cuando la etapa está activa, se representa mediante un punto en el cuadrado. La acción asociada a la etapa se representa gráficamente mediante un rectángulo a la derecha del cuadrado que representa la etapa.

Puesto que un diagrama funcional o secuencia puede devenir en un esquema relativamente grande, muchas veces resulta útil incluir partes de la secuencia en una sola etapa denominada *macro.* La etapa macro, por dento puede incluir cualquier convinación de las estructuras básicas comentadas, así como otras macros.

Finalización de acciones

La acción asociada a la etapa puede tener o no finalización durante el tiempo en el que la acción permanece activa. Es decir, el paso a un estado inactivo de una etapa no implica la finalización de la acción cuyo comienzo

de ejecución ha ordenado dicha etapa. Una acción que no finaliza cuando su etapa pasa a estado inactivo se etiqueta mediante la letra *S (Store)* Una acción sin final es interrumpida cuando otra etapa lo especifique mediante otra acción de interrupción. Las acciones con final tienen asociada una función de "comprobación de terminación" que sirve para comprobar si la acción ha terminado o no.



(a) Dos acciones (*RLA y DLA*) con finalización propia.

(b) La acción *S* finaliza con otra acción posterior.

Fig. B.4: *Representación de acciones con y sin final dentro de una secuencia simple*

Por ejemplo, en la figura B.4 se representan dos ejemplos de diagramas funcionales donde aparecen acciones con final y sin final. En B.4(a) la acción "levantar brazo izquierdo" (RLA) es sucedida de "bajar brazo izquierdo" (DLA) siempre y cuando la primera acción haya finalizado (condición "fin de levantar brazo izquierdo") Sin embargo en B.4(b) la acción "comenzar a girar" (S) no finaliza por sí misma. Esta acción es seguida de una transición trivial, esto es, una condición que siempre se cumple, después de la cual comienza la acción "levantar brazo izquierdo" Por tanto en este punto, el robot estaría girando mientras también levanta el brazo izquierdo. Esta secuencia simple, finaliza cuando, después de haber terminado de levantar el brazo izquierdo se ejecuta la acción "parar de girar" que mandaría finalizar la acción anterior de "comenzar a girar."

Duración de una acción

El estándar permite asociar un límite de tiempo a la acción que comienza cuando se activa una etapa. Este límite puede refererirse tanto a acciones con final como a acciones sin final. Las acciones con límite de tiempo se etiquetan mediante la letra L junto con la cantidad de tiempo límite.

Asímismo, el comienzo de una acción puede retrasarse desde el momento en que la etapa ha pasado a estado activo. Este retardo se especifica mediante la etiqueta D junto con la cantidad de tiempo de retardo.

B.3.2. Transiciones y condiciones

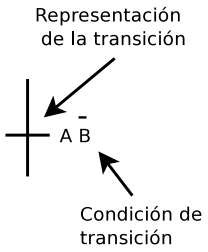


Fig. B.5: *Representación de una transición en una SFC*

Las transiciones caracterizan el comportamiento dinámico del sistema en cuestión, dado que permiten la evolución del estado activo de unas etapas a otras, lo que se denomina *movimiento de marcas.* Cada transición se representa mediante un segmento.

Cada transición está directamente ligada a una función o condición de transición. Esta función puede involucrar variables del entorno o de la propia secuencia. La expresión algebraica booleana de esta función se acompaña al segmento que representa la transición.

Se dice que una transición está activa si todas las etapas conectadas inmediatamente antes están en estado activo. El disparo de una transición ocurre si y solo si está activa y su condición de transición es verdadera. El disparo de una transición involucra el movimiento de marcas, de modo que las etapas precesoras pasan a estado inactivo y todas las etapas inmediatamente posteriores conectadas a la transición pasan a estado activo. Aunque, en la práctica, el tiempo de disparo suele ser tan pequeño que puede despreciarse en comparación a los tiempos característicos del sistema, nunca debe considerarse que el tiempo de disparo es nulo.

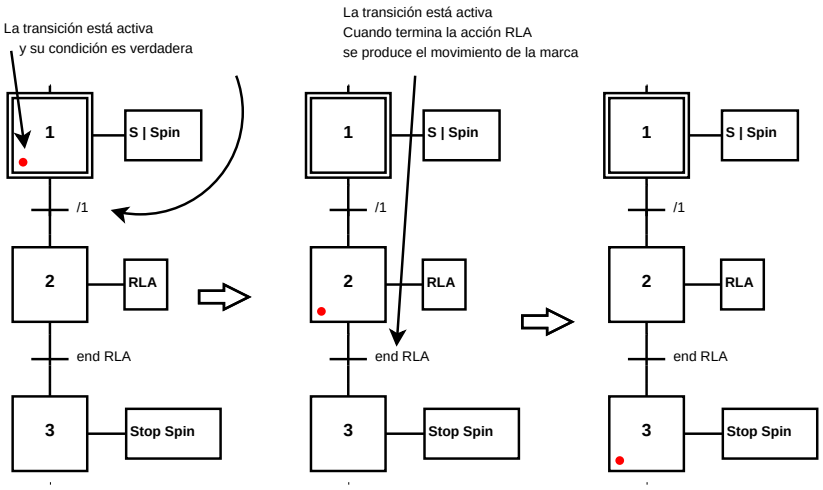


Fig. B.6: *Secuencia de disparos en un diagrama funcional*

B.4. Estructuras básicas de diagramas funcionales

Análogamente a las redes de Petri, no pueden haber dos etapas o dos transiciones unidas entre sí, y todas las uniones en una secuencia tienen que alternar etapa y transición.

Mediante ciertas estructuras elementales en realidad es posible representar cualquier secuencia por compleja que sea. Estas estructuras son tres: secuencia simple, selección de secuencias y secuencias simultáneas.

B.4.1. Secuencia Simple

Una secuencia simple está formada por una serie de etapas seguidas cuyo ciclo de activación-desactivación será sucesivo una detrás de otra. Cada transición, en una secuencia simple, solamente se activa mediante una sola etapa previa. Y en cada disparo cada transición activa solamente una etapa posterior.

B.4.2. Selección de secuencias (modo selección)

En una selección de secuencias se abren varias ramas en paralelo, cada una asociada a una transición, de modo que solamente una de las ramas puede ser activada. De cada transición en la selección solamente puede partir una sola rama.

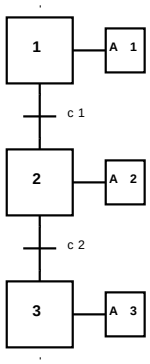


Fig. B.7: *Ejemplo de una secuencia simple*

La condición de transición o selección siempre parte de una etapa origen común a todas las ramas. Para que solamente una de las ramas siga la evolución del estado de activación desde la etapa origen, las condiciones de transición asociadas deben ser exclusivas, de modo que solamente una de ellas sea verdadera en un instante dado.

La apertura de una selección de secuencias corresponde con una convergencia donde todas las ramas selectivas vuelven a unirse en una sola rama destino común. No se permite que exista una transición común a varias ramas, y cada rama debe unirse con la rama destino a través de su propia transición.

B.4.3. Secuencias simultáneas (modo paralelo)

En una apertura de varias ramas de secuencias en paralelo, todas las ramas parten de una misma transición que, una vez se dispara, activa todas las ramas simultáneamente. Una vez activadas, la evolución en cada una de estas ramas es independiente.

Para poder sincronizar la convergencia de las distintas ramas en paralelo en una sola rama principal común se utiliza una misma transición a la que se une la última etapa de cada rama. Por tanto, para que dicha transición se active, todas y cada una de las ramas en paralelo tienen que haber terminado, es decir, tienen que haber activado su última etapa.

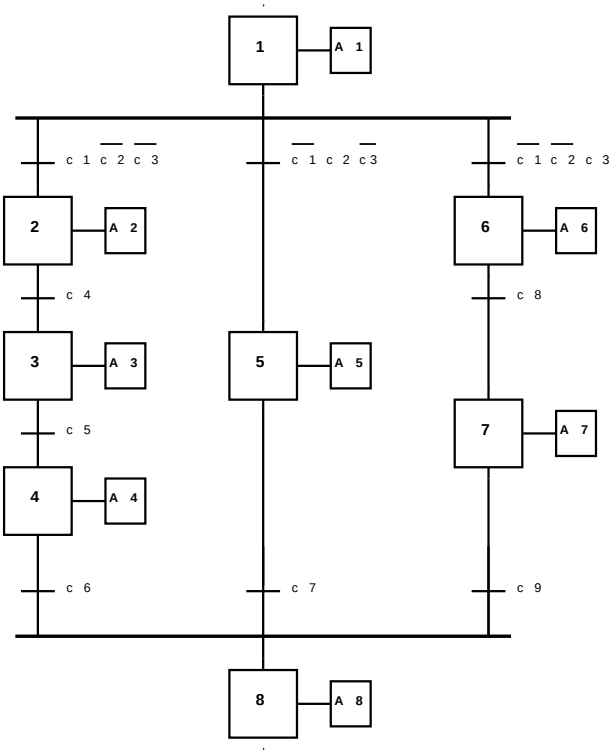


Fig. B.8: *Ejemplo de una estructura en modo selección*

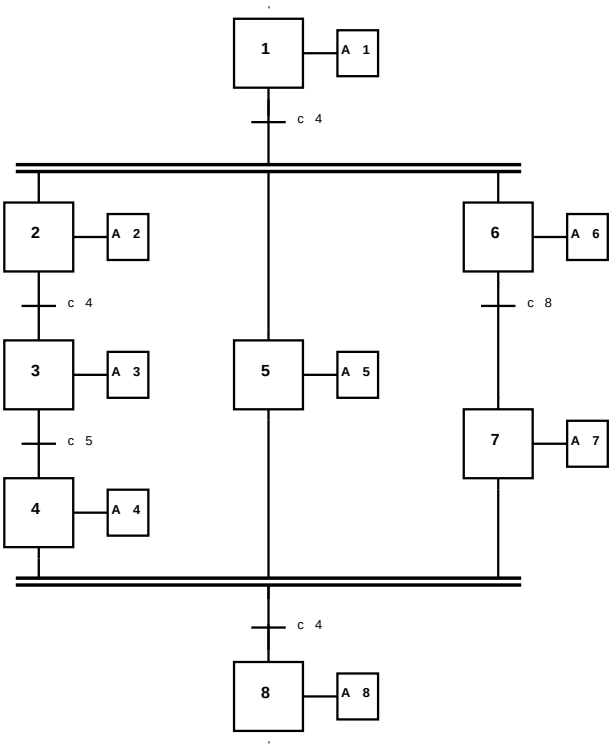


Fig. B.9: *Ejemplo de una estructura en modo paralelo*

C. LENGUAJES FORMALES Y GRAMÁTICAS DE CONTEXTO LIBRE

C.1. Introducción

Podemos concebir un lenguaje como *un conjunto de secuencias de entidades indivisibles representaciones distinguibles de cualquier información, o* símbolos *pertenecientes a otro conjunto no vacío que denominamos* alfabeto. Por ejemplo, el castellano consta de un alfabeto que va del símbolo "a" al símbolo "z" (obviando las letras mayúsculas)

Al definir lenguaje como un conjunto, le dotamos de todas las posibilidades que ofrece la *Teoría de Conjuntos* al respecto: operaciones, relaciones y funciones. Por ejemplo: unión, intersección, diferencia, producto cartesiano, ley conmutativa, distributiva, etc.

Una *palabra* es una *secuencia de símbolos de un alfabeto.* Por tanto un lenguaje es un conjunto de palabras. Un caso particular de palabra es la cadena vacía, ϵ

El conjunto de todas las palabras que pueden ser formadas por un alfabeto Σ se denomina Σ^* Es un conjunto infinito enumerable, esto es, cada elemento del conjunto puede ponerse en correspondencia con un número natural. A partir de este conjunto Σ^* definimos otro conjunto: *clausura de Kleene* (L^*) de modo recursivo como el conjunto más pequeño que contiene los siguientes elementos:

- ϵ, cadena vacía.
- L el conjunto lenguaje.
- Todas las palabras formadas por concatenación del propio L^*

Como se verá más adelante, este conjunto es ampliamente utilizado en lo que se denominan *expresiones regulares.*

Definicion C.1.1. ***Un lenguaje formal*** *es un lenguaje en el que se definen las reglas de construcción de las palabras de las que consta a partir de los símbolos de su alfabeto.*

Por ejemplo Podemos definir el "lenguaje baby" a partir de un alfabeto $\Sigma = b, a, !$ y como regla de construcción de las palabras diremos que todas han de comenzar por el símbolo "`b`" terminar con el símbolo "`!`" y contener en medio de estos uno o más símbolos "`a`" De este modo "el lenguaje baby" constaría de palabras como `ba!`, `baa!`, `baaaaaaaaaaa!`, etc.

El conjunto "clausura de Kleene" del lenguaje baby contendría todas las cadenas del lenguaje y además concatenaciones de estas cadenas: `ba!ba!`, `baa!baa!baaaaa!`, `baaaaaaaa!ba!baaa!`, etc.

C.2. Lenguaje formal, gramática generativa y autómata

Relacionado con el concepto de lenguaje formal, cabe destacar otros dos conceptos también muy relacionados entre sí: gramática generativa (o gramática a secas) y autómata.

Paralelamente al desarrollo de la teoría de lenguajes formales se ha ido desarrollando el concepto de *gramática*, como una herramienta no sólo de generación, sino también de reconocimiento de un lenguaje formal. La construcción de un lenguaje formal se realiza mediante unas reglas representadas en la gramática. La gramática de un lenguaje formal consta de tres elementos:

- **Símbolos terminales.** Son cadenas de caracteres que no pueden subdividirse en otras más pequeñas realizando cualquiera de las reglas que establece la gramática, de ahí su aspecto "terminal" Los terminales suelen representarse mediante letras minúsculas (`a`, `b`,...) Dos terminales especiales en toda gramática es el terminal inicial S y el terminal vacío ϵ El conjunto de terminales se denomina alfabeto y se denota por el símbolo Σ

- **Símbolos no terminales.** Son variables que pueden asignarse a símbolos terminales y no terminales. Se suelen representar mediante letras mayúsculas (`A`, `B`,...) El conjunto de no terminales se denota por el símbolo V

  Mediante letras griegas (α, β,...) suelen representarse símbolos que pueden ser tanto terminales como no terminales. El conjunto de estos

símbolos se denota mediante el conjunto $(V \cup \Sigma)*$, es decir la unión de los símbolos del alfabeto combinados mediante la clausura Kleene con símbolos no terminales que derivan del alfabeto.

- **Relaciones** entre terminales y no terminales que establecen las reglas de generación dentro de la gramática formal. El conjunto de estas relaciones se denota mediante el símbolo R

Según cómo sean las relaciones entre terminales y no terminales así se clasifican los distintos tipos de gramáticas. Tradicionalmente se clasifican siguiendo la *jerarquía de Chomsky* ([Chomsky, 1965]), que establece cuatro tipos de gramáticas:

- **Tipo 0:** $\alpha \rightarrow \beta$, siendo α y β cadenas de terminales y/o no terminales, es decir, que son gramáticas sin restricciones: en cada miembro de la asignación pueden aparecer terminales y no terminales en cualquier orden. Las gramáticas de tipo 0 generan lenguajes denominados *recursivamente enumerables.* Estos son los lenguajes más complejos dentro de la jerarquía.

- **Tipo 1:** $\alpha A \beta \rightarrow \alpha \beta \gamma$, siendo α, β y γ cadenas de terminales y/o no terminales; y A un no terminal. Las gramáticas de tipo 1 generan lenguajes sensibles o dependientes del contexto.

- **Tipo 2:** $A \rightarrow \alpha$, siendo A un no terminal y α una cadena de terminales y/o no terminales. Las gramáticas de tipo 2 generan lenguajes libres del contexto. Estas gramáticas serán las utilizadas en nuestro reconocedor automático del habla.

- **Tipo 3:** $A \rightarrow a$, siendo A un no terminal y a un terminal. Las gramáticas de tipo 3 generan lenguajes regulares. Estos son los lenguajes más simples dentro de la jerarquía.

De este modo obtenemos cuatro tipos de lenguajes formales. La jerarquía de Chomsky, tal y como se ha descrito, establece además una inclusión de un lenguaje sobre el anterior. De este modo podemos enumerar los cuatro lenguajes en orden creciente de complejidad: lenguaje regular, lenguaje libre del contexto, lenguaje sensible del contexto y lenguaje recursivamente enumerable.

También existen otros tipos de lenguajes regulares no considerados en la jerarquía de Chomsky, como son: lenguaje recursivo, lenguaje indexado, lenguaje semisensitivo del contexto, lenguaje determinista libre del contexto

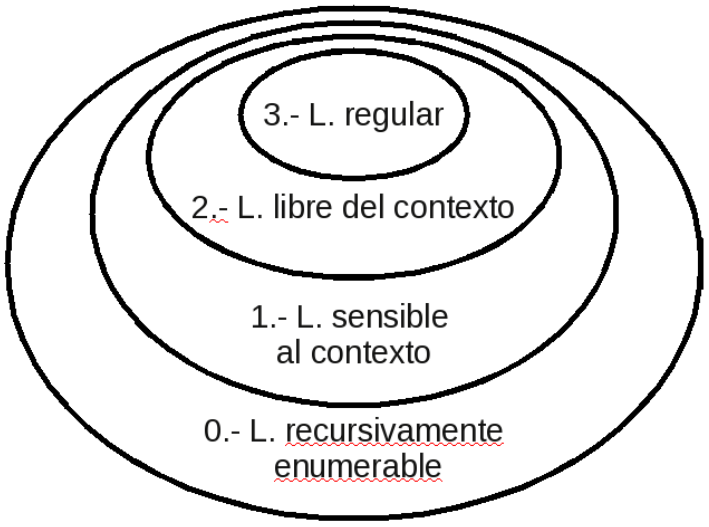


Fig. C.1: *Jerarquía de Chomsky para los lenguajes formales*

y lenguaje libre∗. En la tabla C.1 se representan todos estos lenguajes formales en jerarquía inclusiva, junto con las gramáticas que los generan y los autómatas asociados a estas gramáticas.

Paralelamente al concepto de gramática y de lenguaje formal se ha desarrollado el concepto de *autómata* en distintos tipos: máquina de Turing, de decisión, linealmente acotado, autómata con pila anidada, con pila incrustada, con pila no determinista, autómata con pila determinista, máquina de estados finitos o autómata finito, y autómata aperiódico finito. De hecho gramática y autómata pueden considerarse conceptos equivalentes, en el sentido de que representan la misma información.

Para la presente tesis, la aplicación más importante de un autómata es su capacidad para *reconocer* y *generar* un conjunto concreto de secuencias de símbolos. El reconocimiento se realizaría mediante un algoritmo que haga pasar en serie cada uno de los símbolos de la secuencia. La máquina parte de su estado inicial. Si al finalizar toda la serie de símbolos la máquina llega a alguno de sus estados finales, entonces queda establecido que el autómata ha reconocido tal secuencia de símbolos.

El reconocimiento de una palabra o cadena por parte de un autómata puede expresarse formalmente a partir de la definición de *clausura reflexiva y transitiva de* $\Rightarrow$, que se representa como $\Rightarrow^*$ La clausura reflexiva y transitiva de $\Rightarrow$ simboliza la secuencia de pasos de *derivación* o generación desde el símbolo inicial de un lenguaje hasta la palabra final. De este modo podemos

Jerarquía de Chomsky	Lenguaje	Gramática	autómata
Tipo-0	Recursivamente enumerable	$\alpha \rightarrow \beta$	de Turing
-	Recursivos	-	máquina de decisión
Tipo-1	sensibles al contexto	$\alpha A \beta \rightarrow \alpha \beta \gamma$	linealmente acotado
-	indexado	indexado	de pila anidada
-	semisensibles al contexto	árbol	de pila incrustada
Tipo-2	libres del contexto	$A \rightarrow \alpha$	de pila
-	determinista libres del contexto	determinista libres del contexto	de pila no determinista
Tipo-3	regular	$A \rightarrow a$	finito
-	libre*	-	finito aperiódico

Tab. C.1: *Tabla que relaciona los tipos de gramáticas con los lenguajes que generan mediante el determinado autómata*

definir el lenguaje generado L por la gramática G como

$$L(G) = \{w \in \Sigma^*, \text{ tal que } S \Rightarrow^* w\}$$

siendo Σ^* la clausura de Kleene sobre un alfabeto Σ. Es decir que existe una secuencia de pasos de derivación tales que partiendo del símbolo inicial S se llega a la palabra w y por tanto ésta pertenece al lenguaje.

$$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow w$$

C.3. Lenguaje regular

Un lenguaje L es regular si y solo si cumple al menos alguna de las siguientes condiciones:

- L es finito.
- $L = M \cup N$, siendo N y M también lenguajes regulares, su unión es también un lenguaje regular.

- $L = MN$, siendo N y M también lenguajes regulares, su concatenación es también un lenguaje regular.
- $L = M^*$, siendo M un lenguaje regular, la clausura de Kleene es también un lenguaje regular.

Un lenguaje regular L basado en un alfabeto Σ es aquel que es generado por derivación de una gramática regular G a partir de un símbolo inicial $S \in \Sigma$

Un ejemplo muy conocido de lenguaje regular es el de los números binarios, cuyo alfabeto consta de dos símbolos $\Sigma = \{0, 1\}$ y cuyos elementos o palabras se realizan mediante concatenación de estos símbolos.

C.3.1. Expresión regular

La *expresión regular* es un lenguaje de especificación de búsqueda de cadenas de caracteres en un texto. Es una expresión algebraica para especificar clases simples de secuencias de caracteres alfanuméricos (letras, dígitos, espacios, tabuladores y signos de puntuación), o lo que es lo mismo, un *patrón*. Su definición ha resultado ser uno de los mayores estándares en ciencia computacional. Su uso se extiende por sistemas UNIX, WINDOWS y buscadores WEB. Comenzó a desarrollarse por Kleene en 1956, que probó la equivalencia entre la expresión regular y un autómata finito, dado que ambos son capaces de generar y reconocer el mismo lenguaje (el denominado lenguaje regular)

Para la presente tesis, las expresiones regulares juegan un papel muy importante porque van a ser los constituyentes de las gramáticas utilizadas para el reconocimiento automático del habla.

Existen distintas descripciones de una expresión regular según el entorno de uso. En este trabajo describimos sus elementos más importantes:

- **Símbolos terminales** como son los caracteres alfanuméricos. Ejemplo: `aprender` casaría con la cadena de caracteres "aprender"
- **Operador** `OR` "|" Selecciona una de los dos elementos que los separa. Ejemplo: `aprendo|aprende` casaría con las cadenas "aprendo" y "aprende"
- **Agrupación** mediante "(" y ")" Ejemplo: `aprend(o|e)` casaría con las cadenas "aprendo" y "aprende"
- **Disyunción.** Cadena opcional sin repetición "[" y "]" Ejemplo: `[0123456789]` casaría con cualquier dígito entre 0 y 9

- **Rango** de caracteres siguiendo un orden alfabético o según la codificación `Ascii` Ejemplo: `[0-9]` casaría con cualquier dígito entre 0 y 9
- **Repetición "$*$ Kleene"** de cero o más veces de la expresión anterior. Ejemplo: *[ab]** casaría con cualquier cadena que tenga cero o más "aes" o "bes"; `baa*` casaría con cadenas como "ba", "baaaaaaaaaaaaa"
- **Repetición "$+$ Kleene"** de una o más veces de la expresión anterior. Ejemplo: `ba+` casaría con cadenas como "baa" o "baaaaaaaaaaa"
- **Repetición "?"** de cero o una vez de la expresión anterior. Ejemplo: `baa?` casaría con las cadenas "ba" o "baa"
- **Repetición numerada entre llaves** Donde se especifica el número de veces que se repite una expresión. Ejemplo: `ba{3}` casaría con la cadena "baaa"
- **Sustitución** con el carácter especial "." Ejemplo: `am.` casaría con cadenas como "amo" "ama" "ame" etc.

La descripción anterior puede ser formulada siguiendo una nomenclatura algebraica:

$$ER = \Sigma \cup \{\text{“}\wedge\text{”}, \text{“}+\text{”}, \text{“}\bullet\text{”}, \text{“}*\text{”}, \text{“}(\text{”}, \text{“})\text{”}, \Phi\}$$

Donde,

- ER es el conjunto de expresiones regulares sobre el alfabeto Σ
- $\wedge \in ER$, representa la palabra vacía.
- $\Phi \in ER$, representa el conjunto vacío.
- Si $\sigma \in \Sigma \Rightarrow \sigma \in ER$, es decir, todo elemento del alfabeto es una expresión regular.
- Si $e_1 \in ER$ y $e_2 \in ER \Rightarrow w = e_1 + e_2 \in ER$ Por ejemplo: sea $w = uno|otro$ tal que $uno \in ER$ y $otro \in ER$, entonces $(uno|otro) \in ER$
- Si $e_1 \in ER$ y $e_2 \in ER \Rightarrow w = e_1 \bullet e_2 \in ER$ Por ejemplo: sea $w = entretantos$ tal que $entre \in ER$ y $tantos \in ER$, entonces $entretantos \in ER$
- Si $e \in ER \Rightarrow e^* \in ER$ Por ejemplo: sea $w = ay$ tal que $ay \in ER$, entonces $ayayayayay \in ER$

Por ejemplo La expresión regular asociada a la generación del "lenguaje baby" presentado en C.1 sería,

$$baa*!$$

Otras expresiones regulares interesantes:

```
$brazo[s] | cuello$
```

representa una de estas palabras: "brazo", "brazos" ó "cuello"

```
$hola|buenos días$
```

representa alguna de estas expresiones: "hola" ó "buenos días"

```
$si [al|cuando] (levan(es|ar))$
```

representa alguna de estas expresiones: "si al levantar", "si cuando levantes", "si al levantes", "si cuando levantar"

C.3.2. Gramática regular o de tipo 3

El término "regular" se debe a que las palabras que generan las gramáticas regulares suelen tener una serie de "repeticiones regulares" de algunos de sus símbolos. Una gramática regular se define mediante el cuádruplo (V, Σ, R, S) tal que:

- V corresponde con un alfabeto de no terminales o variables.
- Σ corresponde con un alfabeto de terminales o constantes.
- R es un conjunto de reglas tal que $R \subset V \times (\Sigma V \cup \Sigma)$
- $S \in V$ es el símbolo inicial.

La definición de la regla como $R \subset V \times (\Sigma V \cup \Sigma)$ viene a decir que la forma de una gramática regular es

$$A \to aB$$

siendo A y B símbolos no terminales y a un símbolo terminal.

Por ejemplo De este modo podemos definir "lenguaje baby"a partir de una gramática regular. Sea el alfabeto del lenguaje $\Sigma = \{b, a, !\}$, la gramática que genera el lenguaje será:

- $S \rightarrow !X$
- $X \rightarrow aA$
- $A \rightarrow aA$
- $A \rightarrow b$

C.3.3. Autómata finito o máquina de estados finitos

Es la máquina de estados más sencilla que existe y sobre la cual se basan las definiciones del resto de máquinas de estados (máquina de Turing, autómata con pila, etc) Su importancia, para el presente trabajo es su fuerte vinculación con las expresiones regulares, con las gramáticas regulares (o de tipo 3, según la jerarquía de Chomsky) y, por tanto, vinculación con los lenguajes regulares.

Definición formal de un autómata finito Formalmente se define autómata finito o máquina de estado finitos M como un quíntuplo $(Q, \Sigma, q_0, F, \delta)$ tales que:

- Q es un conjunto de N estados o también denominados símbolos $q_0, q_1, ..., q_N$
- Σ es el alfabeto de símbolos de entrada.
- q_0 es el estado inicial.
- F es el conjunto de estados finales, tal que, $F \subseteq Q$
- $\delta : Q \times \Sigma \longrightarrow Q$ es la función de transición entre estados. Dado un estado $q \in Q$ y un símbolo de entrada $s \in \Sigma$ entonces $\sigma(q, s)$ devuelve un nuevo estado $q' \in Q$

Notación gráfica y en tabla de transición de estados La formulación matemática de un autómata finito tiene su correspondiente *notación gráfica*. Cada estado se representa mediante un círculo con una etiqueta (el nombre del estado) y cada transición mediante una flecha que une dos estados y una etiqueta correspondiente al símbolo de entrada en la máquina de estados

finitos. Un ejemplo de máquina de estados finitos podemos encontrarlo en la figura C.2

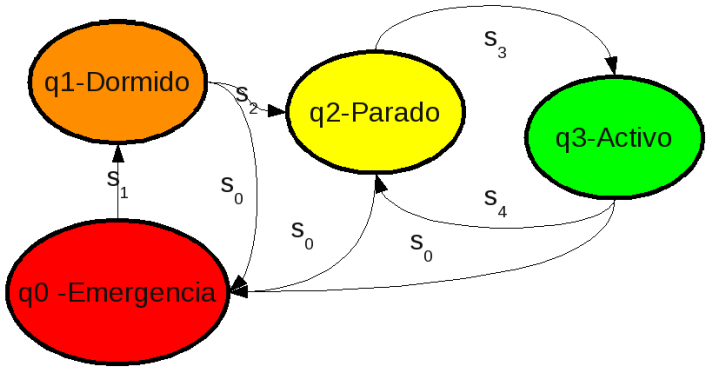


Fig. C.2: *Ejemplo de máquina de estados finitos*

Asimismo, una FSM puede representarse mediante una *tabla de transición de estados*. Como en la notación gráfica, la tabla de transición de estados representa el estado inicial, el resto de estados y qué transición parte de un estado y llega a otro según el símbolo de entrada a la máquina. En la figura C.2 puede verse la representación en tabla de transición de estados del diagrama descrito en la figura C.2 El valor 0 indica que la máquina permanece en el mismo estado cuando llega el correspondiente símbolo.

	Entrada				
Estado	s_0	s_1	s_2	s_3	s_4
q_0	0	q_1	0	0	0
q_1	q_0	0	q_2	0	0
q_2	q_0	0	0	q_3	0
q_3	q_0	0	0	0	q_2

Tab. C.2: *Ejemplo de tabla de transición de estados*

Aplicaciones de un autómata finito Los estados representan las situaciones elementales por las que pasa un proceso y en los que permanece un lapso de tiempo considerable, mientras que los eventos que representan las transiciones entre estados idealmente son instantáneos. En una FSM los estados son excluyentes entre sí, de modo que el sistema no puede encontrarse

en dos o más estados distintos simultáneamente. Dado un estado origen, la función δ está bien definida para cada símbolo de entrada, de modo que el estado destino está bien determinado lo que hace de la máquina de estados finitos un *sistema determinista*.

Las máquinas de estados finitos fueron introducidas por Turing desde 1936 [Turing, 1950] y, para muchos, representan el comienzo de la *ciencia computacional moderna*. La *máquina de Turing* es un modelo avanzado de máquina de estados finitos. En un solo movimiento es capaz de leer un símbolo en una cinta, escribir un símbolo distinto en la cinta, cambiar de estado y avanzar o retroceder otro símbolo.

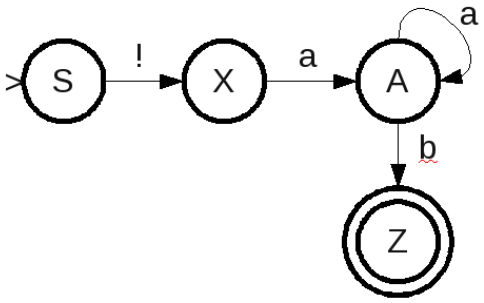


Fig. C.3: *Máquina de estados finitos asociada al lenguaje baby*

Paso de un autómata finito a una gramática regular Para ver la equivalencia entre ambos conceptos, autómata y gramática, vamos a analizar en el caso de los lenguajes regulares, cómo se crearía un autómata finito a partir de la definición de una gramática regular y viceversa.

Para construir una gramática regular a partir de una autómata finito, basta con crear las reglas de la gramática regular a partir de cada transición del autómata finito, del siguiente modo: dada la transición de un autómata finito

$$\delta = (p, \sigma), q$$

tal que $p, q \in K$ son elementos del conjunto de estados del autómata, y $\sigma \in \Sigma$ es el símbolo de entrada al autómata, se construye la regla de la gramática formal de la siguiente forma:

$$X_p \rightarrow \sigma X_q$$

y viceversa. Es decir, que los símbolos no terminales de la gramática pasan a ser estados del autómata, y los símbolos no terminales pasan a ser transiciones.

Así por ejemplo, tomando la gramática del lenguaje baby descrita en la sección C.3.2 podemos construir la máquina de estados finitos descrita en la figura C.3.

C.4. Lenguaje libre del contexto

C.4.1. Introducción a los lenguajes libres del contexto

Un lenguaje se dice que es *libre de contexto* cuando es generado por una gramática libre de contexto. La expresión libre de contexto se entiende analizando otro tipo de gramáticas denominadas *sensibles al contexto*. Éstas son de la forma $\alpha A\beta \rightarrow \alpha\gamma\beta$ siendo α, β y γ cadenas de símbolos terminales y/o no terminales y A un símbolo no terminal. Es decir, que A solo puede generar γ cuando ambos están "rodeados" de las cadenas de símbolos α y β que es lo que se lingüísticamente se denomina *contexto*

Chomsky también define a estas gramáticas como *gramáticas de la estructura de la frase*, dado que gran parte de los idiomas humanos tienen una sintáxis que puede ser modelada con gramáticas libres de contexto. Los trabajos desarrollados en [Chomsky, 1956] y [Chomsky, 1957] comienzan lo que más tarde ha venido a denominarse Procesamiento del Lenguaje Natural (NLP) y que se basa en modelar el lenguaje natural como un lenguaje formal. Puede demostrarse que la sintáxis del idioma español no puede ser formulada en términos de una gramática regular y por eso la necesidad de una nueva gramática formal.

Uno de los aspectos que dan importancia a los lenguajes libres de contexto es el hecho de que a partir de lenguajes de programación como "Algol" (`ALGOL-58` y `ALGOL-60`) o "`Pascal`" la mayoría de los lenguajes de programación actuales se basan en gramáticas libres de contexto. El formalismo de estas gramáticas es equivalente al denominado *Backus-Naur Form* (`BNF`) desarrollado primeramente por John Backus y más tarde por Peter Naur, y que evolucionó hasta convertirse en el estándar de la IOS (International Organization for Standardization) *ISO(IEC 14977:199(E)* en el formalismo *Extended Backus-Naur Form* (`EBNF`). A su vez, el W3C (World Wide Web Consortium) define el estándar `XML` (eXtenden Markup Language) [1] como

[1] http://www.w3.org/XML/

una implementación de EBNF.Este último estándar es la base de otros muchos desarrollados por la W3C, que han sido utilizados ampliamente en la implementación de los sistemas que se describen en esta tesis, como son, voiceXML (voice eXtended Markup Language) [2] SGSR (Speech Recognition Grammar Specification) [3] , SISR (Semantic Interpretation for Speech Recognition) [4] y NLSML (Natural Language Semantics Markup Language) [5] como se verá más adelante.

Cabe destacar, además, que en la implementación de gramáticas libres de contexto para el reconocimiento de voz utilizado en el presente trabajo se utiliza el estándar ABNF (Augmented Backus-Naur Form) que también deriva del formalismo BNF Es definido como el estándar *RFC 5234* por *Internet Standard 68* (STD 68) ABNF describe un lenguaje formal concebido como un protocolo de comunicación bidireccional

C.4.2. Gramática libre del contexto o de tipo 2

Formalmente una gramática libre de contexto (GLC) o de tipo-2 en la jerarquía de Chomsky (ver tabla C.1) se define como una cuádrupla que consta de los siguientes elementos:

- V, conjunto de símbolos no terminales o alfabeto.
- Σ, conjunto de símbolos terminales, tal que $N \cap \Sigma = \Phi$, siendo Φ el conjunto vacío. Es decir que los conjuntos de terminales y no terminales son disjuntos.
- R conjunto finito de reglas tales que $R \subset V \times (V \cup \Sigma)^*$ Es decir reglas de la forma $A \rightarrow \alpha$ siendo A un no terminal y α una cadena de símbolos terminales y/o no terminales.

Un ejemplo de lenguaje basado en una gramática libre de contexto es el conjunto de palabras $\{a^n b^n\}$, siendo n un número natural. Este lenguaje no es regular, pero sí puede ser descrito por reglas de gramáticas libres de contexto:

1. $S \rightarrow aSb$
2. $S \rightarrow ab$

[2] http://www.w3.org/TR/voicexml20/
[3] http://www.w3.org/TR/speech-grammar/
[4] http://www.w3.org/TR/semantic-interpretation/
[5] http://www.w3.org/TR/nl-spec/

Así aplicando la primera regla a la segunda sucesivas veces obtendríamos

$$S \rightarrow aabb \Rightarrow aaabbb \Rightarrow aaaabbbb \Rightarrow ...$$

es decir el lenguaje $L = \{ab, aabb, aaabbb, aaaabbbb, ...\}$

Otro ejemplo más cercano al procesamiento de lenguaje natural es el de la descripción de la sintáxis del español en sus sintagmas y partículas sintagmáticas. Así, por ejemplo, podemos modelar un lenguaje subconjunto del lenguaje español, mediante una gramática que tenga reglas del tipo:

- $\$frase \rightarrow \$sintagmaNominal\ \$sintagmaVerbal$
- $\$sintagmaNominal \rightarrow \$sustantivo$
- $\$sintagmaNominal \rightarrow \$articulo\ \$sustantivo$
- $\$sintagmaVerbal \rightarrow \$verbo$
- $\$sintagmaVerbal \rightarrow \$sintagmaVerbal\ \$complementoDirecto$
- $\$articulo \rightarrow$ "*el*"|"*la*"
- $\$sustantivo \rightarrow$ "*robot*"|"*persona*"
- $\$verbo \rightarrow$ "*anda*"|"*levanta*"|"*gira*"|"*interactua*"
- $\$complementoDirecto \rightarrow$ "*el brazo*" ["*izquierdo*"|"*derecho*"]

Obsérvese que en la construcción de las reglas se ha optado por utilizar el símbolo $ precediendo a los símbolos no terminales y que los símbolos terminales se ponen entrecomillados. Como se explicará más adelante, esta nomenclatura corresponde a la utilizada en la definición de las gramáticas para el reconocedor automático del habla, siguiendo el estándar `ABNF` (Augmented Backus-Naur Form) Obsérvese también que en las reglas se han utilizado expresiones regulares (ver C.3), haciendo uso de los símbolos "|", "[" y "]"

De este modo el lenguaje constaría de cadenas de caracteres como "el robot levanta el brazo izquierdo", "la persona gira" que tienen sentido *semántico*, pero también cadenas como "la robot interactua el brazo" que no tienen sentido semántico. Esto plantea un problema general de no fácil solución de hasta qué punto un lenguaje formal puede representar un lenguaje que se quiere modelar. Asímismo, la construcción de una gramática, que establece la corrección sintáctica de un lenguaje, debe tener en cuenta, además, los aspectos semánticos de las partículas que se manejan en la gramática.

Para comprobar que el diseño de una gramática libre de contexto modela correctamente un determinado lenguaje, se cuenta con dos pruebas generales que toda gramática bien diseñada debe cumplir: prueba de *corrección* y prueba de *completitud*, que se explican a continuación. Dado un lenguaje L que quiere ser modelado por una gramática libre de contexto G tal que genera el lenguaje $Ł(G)$ para un correcto modelado se han de cumplir las siguientes dos reglas:

- $Ł(G) \subseteq L$ o prueba de corrección.
- $L \subseteq Ł(G)$ o prueba de completitud.

La primera asegura que el lenguaje generado, $Ł(G)$, no contiene palabras fuera del lenguaje que se quiere modelar L. La segunda asegura que la gramática es capaz de generar al menos todas las palabras de L

La equivalencia de dos gramáticas se define según los lenguajes que crean y puede ser una equivalencia *fuerte* o una equivalencia *débil.* Así, se dice que dos gramáticas tienen una equivalencia débil si generan el mismo conjunto de palabras, pero no asignan la misma estructura de frases para cada sintagma. Si además de generar el mismo conjunto de palabras las dos gramáticas establecen la misma estructura de frases para cada sintagma, entonces se dice que las dos gramáticas tienen una equivalencia fuerte.

C.4.3. Autómata con pila

Si se analiza qué incorpora de nuevo la gramática libre de contexto respecto a la gramática regular, vemos que aquella incorpora en la forma de sus reglas la posibilidad de tener símbolos no terminales en ambos miembros de la regla. Ésta sutileza implica que puedan existir reglas recursivas, esto es, reglas que incorporen el mismo no terminal en ambos miembros. Un ejemplo de una regla recursiva se ha visto al describir la gramática libre de contexto del lenguaje identificado por el conjunto $\{a^n b^n\}$ en la regla $S \rightarrow aSb$ Ésta recursividad, para ser computada, implica la existencia de algún tipo de "memoria de símbolos"

El autómata con pila es un autómata finito que incorpora una pila (LIFO) de símbolos como mecanismo de memoria. Así, en cada transición, no solo se identifica el símbolo de entrada al autómata, sino también, el símbolo que se extrae de la pila y el que se introduce en la pila en dicha transición, si los hubiere.

Definición formal del autómata con pila Formalmente un autómata con pila se describe con el séxtuplo $(K, \Sigma, \Gamma, s, F, \Delta)$ tal que:

- K, es el conjunto de estados del autómata.
- Σ, es el alfabeto de entrada al autómata.
- Γ, es el alfabeto que maneja la pila. Puede o no coincidir con el alfabeto de entrada al autómata. Por ejemplo, si el autómata utiliza símbolos de separación en la pila.
- $S \in K$, es el símbolo inicial.
- $F \subseteq K$, conjunto de estados finales.
- $\Delta \subseteq (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$ conjunto finito de transiciones.

En el procesamiento, el autómata comienza con la pila vacía. Se dice que un autómata con pila ha aceptado una palabra dentro del lenguaje que genera si y solo si, todos los símbolos de la palabra han sido procesados por el autómata, se ha llegado a uno de los estados finales y la pila del autómata está vacía.

Notación gráfica y traza de ejecución Gráficamente un autómata de pila se representa prácticamente igual que un autómata finito. La única variación es en las transiciones. Mientras en el autómata finito cada transición se denotaba por el símbolo terminal de entrada al autómata, ahora, en el autómata de pila tenemos dos símbolos más, el símbolo que se obtiene de la pila y el que se introduce en la pila. En caso de que no hubiera transacción de símbolos con la pila se utiliza el símbolo vacío ϵ

Análogamente a la tabla de transición de estados, el autómata de pila cuenta, además con una *tabla de traza de ejecución* que representa el estado del autómata en cada paso, para una cadena de símbolos no terminales de entrada dada.

Por ejemplo Para el lenguaje de contexto libre $\{a^n b^n\}$ la notación gráfica del autómata de pila quedaría como en la figura C.4

Y podemos realizar la tabla de traza de ejecución para la entrada de la cadena *aaabbb* para el autómata de la figura C.4

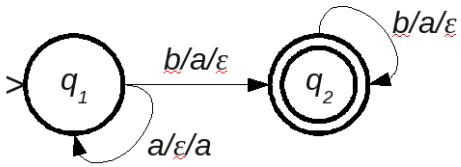


Fig. C.4: *Autómata de pila para el lenguaje* $a^n b^n$

Estado	Entrada	Pila
q_1	a	ϵ
q_1	a	a
q_1	a	aa
q_1	b	aaa
q_2	b	aa
q_2	b	a
q_2	ϵ	ϵ

Tab. C.3: *Ejemplo de tabla de traza de ejecución para la entrada aaabbb*

D. INTRODUCCIÓN AL ESTÁNDAR VOICEXML

D.1. Introducción

De entre los distintos paradigmas de sistemas de manejo de diálogo, `voiceXML` representa una visión del diálogo basada en "huecos de información" El sistema de diálogo realiza sus "movimientos interactivos" para conseguir, dialogando con el usuario, ir rellenando estos huecos que vienen establecidos *a priori* en el diseño del diálogo. Tanto el usuario como el sistema pueden tomar la iniciativa de expresarse en cualquier momento a lo largo del proceso del diálogo (iniciativa mixta) Asimismo, el usuario, mediante un solo enunciado, puede rellenar varios de estos huecos de información.

En esta sección se explican las características más relevantes para el presente trabajo de lo que es un sistema de diálogo que esté basado en el estándar de lenguaje de etiquetas `voiceXML-2.0`. Se explica tanto su estructura general como el significado de cada una de las etiquetas y el algoritmo que las interpreta. No se realiza una explicación exhaustiva del estándar y el lector deseoso de profundizar más en este lenguaje puede encontrar diversa literatura relacionada como por ejemplo: definición de los creadores del estándar[1]; [McTear, 2004], donde se dan ejemplos muy ilustrativos; [Sharma and Kunins, 2002], donde se proponen algunos consejos prácticos de diseño.

D.2. Estructura y ejecución de un diálogo

Un sistema de diálogo basado en `voiceXML` funciona como una máquina de estados finitos *conversacional*. Cada estado de este autómata se denomina *diálogo* de modo que cada diálogo puede tener un sucesor. Es importante no confundir este concepto de diálogo con el concepto lingüístico o habitual.

[1] http://www.w3.org/TR/voicexml20/

En lo sucesivo habrá de entenderse por "diálogo" como un estado dentro del autómata "sistema de diálogo".

El proceso de diálogo, que se denomina *sesión*, solamente puede encontrarse en uno solo de estos estados o diálogos. La sesión termina cuando un diálogo así lo explicita o cuando no existe ningún diálogo sucesor. Si la sesión está bien implementada, también debe terminar en el momento en que el usuario así lo especifique. En cada diálogo[2] se realiza un intercambio verbal entre el usuario y la aplicación hasta que ésta obtiene cierta cantidad de información predefinida.

Por otro lado, la implementación de una sesión típicamente se realiza en un conjunto de documentos. Cada documento puede albergar uno o varios diálogos. Existen dos tipos de documentos, uno principal único y obligatorio, que define lo que se denomina *aplicación*; y uno o varios secundarios, que son optativos y que están relacionados con una única *aplicación*. El documento principal de una aplicación siempre está cargado en el intérprete de `voiceXML` y sus variables y scripts son visibles al resto de documentos secundarios. En una sesión puede pasarse por varias aplicaciones a lo largo de las distintas transiciones entre diálogos. Si ocurre una transición entre diálogos de distintas aplicaciones ocurre una transición de aplicación. Ello consiste en la descarga de todos los documentos cargados de una aplicación y la carga de, al menos, el documento principal de la nueva aplicación.

Dentro de un mismo documento, salvo que un diálogo haga mención explícita de realizar una transición a otro, la ejecución de los distintos diálogos en un mismo documento se realiza siguiendo la secuencia en la que aparecen en el documento. Asimismo, cuando se realiza una transición de un diálogo a un documento, sin especificar a qué diálogo dentro del documento, se toma por defecto el primer diálogo implementado en el documento receptor de la transición.

Cada transición explícita de un diálogo a otro se realiza especificando el nombre del diálogo receptor mediante una dirección `URI`. Esta dirección puede contener dos partes: la dirección del documento en sí y el denominado "fragmento" de la `URI`, que especifica el nombre del diálogo receptor, y es opcional. Estas partes vienen separadas por el símbolo `#`. Además, la dirección del documento receptor de la transición puede estar expresada de modo absoluto o de modo relativo. En el primer caso, si el documento es un fichero local, la dirección `URI` coincide con la ruta absoluta del fichero dentro de la máquina local; si el documento está en un servidor la dirección `URI`

[2]Se insiste, diálogo entendido como estado dentro del autómata conversacional.

coincide con la URL de dicho documento. En caso de que la URI se exprese de modo relativo, se toma como base de la dirección, la ruta (tanto local como remota, según corresponda) del documento raíz de la aplicación en curso. Es decir, que la localización del documento principal a una aplicación, tanto si reside en un servidor como si es local, establece una dirección raíz a partir de la cual se buscan los otros documentos donde residen los diálogos sucesivos. Por ejemplo, imaginemos las siguientes líneas en un documento raíz de una aplicación llamado (incluyendo la ruta absoluta, en este caso local) /home/usuario/dialogos/comienzo.vxml

```
<goto next = http://servidor.com/siguiente.vxml />
<goto next = http://servidor.com/siguiente.vxml#dialogo4 />
<goto next = file:///home/usuario/siguiente.vxml#dialogo4 />
<goto next = siguiente.vxml#dialogo4 />
```

Todas establecen una transición a otro diálogo mediante la etiqueta <goto> que se explicará con mayor detalle en la sección D.8.

En la primera línea, se establece la dirección absoluta URI del documento receptor, en este caso, documento remoto. En esta dirección no se especifica el *fragmento* de la URI, luego la transición se realizará al primer diálogo dentro del documento siguiente.vxml.

En la segunda línea, sí se establece dicho fragmento, especificando así a qué diálogo se realiza la transición, dentro de los diálogos implementados en el documento siguiente.vxml.

En la tercera línea, se realiza una transición a un diálogo concreto de un documento local. Éste se especifica mediante su dirección absoluta.

Por último, en la cuarta línea se realiza una transición a un diálogo concreto de un documento local, pero se especifica mediante su dirección relativa, y se toma como base de dicha dirección la ruta del documento origen, que es /home/usuario/dialogos/

En resumen, un sistema de diálogo basado en voiceXML está estructurado en cinco ámbitos jerárquicos ordenados de mayor a menor amplitud:

1. **Sesión.** Proceso desde que comienza el primer diálogo hasta que termina el último diálogo. Una sesión termina si un diálogo así lo especifica o, simplemente, no realiza ninguna transición a otro diálogo, o si lo especifica el usuario.

2. **Aplicación.** Una sesión cuenta con una o varias aplicaciones. Cada una consta de un documento principal y, opcionalmente, de uno o varios documentos secundarios.

3. **Documento.** En cada documento se implementan uno o varios diálogos. La extensión típica utilizada para un documento de `voiceXML` es `.vxml`

4. **Diálogo.** Dentro de un documento puede implementarse uno o más diálogos que pertenecen, por tanto, a la misma aplicación.

5. **Anónimo.** Dentro de un diálogo existen ámbitos locales específicos que se explicarán más adelante.

La ejecución de un diálogo se realiza, por un lado, interpretando las etiquetas contenidas en los documentos que lo implementan y convirtiendo lo interpretado a código interno en lenguaje `ECMAScript`. Por otro lado un algoritmo principal se encarga del control de ejecución de cada diálogo. El algoritmo se denomina `FIA` (Form Interpretation Algorithm) cuyo funcionamiento será explicado más adelante.

Cada uno de estos ámbitos jerárquicos presenta un espacio de visibilidad y vida de las variables y "scripts" que maneja la sesión. De modo que las variables o scripts declarados en un ámbito son visibles y utilizables en los ámbitos subordinados.

Cada documento `.vxml` consta de una serie de etiquetas. Cada una, además, admite una serie de *atributos*. Existen dos tipos de etiquetas, las que albergan a otras y las que no. De modo que el formato de implementación es

```
<etiqueta atrb1 = ‘‘valor1’’ atrb2 = ‘‘valor2’’ ...
atrbN = ‘‘valorN’’/>
```

para las etiquetas de una sola línea y

```
<etiqueta1 atrb1 = ‘‘valor1’’ atrb2 = ‘‘valor2’’ ...
atrbN = ‘‘valorN’’>
...
  <etiqueta2 ...>
  ...
  </etiqueta2>
...
</etiqueta1>
```

para etiquetas que albergan a otras. Las etiquetas "albergadas" se denominan *hijos* de la etiqueta que los alberga.

Un documento `voiceXML` está compuesto por etiquetas de "alto nivel" que especifican cada uno de los diálogos. Éstos se especifican mediante etiquetas `<form>` y `<menu>` que serán explicadas más adelante.

La herramienta utilizada para interpretar los distintos documentos `voiceXML` se denomina *plataforma*.

D.3. Entrada del usuario. Gramáticas literales.

La entrada del usuario es verbal, por tanto, para captar la información que el usuario expresa, es necesario un reconocedor automático del habla. El estándar `voiceXML` está diseñado para incorporar tal reconocedor, pero no se dan instrucciones concretas de como realizar la integración. El estándar solamente menciona que, mediante la etiqueta `<grammar>` admite gramáticas literales y semánticas implementadas en `SRGS` y `SISR` [3], por tanto la plataforma concreta que realice la interpretación de `voiceXML` también debe ser capaz de realizar una interpretación de la gramática escrita en estos estándares junto con la entrada del usuario. Sin embargo, esto se contradice con el hecho de que *es la herramienta de reconocimiento automático del habla quien ha de realizar la interpretación del habla mediante gramáticas literales y/o semánticas*. La resolución de este problema de integración depende, por tanto, de las plataformas específicas que se utilicen tanto para el reconocimiento del habla como para la interpretación de `voiceXML`.

La gramática especificada mediante la etiqueta `<grammar>` establece una gramática literal de contexto libre o una gramática semántica, ambas basadas en *reglas*. Estas reglas no son sino símbolos no terminales, pudiendo existir una regla principal que se establece mediante el atributo *root*.

Cada regla se define mediante la etiqueta `<rule>` especificando su nombre mediante el atributo `id`. Dentro de cada regla puede haber una expresión formal de símbolos terminales y no terminales, pues el estándar `SRGS` define una gramática de contexto libre, pero la manera de implementar tal expresión regular es mediante los siguientes elementos:

- `<one-of>` Equivale al operando `OR` de una expresión regular.
- `<item>` Especifica una secuencia obligatoria de símbolos terminales o no terminales dentro de la regla.

[3] La definición del estándar `voiceXML-2.0` establece como no obligatorio la interpretación de gramáticas implementadas en el estándar `ABNF`

- `<rule-ref/>` Se utiliza para hacer referencia a un símbolo no terminal, es decir, una regla dentro de la misma gramática o de otra.

Por ejemplo La gramática especificada a continuación consta de una regla principal "`command`" que consta de la secuencia de dos reglas obligatorias: "`action`" seguida de "`object`" A su vez, cada una de estas reglas tiene una serie de símbolos literales opcionales.

```
<grammar mode="voice" xml:lang="es-ES" version="1.0" root="command">
 <rule id="command" scope="public">
   <ruleref uri="#action"/> <ruleref uri="#object"/>
 </rule>

 <rule id="action">
   <one-of>
     <item> abrir </item>
     <item> cerrar </item>
     <item> borrar </item>
     <item> mover </item>
   </one-of>
 </rule>

 <rule id="object">
  <item repeat="0-1">
     <one-of> <item> el </item> <item> una </item> </one-of>
   </item>
   <one-of>
     <item> ventana </item>
     <item> fichero </item>
     <item> menu </item>
   </one-of>
 </rule>
</grammar>
```

La gramática reconocería enunciados como "abrir ventana", "cerrar una ventana", "mover el menu", etc. Nótese que en la regla "`object`" aparece el atributo `repeat`. Este atributo establece el número de repeticiones que pueden ocurrir de un símbolo en un mismo enunciado del usuario. Es análogo a los "operandos" $*$, $+$ y $\{\ \}$ de una expresión regular (véase sección C.3)

La gramática puede ser implementada en *línea*, como en el ejemplo anterior, o de forma *externa*. Las gramáticas externas se implementan en un documento a parte del documento `voiceXML` desde el cual son referenciadas mediante el atributo `src` dentro de la etiqueta `<grammar>` La referencia es una dirección `URI` y su significado y funcionamiento es análogo al significado y funcionamiento de las direcciones de documentos y diálogos en las transiciones, tal y como se ha visto anteriormente. Lo único que cambia es que, ahora, el contenido del fragmento de la `URI` (la parte de la `URI` precedida por el símbolo `#`) especifica una de las reglas de la gramática. Y esta es la manera en que una regla de una gramática puede ser mencionada por otra gramática.

D.4. Salida de voz sintetizada

La salida del sistema de diálogo consta, principalmente, de una frase que se sintetiza en voz mediante un sintetizador de texto a voz externo al manejador del diálogo. En `voiceXML` estas salidas se especifican mediante la etiqueta `<prompt>` Este elemento admite unos atributos opcionales. Los principales para el presente trabajo son:

- `bargein`, permite la interrupción de la síntesis de voz si el sistema detecta que el usuario ha comenzado a hablar.
- `cond`, es una expresión que representa una condición booleana. Debe ser verdadera para que se produzca la síntesis de voz.
- `count`. Si el algoritmo de control del diálogo (`FIA`) visita varias veces esta etiqueta, este atributo permite cambiar el texto que se sintetiza en cada visita.
- `timeout`, tiempo de espera máximo (en milisegundos) que el sistema espera a una entrada del usuario después de haber sintetizado el texto de esta etiqueta.
- `xml:lang`, idioma de habla en el sistema. Si este atributo no se especifica se hereda del atributo de igual nombre en la etiqueta principal `<vxml>`

Por ejemplo El siguiente código envía al sintetizador de voz las frases “no te he entendido” y “repítelo otra vez” con la opción de `bargein` con lo que

las frases son sintetizadas sin interrupción, salvo que en mitad del habla el usuario comience él mismo a hablar.

```
<prompt bargein = ‘‘true’’>no te he entendido</prompt>
<prompt bargein = ‘‘true’’>repítelo otra vez</prompt>
```

Se verán más ejemplos más adelante.

D.5. Construcción del diálogo: formularios y menú

La etiqueta principal de un documento de `voiceXML` es `<vxml>` Este elemento admite varios atributos como el "namespace" para `voiceXML`, el idioma utilizado, etc. Los documentos secundarios incluyen un atributo `application` cuyo valor establece el nombre del documento principal de la aplicación a la que pertenecen.

La etiqueta `<vxml>` puede albergar varias instancias de diálogos de dos tipos: formularios y/o menúes.

D.5.1. Formulario y algoritmo de interpretación `FIA`

Un formulario implementa un diálogo mediante la etiqueta `<form>`. Un formulario viene especificado por su nombre mediante el atributo `id`.

Tal y como se observa en la figura D.1, el formulario establece ciertos "huecos de información" o *campos*. Estos están definidos por una serie de etiquetas `<field>`. La etiqueta `<field>` tiene un nombre que viene especificado por el atributo `name`. Cada uno de estos campos crea una variable interna que por defecto está "vacía", o en estado *false*. Esta variable creada se denomina *variable sombra* y corresponde con una variable en `ECMAScript` visible en el ámbito del diálogo. Mediante el atributo `expr` puede inicializarse su valor. La variable es un tipo de objeto especial que tiene los siguientes propiedades:

- *campo*$.*utterance*, que alberga la serie de palabras obtenidas por el reconocedor automático del habla.
- *campo*$.*inputmode*, indica si el tipo de modo utilizado por el usuario es `voice` (voz) o `dtmf` (dígitos de teléfono)
- *campo*$.*interpretation*, objeto que alberga la interpretación semántica dada por el reconocedor automático del habla.
- *campo*$.*confidence*, grado de confianza del enunciado entero, dado por el reconocedor automático del habla.

donde *campo* es el nombre del campo colocado en el atributo `id` de la etiqueta `<field>`.

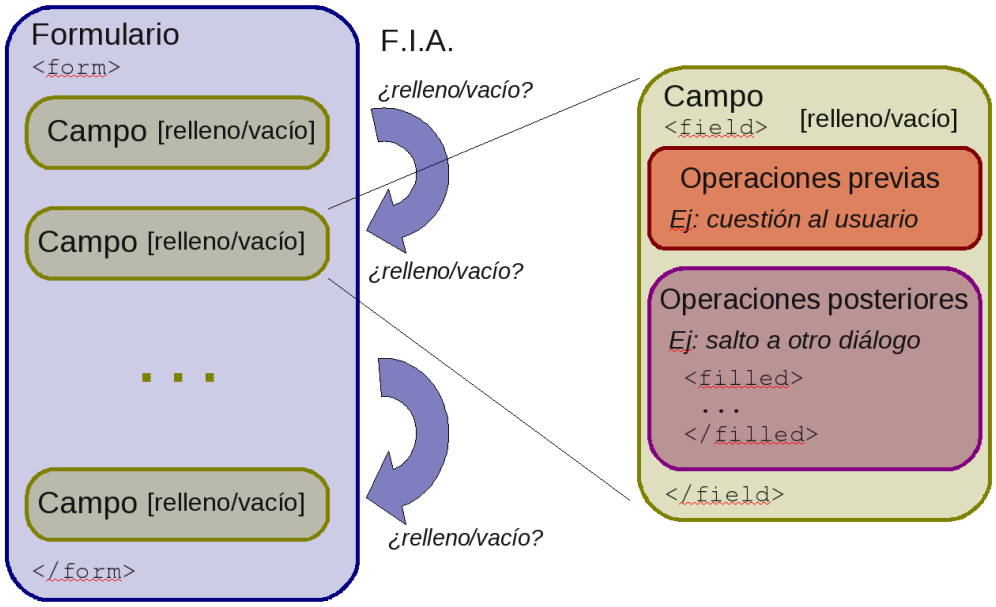


Fig. D.1: *Esquema de cómo el FIA recorre los campos de un formulario*

Existe una variable interna que se completa cada vez que ha habido reconocimiento de habla con una gramática cargada: *application.lastresult*$ Es un objeto `ECMAScript` que contiene las mismas propiedades que la variable *campo* enunciada más arriba: *utterance*, *inputmode*, *interpretation* y *confidence* Esta variable, en realidad es un vector de dichos objetos. Cada una de sus componentes contiene uno de los `n-best` (los n mejores resultados del reconocedor de habla) siendo el primero de estos objetos (*application.lastresult*$) el mejor resultado del reconocimiento.

El algoritmo de interpretación del formulario (`FIA`) recorre en bucle cada uno de los campos del formulario. En los campos vacíos ejecuta unas *operaciones previas* (ver figura D.1) que, típicamente suele ser una solicitud al usuario para que, mediante voz, complete la información asociada al campo. Puede establecerse una condición de entrada al campo mediante el atributo `cond` de modo que el `FIA` entra en el campo cuando la condición fuere verdadera.

Una vez completado un campo, el intérprete pasa a ejecutar el contenido que viene definido dentro de la etiqueta `<filled>` En la figura se ha repre-

sentado mediante las *operaciones posteriores*. En esta parte del documento es donde se interpreta la información recibida para saltar a otro diálogo, autocompletar otros campos vacíos, vaciar campos ya completados, definir objetos en `ECMAScript`, ..., es decir, para realizar la interpretación de la información completada.

En un formulario puede establecerse una única gramática para todos los campos, o bien, cada campo puede especificar su propia gramática.

La salida por voz puede colocarse tanto al comienzo del formulario como en cada uno de sus campos.

Por ejemplo En el siguiente código se muestra un simple diálogo en el que el sistema pregunta por su nombre al usuario.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" version="2.0">

  <form id = "main">
    <field name = "nombre">

      <grammar root = "r" mode = "voice" version = "1.0">
        <rule id = "r">
          <one-of>
            <item>Miguel Ángel</item>
            <item>Ramón</item>
            <item>Javi</item>
            <item>Sini</item>
          </one-of>
        </rule>
      </grammar>

      <prompt count="1">Cómo te llamas?</prompt>
      <prompt>Simplemente quiero saber tu nombre</prompt>

      <filled>
        <prompt>Vale, te llamas <value expr = "nombre"/></prompt>
      </filled>
```

```
    </field>
  </form>
</vxml>
```

En el ejemplo se hace uso de una etiqueta cuya explicación no se ha realizado todavía: la etiqueta `<value/>` Su significado resulta fácil de entender por el contexto. La etiqueta `<value/>` hace referencia al valor de una de las variables visibles en el sistema. El atributo `expr` de esta etiqueta denota la variable que se quiere mencionar. Como se ha explicado anteriormente, cada campo dentro de un formulario está asociado a una variable `ECMAScript` que es visible dentro del diálogo, por eso en el ejemplo se menciona el nombre del campo `nombre`.

Obsérvese también que se ha hecho uso del atributo `count` dentro de la etiqueta `<prompt>` De este modo el control del diálogo entrará en el primer prompt la primera vez, entrando en el segundo en sucesivas veces, si las hubiere.

Un ejemplo de un caso posible de diálogo:

SISTEMA: Cómo te llamas?
USUARIO: Y a tí que te importa!
[*El* SISTEMA *no entiende el enunciado del usuario y pasa al siguiente* `<prompt>` *hasta rellenar el campo* `nombre`]
SISTEMA: Simplemente quiero saber tu nombre
USUARIO: Sini
[*El* SISTEMA *entiende el enunciado del usuario y pasa al correspondiente etiqueta* `<filled>`]
SISTEMA: Vale, te llamas Sini

D.5.2. Completado de varios campos con un solo enunciado

La etiqueta `<initial>` permite que un formulario sea de *iniciativa mixta.* Así, el usuario puede tomar la iniciativa y completar a la vez varios campos del formulario con un solo enunciado, previamente a que el `FIA` comience a visitar cada uno de los campos no completados, y el sistema comience a realizar preguntas para cada campo. La etiqueta `<initial>` es hija de la etiqueta `<form>` y se coloca antes del primer campo del formulario. Funciona de un modo muy parecido a un campo `<field>`, salvo que no admite una gramática específica y cuando se utiliza debe definirse una gramática general a nivel formulario; tampoco admite una etiqueta específica `<filled>`.

Por ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<form id = "hours">
   <grammar src = "hours.grxml" type = "application/srgs+xml"/>
   <block>
       Hola. Vamos a programar mi horario de funcionamiento.
   </block>
   <initial name = "hour">
     <prompt>
       Dime a qué hora quieres que me encienda y
a qué hora quieres que me apague.
     </prompt>
   </initial>
   <field name = "on_hour">
     <prompt>Dime la hora de encendido</prompt>
   </field>
   <filled>
     <prompt>Me enciendo a las <value expr = "on_hour" /></prompt>
   </filled>
   <field name = "off_hour">
     <prompt>Dime la hora de apagado</prompt>
   </field>
   <filled>
     <prompt>Me apago a las <value expr = "off_hour" /></prompt>
   </filled>
 </form>
</vxml>
```

Se da por supuesto que el fichero externo de gramáticas `hours.grxml` implementa la posibilidad, por parte del usuario, de decir cualquier hora dentro del rango factible.

La etiqueta `<block>` sirve para contener código ejecutable no interactivo, por ejemplo frases para ser sintetizadas a voz fuera del campo de un formulario.

En el ejemplo, después de la ejecución de la etiqueta `<filled>`, el sistema queda a la espera de un enunciado del usuario. Este enunciado puede contener información referente a uno o varios campos del resto del formulario y, por tanto, completar a la vez varios de estos campos. El resto de campos sin completar serían visitados por el `FIA`.

Un posible diálogo para el ejemplo anterior sería:

SISTEMA: Hola. Vamos a programar mi horario de funcionamiento.

SISTEMA: Dime a qué hora quieres que me encienda y a qué hora quieres que me apague.

USUARIO: Quiero que te enciendas a las nueve de la mañana y que te apagues a las diez de la noche.

[*El* SISTEMA *entiende ambas horas y las asocia a cada uno de los campos. Por tanto no llega a entrar en ninguno de ellos. Sí lo hace en las etiquetas* `<filled>`]

SISTEMA: Me enciendo a las nueve de la mañana.

SISTEMA: Me apago a las diez de la noche.

Otro posible diálogo donde se muestra qué ocurre ante un fallo de reconocimiento de voz sería:

SISTEMA: Hola. Vamos a programar mi horario de funcionamiento.

SISTEMA: Dime a qué hora quieres que me encienda y a qué hora quieres que me apague.

USUARIO: Quiero que te enciendas a las nueve de la mañana y que te apagues a las "nuever".

[*El* SISTEMA *entiende la hora de encendido y la asigna al primer campo, pero no entiende la segunda hora. Así que el* `FIA` *entra solamente en el segundo campo.*]

SISTEMA: Me enciendo a las nueve de la mañana.

SISTEMA: Dime la hora de apagado.

USUARIO: Nueve de la noche.

[*El* SISTEMA *entiende el enunciado del usuario y asigna el resultado al segundo campo.*]

SISTEMA: Me apago a las nueve de la noche.

D.5.3. El menú

Un menú define un diálogo mediante la etiqueta `<menu>`. Este diálogo tiene un nombre que viene especificado por el atributo `id`.

Un menú enumera una serie de opciones al usuario, de modo que solamente una de ellas puede ser seleccionada. Cada opción está identificada por la etiqueta `<choice>` Las frases contenidas en cada una de las opciones sirven a la vez para ser sintetizadas por el sistema, mediante la etiqueta `<enumerate/>`, y para generar la gramática literal que sirva para reconocer la entrada de voz del usuario. Una vez realizada la selección, la aplicación salta al diálogo o subdiálogo establecido en la opción mediante el atributo `next`

Tanto la etiqueta `<menu>` como cada uno de las etiquetas `<choice>` admiten la especificación de una gramática general o particular para cada elección. Pero no es obligatoria y si no se especifica ninguna gramática, las frases contenidas en cada uno de las etiquetas `<choice>` sirven para que la propia plataforma de interpretación de `voiceXML` construya la gramática adecuada. En este caso tanto la etiqueta `<menu>` como cada uno de las etiquetas `<choice>` admiten un atributo `accept` que permite regular la construcción de dicha gramática. Este atributo puede tomar dos valores: "`exact`" o "`approximate`". En el primer caso, el usuario tendrá que decir la frase exacta colocada en la etiqueta `<choice>` En el segundo caso, bastará con que diga alguna de las palabras de dicha frase.

Por ejemplo En el siguiente ejemplo se ofrecen tres opciones al usuario y solamente en la última opción se obligaría al usuario a enunciar al completo la frase "salir del programa"

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu accept = "approximate">
  <prompt>
    Bienvenido al sistema. Escoja entre una de las siguientes opciones.
    <enumerate/>
  </prompt>
  <choice next="begin.vxml">
    Comenzar la aplicación
```

```
  </choice>
  <choice next="opciones.vxml">
   Configurar las opciones
  </choice>
  <choice accept = "exact" next="salir.vxml">
   Salir del programa
  </choice>
 </menu>
</vxml>
```

Cada opción tiene asociado un nuevo documento `voiceXML`. El sistema de diálogo realizará la transición al primer diálogo del documento asociado a la selección que el usuario realice.

Un ejemplo de un posible diálogo:

SISTEMA: Bienvenido al sistema. Escoja entre una de las siguientes opciones.

SISTEMA: Comenzar la aplicación.

SISTEMA: Configurar las opciones.

SISTEMA: Salir del programa.

USUARIO: Salir

[*El* SISTEMA *no entiende el enunciado del usuario y vuelve a repetir los enunciados del principio*]

SISTEMA: Bienvenido al sistema. Escoja entre una de las siguientes opciones.

SISTEMA: Comenzar la aplicación.

SISTEMA: Configurar las opciones.

SISTEMA: Salir del programa.

USUARIO: Comenzar

[*El* SISTEMA *entiende el enunciado del usuario y pasaría a cargar el documento begin.vxml*]

...

En la práctica, un formulario permite implementar también un menú de opciones, por tanto, para la realización del sistema de diálogo de la presente tesis, no se ha incorporado ninguna etiqueta `<menu>`.

D.6. Manejo de excepciones

La plataforma puede enviar y manejar distintos eventos a lo largo de la sesión. Los eventos pueden ser de dos tipos: predefinidos y definidos por el usuario. Cada uno de los eventos predefinidos se asocia a una gramática por defecto, que suele estar implementada, junto con otras opciones por defecto, en un fichero que la plataforma carga al iniciarse. Los eventos predefinidos más importantes que admite `voiceXML`, para la realización de esta tesis han sido:

- **cancel** El usuario ha solicitado interrumpir la síntesis de texto a voz de la frase en curso.
- **exit** El usuario solicita terminar la sesión.
- **help** El usuario solicita ayuda.
- **noinput** Ha pasado un tiempo umbral sin que se haya producido ninguna entrada por parte del usuario.
- **nomatch** El enunciado del usuario no casa con ninguna gramática cargada.
- **maxspeechtimeout** El enunciado del usuario ha sobrepasado el tiempo máximo de habla.
- **error.badfetch** Fallo al buscar un documento, por ejemplo porque su ruta no es correcta.
- **error.semantic** Error en tiempo de ejecución debido a alguna errata en el código `voiceXML`, al intentar utilizar una variable no declarada, al realizar una asignación incorrecta a una propiedad de la sesión, etc.
- **error.noresource** La plataforma solicita un recurso que no está disponible.
- **error.unsupported.builtin** Se ha solicitado el uso de gramáticas predefinidas (como las que se refieren a dinero, fechas, nombre de ciudades, etc...) y la plataforma no soporta dichas gramáticas.
- **error.unsupported.format** El recurso solicitado está en un formato no soportado, por ejemplo, el formato de una gramática, etc.

- **error.unsupported.language** El idioma escogido no está soportado por el sintetizador de texto a voz o por el reconocedor automático del habla.

- **error.unsupported.element** Se ha utilizado una etiqueta no soportado por la plataforma aun siendo una etiqueta válido en la definición del estándar `voiceXML`.

El evento `noinput` se emite gracias a un cronómetro interno del manejador de diálogo. Este cronómetro se inicializa cada vez que el usuario comienza a hablar, como se explicará más adelante, en la sección 7. Algunos ejemplos de cómo el manejador de diálogo implementado en el presente trabajo gestiona el silencio del usuario cuando éste es propietario del turno pueden verse en 7.2.1.

`<throw>` Para enviar un evento se utiliza la etiqueta `<throw/>` especificando el nombre del evento mediante el atributo `event`. Junto con el evento puede ir una cadena de caracteres a modo de mensaje, que se especifica mediante el atributo `message`. Por ejemplo:

```
<throw event = ‘‘help’’ message = ‘‘’tomandoDatos’’’/>
```

`<catch>` Para capturar un evento se utiliza la etiqueta `<catch>` En el atributo `event` se especifica qué evento va a ser manejado. La etiqueta `<catch>` asocia una captura a un documento, a un diálogo o a un formulario. Su contenido es de ámbito anónimo y contiene dos variables "ocultas" especiales `_event` y `_message`. La primera contiene el nombre del evento recibido, la segunda el contenido del atributo `message` de la etiqueta `<throw/>`, que fue el que envió el evento.

```
<catch event = "help noinput">
  <if cond = "_event == 'help'">
     <prompt>
       Ha entrado en ayuda dentro de <value expr = "_message"/>
     </prompt>
  <elseif cond = "_event == ' noinput'"/>
     <prompt>Lo siento pero no oigo nada</prompt>
  </if>
</catch>
```

Como se ve en el ejemplo, una sola etiqueta <catch> puede manejar varios eventos. Por otro lado, en un mismo ámbito, un mismo evento solamente puede ser manejado por un solo <catch> No obstante pueden existir varios capturadores de un mismo evento si residen en distintos ámbitos. Existe toda una política de manejo del evento, de modo que, si no existe un capturador en un ámbito la plataforma va buscando otro capturador en el ámbito superior. Así por ejemplo, en un formulario puede haber un capturador para un evento general a todos los campos del formulario o bien, cada campo puede escoger su propio capturador, etc.

Cuando un evento es capturado se maneja mediante el contenido ejecutable que alberga la etiqueta <catch> El ámbito de este contenido no es el ámbito donde reside el capturador, sino que la ejecución adquiere el ámbito donde fue lanzado el evento. Así por ejemplo, si un capturador se coloca a nivel de diálogo (por ejemplo en un formulario <form>) y el evento se produce a nivel anónimo (por ejemplo dentro de una etiqueta <filled>) el contenido ejecutable del capturador "hereda" por así decirlo, el ámbito anónimo, esto es, las variables y scripts albergadas en la etiqueta <filled> Además, si el contenido ejecutable del capturador envía a su vez un evento, su ámbito será también el ámbito anónimo, no el ámbito del capturador.

Existen cuatro etiquetas específicas para el manejo de los "eventos predefinidos" error, help, nomatch y noinput que son <error>, <help>, <nomatch> y <noinput> Su funcionamiento es exacto a la etiqueta <catch> y representan simplemente una forma abreviada de expresarlo así:

```
<catch event = evento_predefinido>
```

D.7. Variables, expresiones y ámbitos

<var> La etiqueta <var> permite declarar y hacer una asignación a una variable ECMAScript. El ámbito de esta variable depende del lugar, dentro del documento voiceXML, en el que se coloque esta etiqueta.

Si se coloca como hijo de una etiqueta <block>, <filled> o <catch> el ámbito se denomina anónimo. Y la visibilidad y vida de la variable es la que marcan estas etiquetas.

Si se coloca como hijo de una etiqueta <form>, puesto que esta etiqueta define un diálogo, el ámbito de la variable es el diálogo que se define.

Si se coloca como hijo de una etiqueta <vxml>, puesto que esta etiqueta define un documento, el ámbito de la variable es todo el documento. Ahora bien, si este documento es el documento principal de una aplicación, el ámbito de la variable será toda la aplicación, es decir, incluyendo al resto de

documentos secundarios de la aplicación. Si el documento es un documento secundario de una aplicación, el ámbito de la variable será solamente el de ese documento, y no el de otros de la misma aplicación.

El nombre y el valor de la variable que se define vienen dados, respectivamente, por los atributos `name` y `expr`.

El valor de una variable puede modificarse mediante la etiqueta `<assign/>` Esta etiqueta tiene los mismos atributos `name` y `expr` que denotan, respectivamente, el nombre y el valor de la variable que se quiere modificar.

El valor de una o más variables pueden borrarse mediante la etiqueta `<clear/>` En el atributo `namelist` se coloca la lista de las variables que van a ser borradas.

Nótese que el nombre de los campos de un formulario son variables `ECMAScript` con lo cual, también pueden verse afectados por las etiquetas `<assign/>` y `<clear/>`

`<property/>` Esta etiqueta permite definir propiedades que pueden afectar al reconocedor automático del habla, al sintetizador de texto a voz, a la plataforma o intérprete de `voiceXML` o a en cómo buscar ciertos recursos. También, mediante esta etiqueta, puede asignarse un valor inicial a un campo `<field>` dentro de un diálogo `<form>`, por lo que dicho campo no sería visitado por el `FIA` hasta que en algún contexto de ejecución se borrase su valor.

Las propiedades se definen mediante dos atributos en la etiqueta: `name`, que especifica el nombre de la propiedad, y `value`, que le asigna un valor.

Las propiedades, como las variables, tiene sus reglas de visibilidad dependiendo del ámbito donde se definan. Puede haber propiedades para toda la aplicación, para todo un documento (hijas de la etiqueta `<vxml>`), o simplemente para un diálogo (hijas de una etiqueta `<form>` o de una etiqueta `<menu>`) Una propiedad redefinida en un ámbito inferior sobrescribe a la definición en el ámbito superior. Si en un mismo ámbito hay dos asignaciones de una misma propiedad, permanece la última asignación.

Una plataforma puede contener propiedades específicas, es decir, propiedades que solo existen y afectan al entorno y funcionamiento de esa plataforma o intérprete `voiceXML`.

Las propiedades más estándar para el reconocedor automático del habla son:

- **confidencelevel**. Es un valor en el rango `0.0 - 1.0` que establece un valor umbral para el grado de confianza de un enunciado. Si el enunciado viene con un valor de confianza inferior al valor que establece esta propiedad es rechazado.
- **sensitivity**. Es un valor en el rango `0.0 - 1.0` que establece el grado de sensibilidad acústica de la captura de audio por parte del reconocedor. Valores altos hacen al reconocedor más sensible ante una señal de volumen bajo.
- **speedaccuracy**. Algunos reconocedores tienen un parámetro en sus algoritmos de reconocimiento que optimizan la velocidad de respuesta desfavoreciendo el grado de confianza del reconocimiento y viceversa. Es un valor en el rango `0.0 - 1.0`.
- **incompletetimeout**. Valor en segundos del tiempo de silencio que el reconocedor debe esperar después de que el usuario haya supuestamente terminado un enunciado, para terminar de procesar la señal que se ha ido capturando, si el enunciado del usuario todavía no ha completado totalmente una gramática. Por ejemplo ante un enunciado incompleto o "entrecortado". Este tiempo permite al usuario completar el enunciado o, lo que es lo mismo, utilizar silencios más o menos largos en la expresión de un mismo enunciado.
- **completetimeout**. Valor en segundos del tiempo de silencio que el reconocedor debe esperar después de que el usuario haya supuestamente terminado un enunciado, para terminar de procesar la señal que se ha ido capturando, si el enunciado del usuario ya ha completado totalmente una gramática. Esto permite al reconocedor asegurarse de que el enunciado efectivamente está expresado al completo y de separar un enunciado completo del siguiente.
- **maxspeechtimeout**. Duración máxima, en segundos, del habla del usuario. Sobrepasado este valor se emite el evento `maxspeechtimeout`.

Las propiedades estándar para el sintetizador de texto a voz son:

- **bargein**. Esta propiedad permite que un enunciado que está siendo sintetizado por el sistema se interrumpa cuando el usuario comienza a hablar. Toma dos valores: `true` o `false`.

- **bargeintype**. La ruptura de un enunciado en proceso de síntesis puede producirse porque el usuario simplemente comienza a hablar o porque dice una o varias palabras específicas, tomando así la propiedad dos posibles valores `speech` y `hotword`, respectivamente.

- **timeout**. Tiempo de espera después de un enunciado sintetizado por el sistema

Aunque existen más propiedades predefinidas, no han sido considerados de interés para los propósitos de esta tesis. Por último se comentan dos propiedades misceláneas:

- **universal**. Establece si se activan o no las gramáticas por defecto asociadas a los eventos predefinidos. Estas gramáticas se implementan en un archivo inicial que la plataforma carga al comienzo de su ejecución, tal y como se ha explicado en el apartado D.6. Puede tomar los siguientes valores: `all`, que carga todas esas gramáticas, `none`, que no carga ninguna, o bien el nombre del evento asociado a la gramática: `nomatch`, `cancel`, `help`, etc.

- **maxnbest**. Establece la dimensión de los "n" mejores resultados asignados a la variable vector `application.lastresult$`

Por ejemplo Algunas propiedades utilizadas en los documentos implementados para el sistema de diálogo del presente trabajo son:

```
<property name = bargein value = true/>
<property name = timeout value = 5s/>
<property name = confidencevalue value = 0.2/>
<property name = universal value = true/>
```

...

Expresiones Puesto que un documento `voiceXML` se "traduce" por la plataforma en un script en `ECMAScript` el documento acepta algunas etiquetas de control típicos en programación, e incluso un `script` completo.

Así, las etiquetas `<if>` `<else/>` y `<elseif>` permiten evaluar condiciones y desviar el flujo del diálogo hacia un diálogo u otro, lanzar un evento, realizar un cambio en alguna variable, etc.

Por ejemplo

```
<if cond="total > 1000">
  <prompt>El valor dado es demasiado grande</prompt>
  <throw event="AD.badValue"/>
</if>

<if cond="cantidad &lt; 0">
  <assign name="p" expr="20"/>
<else/>
  <assign name="x" expr="cantidad"/>
</if>

<if cond="color == 'azul'">
  <assign name="color_elegido" expr="'b'"/>
<elseif cond="color == 'rojo'"/>
  <assign name="color_elegido" expr="'r'"/>
<elseif cond="color == 'verde'"/>
  <assign name="color_elegido" expr="'g'"/>
<else/>
  <assign name="color_elegido" expr="'?'"/>
</if>
```

La etiqueta `<script>` permite encapsular todo un código en `ECMAScript`: definir variables, objetos, funciones, etc.

Por ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
    <script> <![CDATA[
       function factorial(n)
       {
         return (n <= 1)? 1 : n * factorial(n-1);
       }
    ]]> </script>
```

```
  <form id="form">
    <field name="fact">
      <grammar type="application/srgs+xml" src="/grammars/number.grxml"/>
      <prompt>
        Dime un número para conocer su factorial
      </prompt>
      <filled>
        <prompt>
          El factorial de <value expr="fact"/> es
          <value expr="factorial(fact)"/>
        </prompt>
      </filled>
    </field>
  </form>
</vxml>
```

que pregunta al usuario un número entero y comunica su factorial mediante la función implementada en `ECMAScript` Obsérvese la inclusión de `![CDATA[...]]` Este símbolo estándar, que es utilizado por la plataforma, indica, dentro de cualquier código en `XML`, un "contenido script"

Otro ejemplo sería:

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
  <form>
    <var name="horas"/>
    <var name="minutos"/>
    <var name="segundos"/>
    <block>
      <script>
        var d = new Date();
        hora = d.getHours();
        minutos = d.getMinutes();
        segundos = d.getSeconds();
      </script>
```

```
    </block>
    <field name="otra_hora">
      <grammar type="application/srgs+xml" src="/grammars/boolean.grxml"/>
      <prompt>
        Son las <value expr="hora"/> horas,
        <value expr="minutos"/> minutos, y
        <value expr="segundos"/> segundos.
      </prompt>
      <prompt>Quieres que te diga la hora actual?</prompt>
      <filled>
        <if cond="otra_hora">
          <clear/>
        </if>
      </filled>
    </field>
  </form>
</vxml>
```

que hace uso de la función estándar `Date()` para tomar la hora del sistema. Luego pregunta al usuario si quiere volver a saber la hora. Si contesta afirmativamente borra el campo correspondiente, haciendo que el `FIA` vuelva a entrar en él y, por tanto, vuelva a ser dicha la hora actual.

La etiqueta `<log>` es muy útil para realizar la depuración de un documento `voiceXML` ya que escribe por pantalla un mensaje. Puede hacer uso de etiquetas `<value/>` y contenido `CDATA` Por ejemplo:

```
<log> Posicion = <value expr = "posicion"/></log>
```

imprimiría por pantalla el valor de la variable `posicion`.

Ámbito anónimo Al hablar de los ámbitos de una sesión de un sistema de diálogo basado en `voiceXML` se nombró el ámbito *anónimo* que ahora se explica. Éste ámbito local es el que se encuentra dentro de alguno de las siguientes etiquetas: `<block>`, `<filled>` y `<catch>`.

D.8. Enlaces entre diálogos

Transición a un subdiálogo Dentro de las transiciones entre diálogos, además de lo comentado en D.2, cada diálogo puede derivar en uno o varios subdiálogos, y estos a su vez en otros subdiálogos. Un subdiálogo es un diálogo "hijo" que, al terminar, vuelve al diálogo "padre" en el contexto en que aquel fue llamado. Funciona de modo análogo a lo que es una llamada a una función.

En `voiceXML`, la diferencia entre un diálogo y un subdiálogo reside, no tanto en cómo se implementa uno respecto al otro, sino en cómo son llamados desde otro diálogo. La etiqueta `<subdialog>` se utiliza para realizar una llamada a un subdiálogo, que es, un diálogo al que se le pasan parámetros. La dirección `URI` del subdiálogo se coloca en el campo `src` de la etiqueta.

La llamada a un subdiálogo abre un nuevo contexto de ejecución en donde no es visible ninguna variable o script de un ámbito superior (documento, aplicación o sesión) del diálogo que realiza la llamada, aunque éste resida en el mismo documento.

El subdiálogo termina una vez el `FIA` haya visitado todos los campos del subdiálogo o encuentre una etiqueta `<return/>` o un `<exit/>`. Cuando el subdiálogo termina, el nuevo contexto creado es borrado completamente y el diálogo que realizó la llamada solamente recoge la información que el subdiálogo le envíe mediante la etiqueta `<return/>`

La etiqueta `<return/>` finaliza la ejecución de un subdiálogo devolviendo el control de la ejecución al diálogo que realizó la llamada y devolviendo también una serie de variables especificadas mediante el atributo `namelist`. Estas variables se devuelven como propiedades de un objeto que tiene por nombre el nombre del subdiálogo, dado en el atributo `name` de la etiqueta `<subdialog>` Además, `<return/>` puede lanzar un evento, lo cual viene especificado mediante el atributo `event`

La etiqueta `<subdialog>` puede albergar las mismas etiquetas que un formulario (`<form>`). Además puede contener etiquetas `<param/>` que sirven para especificar las variables que, en la llamada, se pasan al subdiálogo. Estas variables tienen que estar declaradas en el subdiálogo con el mismo nombre con que se especificaron en `<param/>`

Por ejemplo En el siguiente ejemplo se muestra una llamada a un subdiálogo, dentro de un diálogo-formulario. El subdiálogo se implementa como un diálogo-formulario, en el que se declara la variable *edadUmbral* y el campo *edad* El subdiálogo pregunta al usuario su edad y comprueba si es mayor

que *edadUmbral* en cuyo caso pone una variable *mayor* a verdadero (en caso contrario a falso) La llamada al subdiálogo se realiza mediante la etiqueta `<subdialog name="mayorEdad"src="#verificaMayorEdad>` que utiliza la etiqueta `<param name=.edadUmbral.expr="'18'/>` para declarar la variable *edadUmbral* y asignarle un valor. Esta variable es asignada en la variable local *edadUmbral* en el subdiálogo, gracias a que tienen el mismo nombre. El subdiálogo finaliza mediante la etiqueta `<return namelist=.edad mayor/>` que devuelve las variables *edad* y *mayor* al diálogo que realizó la llamada. Estas variables son utilizadas en el diálogo como propiedades del objeto *mayorEdad*, que coincide con el nombre del subdiálogo.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<!-- formulario diálogo que llama a un subdiálogo -->
<form>
  <subdialog name="mayorEdad" src="#verificaMayorEdad">
   <param name="edadUmbral" expr="'18'"/>
   <filled>
     <prompt>Tienes <value expr = "mayorEdad.edad"> años</prompt>
     <if cond = "mayorEdad.mayor">
       <prompt>Y eres mayor de edad</prompt>
     <else/>
       <prompt>Y no eres mayor de edad</prompt>
     </if>
   </filled>
  </subdialog>
</form>

<!-- subdiálogo para verificar si es mayor de edad -->
<form id="verificaMayorEdad">
  <var name="edadUmbral"/>
  <field name="edad">
   <grammar src="numeros.grxml" type="application/srgs+xml"/>
   <prompt>Cuántos años tienes?</prompt>
   <filled>
     <if cond="edad &gt; edadUmbral">
       <var name="mayor" expr="true"/>
```

```
    <else/>
      <var name="mayor" expr="false"/>
    </if>
    <return namelist="edad mayor"/>
   </filled>
  </field>
</form>
</vxml>
```

Transición a otro diálogo Al comenzar a describir el estándar propuesto para un sistema de diálogo `voiceXML` se habló de que tal sistema no es sino un autómata cuyos estados se denominan *diálogos*. En esta sección se habla de cómo se implementa en un documento `voiceXML` la transición entre dichos diálogos.

La etiqueta `<link>` permite enlazar una o más gramáticas a un documento o a un diálogo concreto dentro de un documento. El ámbito donde esté implementado determina el ámbito de la gramática enlazada. Así, puede haber enlaces a nivel de aplicación, de diálogo o dentro de una etiqueta de un diálogo (ámbito anónimo) Además, la etiqueta `<link>` permite lanzar un evento cuando el enunciado del usuario casa con alguna de las gramáticas enlazadas. El diálogo o documento al cual se realiza el enlace viene especificado por el atributo `next`. Por ejemplo:

```
<link next = ayuda.vxml>
  <grammar mode="voice" version="1.0" root="root">
   <rule id="root" scope="public">
      <one-of>
        <item>ayuda</item>
        <item>no entiendo nada</item>
      </one-of>
   </rule>
 </grammar>
 <grammar mode="voice" version="1.0" root="r2">
    <rule id="r2" scope="public"> socorro </rule>
 </grammar>
</link>
```

En este ejemplo, ante alguno de los enunciados “ayuda”, “no entiendo

nada" o "socorro", el diálogo en curso realizaría una transición al primer diálogo (etiqueta <form> o <menu>) del documento "ayuda.vxml"

La etiqueta <goto/> permite hacer alguna de las siguientes transiciones:

- De un campo de un formulario a otro campo del mismo formulario (ámbito "diálogo")
- De un diálogo a otro en el mismo documento (ámbito "documento")
- De un diálogo a otro en distintos documentos en la misma aplicación (ámbito "aplicación")
- De un diálogo a otro en distintos documentos de distintas aplicaciones (ámbito "sesión")

La especificación del diálogo al que se realiza la transición viene dada en el atributo next y ha de ser una URI existente. Si no, la plataforma emite el evento error.badfetch. Cuando la URI contenga solamente su fragmento (es decir, que sea de la forma #nuevo-dialogo, se considera que el diálogo al que se realiza la transición está implementado en el mismo documento. Solamente en este caso no se realiza una búsqueda de un nuevo documento, por tanto, se mantienen tanto la visibilidad como el valor de las variables en el actual ámbito. En el resto de casos, se realiza una "carga" del nuevo documento y, por tanto, se pierden los datos asociados al actual ámbito (variables y scripts), es decir, no se mantiene le contexto. Esto ocurre incluso si la transición se realiza al mismo documento. Así por ejemplo, imaginemos las siguiente lineas en un documento llamado miDocumento.vxml:

```
<goto next = "miDocumento.vxml#miDialogo"/>
<goto next = "#miDialogo"/>
```

Mientras que la segunda línea mantiene el contexto en la transición, pues solamente especifica el denominado fragmento de una URL, en la primera línea el contexto no se mantiene ya que se vuelve a cargar el propio documento. Ambas líneas van al mismo diálogo: miDialogo, pero llegan con distinto contexto.

La etiqueta <submit/> está diseñado para enviar y recoger datos de un servidor. Esto se hace mediante las llamadas GET y POST propias de HTTP. Pero en el presente trabajo no se utiliza con este fin, dado que también puede utilizarse para realizar una transición entre diálogos análoga a la que se realiza con <goto/> pero manteniendo el valor de ciertas variables. El

documento al que se realiza la transición se especifica en el atributo `next` y la lista de variables que se quiere enviar en el atributo `namelist`.

```
<submit next = "miDocumento.vxml#miDialogo"/>
<submit next = "#miDialogo"/>
```

E. GRAMÁTICA PARA EL EDITOR VERBAL NO INTERACTIVO DE SECUENCIAS

En este apéndice se muestra las gramáticas utilizadas para el sistema descrito en el capítulo 6.

maggieSelf.gram :

```
#ABNF 1.0 ISO-8859-1;

language es-ES;
tag-format <loq-semantics/1.0>;
public $root = $root_maggieSelf;

public $root_maggieSelf = $body;

public $body = ("cabeza":head | "ojo|ojos|parpado|parpados":eye |
                "cuello":neck | "hombro|brazo":arm | "mano":hand |
"base|(ruedas de abajo)":base){<@body $value>};
```

sve.gram :

```
#ABNF 1.0 ISO-8859-1;

language es-ES;
tag-format <loq-semantics/1.0>;
public $root = $sve;


/* gramatica utilizada por la clase Cdialog. Gramatica basada en DASML */

$sve = ( $nd | $ad | $sd | $e);
```

```
$polite = porfi | por_favor;

/* articulos */
$aDet = el | la | lo | los | las;
$aIn = un | una | uno | unos | unas;
$a = $aDet | $aIn;

/* nd - Dispreferred answers */
$simpleseqEx = [baile | secuencia] simple {<@da "nd"><@type "simpleseq">};
$atonceEx = [varios] [bailes | secuencias] en paralelo
           {<@da "nd"><@type "atonce">};
$selseqEx = seleccion [de (bailes | secuencias)]
           {<@da "nd"><@type "selseq">};
$bodyEx = $<maggieSelf.gram#body>  {<@da "nd"><@type "body">};
$sideEx = $side {<@da "nd"><@type "side">};
$actionEx = accion {<@da "nd"><@type "action">};
$conditionEx = condicion {<@da "nd"><@type "condition">};

$nd = $simpleseqEx | $atonceEx | $selseqEx | $bodyEx |
     $sideEx | $actionEx | $conditionEx;

/* ad - action-directive */
$ad = ($action|$condition|$then|$atonce|$selseqinit|$loop|
     $runseq|$stopseq|$rmseq){<@da "ad">};

/* statement-non-opinion */
$sd = (([$GARBAGE|en (el|la)]($cardinal| $ordinal)[$GARBAGE|"baile"])|
     $conclution|$branchend){<@da "sd">};

/* uninterpretable */
$u = $GARBAGE {<@da "u">};

/* turn-exit */
$e = [quiero] (sal|termina|salir|terminar) {<@da "e">};

/* numeros */
$cardinal = ( "un | uno | una":1 | "dos":2 | "tres":3 |
           "cuatro":4 | "cinco":5 | "seis":6 | "siete":7 |
```

```
           "ocho":8 | "nueve":9 ) {<@cardinal $value>};
$uncontable = ("varios | varias | distintos | distintas":2 | "alguno |
             alguna":1 ){<@uncontable $value>};
$ordinal = ("primer|primero|primera":1 | "segun|segundo|segunda":2 |
          "tercer|tercero|tercera":3 | "cuarto|cuarta":4 |
          "quinto|quinta":5 | "sexto|sexta":6 | "septimo|septima":7 |
          "octavo|octava":8 | "noveno|novena":9 )
          {<@ordinal $value><@type "ordinal">};

/* acciones y condiciones */
/*$action = ([magui] $verb [$howmuch] [$<maggieSelf.gram#body>]
           [$side] [$howmuch]){<@type "action">};*/
$action = ([magui] $verb [$<maggieSelf.gram#body>] [$side])
           {<@type "action">};
$condition = ([magui] (cuando | si |esperas [a | que]) [te] $verb
            [$<maggieSelf.gram#body>] [$side]){<@type "condition">};
$then = (despues  [de ($action | $condition)] | lo siguiente |
       a continuacion) {<@type "then">};

$verb = ("levanta|levantas|sube|subes|abre|abres":rise |
       "baja|bajas|cierra|cierras":down | "gira|giras":spin |
       "mueve":move | "toque|toco":touch){<@verb $value>};
$side = ("izquierda|izquierdo":left | "derecha|derecho":right |
       "ambas|ambos":both){<@side $value>};
$howmuch = ("[muy]poco|[un] poquitin":0.2 | "ni_mucho_ni_poco|algo":0.5 |
          "mucho|a_tope|todo":1.0){<@howmuch $value>};

/* comandos de edicion de la secuencia */
$selseqinit = ((( abres | consideras | creas ) ($cardinal|$uncontable)
   ( opciones | posibilidades )) | (puedo hacer $cardinal cosas))
   {<@type "selseq">};
$atonce = (a_la_vez [haces] [$cardinal|$uncontable] [cosas])
         {<@type "atonce">};
$branchend = (Terminas | Por ultimo | para terminar | Y finalmente | despues
   [de (todo | algo)] ($action | [esperas a que] $condition))
   {<@type "branchend">};
$conclution = ( (despues de [hacer [alguna de]] las [$cardinal] [ramas]) |
             ([luego|y] ( cierras | unes | terminas) las
             [$cardinal] opciones) ) {<@type "conclution">};
```

```
$loop = [despues | (por ultimo)] ((vuelves al principio)|
       (repites la secuencia)) {<@type "loop">};

/* comandos de ejecucion de la secuencia */
$runseq = ([magui] ejecuta la secuencia){<@type "runseq">};
$stopseq = ([magui] ( (para de ejecutar [la secuencia]) | (estate quieta) ))
          {<@type "stopseq">};
$rmseq = ([magui] (nueva | borra) secuencia){<@type "rmseq">};
```

F. GRAMÁTICA PARA EL EDITOR VERBAL INTERACTIVO DE SECUENCIAS

En este apéndice se muestra las gramáticas utilizadas para el sistema descrito en el capítulo 8.

maggieSelf.gram :

```
#ABNF 1.0 ISO-8859-1;

language es-ES;
tag-format <loq-semantics/1.0>;
public $root = $root_maggieSelf;

public $root_maggieSelf = $body;

public $body = ("cabeza":head | "ojo|ojos|parpado|parpados":eye |
                "cuello":neck | "hombro|brazo":arm | "mano":hand |
"base|(ruedas de abajo)":base){<@body $value>};
```

dialog.gram :

```
#ABNF 1.0 ISO-8859-1;

language es-ES;
tag-format <loq-semantics/1.0>;
public $root = $dialog;

/* gramatica utilizada por la clase Cdialog. Gramatica basada en DASML */

$dialog = ($fp | $fc | $ky | $nn | $no | $nd | $ad | $sd | $h | $e);
```

```
/**/
$polite = porfi | por_favor;

/* articulos */
$aDet = el | la | lo | los | las;
$aIn = un | una | uno | unos | unas;
$a = $aDet | $aIn;

/* conventional-opening */
$fp = hola [magui]{<@da "fp">};

/* conventional-closing */
$fc = adios [magui]{<@da "fc">};

/* response acknowledgement or yes answer */
$ky = (okei|vale|de_acuerdo|si|afirmativo|correcto|[que] bien){<@da "ky">};

/* no answer */
$nn = (no|negativo|incorrecto|[que] mal){<@da "ny">};

/* other answer */
$no = (no_lo_se | no_se ) {<@da "no">};

$simpleseqEx = [baile | secuencia] simple {<@da "nd"><@type "simpleseq">};
$atonceEx = [varios] [bailes | secuencias] en paralelo
           {<@da "nd"><@type "atonce">};
$selseqEx = seleccion [de (bailes | secuencias)]
           {<@da "nd"><@type "selseq">};
$bodyEx = $<maggieSelf.gram#body>  {<@da "nd"><@type "body">};
$sideEx = $side {<@da "nd"><@type "side">};
$actionEx = accion {<@da "nd"><@type "action">};
$conditionEx = condicion {<@da "nd"><@type "condition">};

$nd = $simpleseqEx | $atonceEx | $selseqEx | $bodyEx |
     $sideEx | $actionEx | $conditionEx;

/* action-directive */
$ad = ($action|$condition|$then|$atonce|$selseqinit|$loop|
     $runseq|$stopseq|$rmseq){<@da "ad">};
```

```
/* Hedge */
$h = [$polite] (([necesito] ayuda) | (podrias ayudarme) | ayudame |
    (no [te] entiendo [nada]) | (echame una mano)) [$polite]{<@da "h">};

/* statement-non-opinion */
$sd = (([$GARBAGE|en (el|la)]($cardinal| $ordinal)[$GARBAGE|"baile"])|
     $conclution|$branchend){<@da "sd">};

/* uninterpretable */
$u = $GARBAGE {<@da "u">};

/* turn-exit */
$e = [quiero] (sal|termina|salir|terminar) {<@da "e">};

/* numeros */
$cardinal = ( "un | uno | una":1 | "dos":2 | "tres":3 |
           "cuatro":4 | "cinco":5 | "seis":6 | "siete":7 |
           "ocho":8 | "nueve":9  ){<@cardinal $value>};
$uncontable = ("varios | varias | distintos | distintas":2 |
             "alguno | alguna":1 ){<@uncontable $value>};
$ordinal = ("primer|primero|primera":1 | "segun|segundo|segunda":2 |
"tercer|tercero|tercera":3 | "cuarto|cuarta":4 | "quinto|quinta":5 |
         "sexto|sexta":6 | "septimo|septima":7 | "octavo|octava":8 |
         "noveno|novena":9 ){<@ordinal $value><@type "ordinal">};

/* acciones y condiciones */
$action = ([magui] $verb [$<maggieSelf.gram#body>] [$side])
          {<@type "action">};
$condition = ([magui] (cuando | si |esperas [a | que])
           [te] $verb [$<maggieSelf.gram#body>] [$side])
           {<@type "condition">};
$then = (despues  [de ($action | $condition)] | lo siguiente |
        a continuacion){<@type "then">};

$verb = ("levanta|levantas|sube|subes|abre|abres":rise |
       "baja|bajas|cierra|cierras":down | "gira|giras":spin |
       "mueve":move | "toque|toco":touch){<@verb $value>};
```

```
$side = ("izquierda|izquierdo":left | "derecha|derecho":right |
       "ambas|ambos":both){<@side $value>};
$howmuch = ("[muy]poco|[un] poquitin":0.2 | "ni_mucho_ni_poco|algo":0.5 |
          "mucho|a_tope|todo":1.0){<@howmuch $value>};

/* comandos de edicion de la secuencia */
$selseqinit = ((( abres | consideras | creas ) ($cardinal|$uncontable)
             ( opciones | posibilidades )) | (puedo hacer $cardinal cosas))
             {<@type "selseq">};
$atonce =(a_la_vez [haces] [$cardinal|$uncontable] [cosas]){<@type "atonce">};
$branchend = (Terminas | Por ultimo | para terminar | Y finalmente |
            despues [de (todo | algo)] ($action | [esperas a que] $condition))
            {<@type "branchend">};
$conclution = ( (despues de [hacer [alguna de]] las [$cardinal] [ramas]) |
       ([luego|y] ( cierras | unes | terminas) las [$cardinal] opciones) )
       {<@type "conclution">};
$loop = [despues | (por ultimo)] ((vuelves al principio)|
       (repites la secuencia)){<@type "loop">};

/* comandos de ejecucion de la secuencia */
$runseq = ([magui] ejecuta la secuencia){<@type "runseq">};
$stopseq = ([magui] ( (para de ejecutar [la secuencia]) |
          (estate quieta) )){<@type "stopseq">};
$rmseq = ([magui] (nueva | borra) secuencia){<@type "rmseq">};
```

BIBLIOGRAFíA

[Ansina, 1989] Ansina, M. R. (1989). *Los modelos de la comunicación*. Tecnos, Madrid, España.

[Ausitn, 1962] Ausitn, J. (1962). *How to do things with words: The William James Lectures delivered at Harvard University*. J. O. Urmson. Oxford: Clarendon.

[Barber, 2001] Barber, R. (2001). *Desarrollo de una aquitectura para robots móviles autónomos. Aplicación a un sistema de navegación topológica.* PhD thesis, Universidad Carlos III de Madrid, Spain.

[Barber and Salichs, 2001] Barber, R. and Salichs, M. (2001). A new human based architecture for intelligent autonomous robots. In *The Fourth IFAC Symposium on Intelligent Autonomous Vehicles*, pages 85–89.

[Biggs and Macdonald, 2003] Biggs, G. and Macdonald, B. (2003). A survey of robot programming systems. In *in Proceedings of the Australasian Conference on Robotics and Automation, CSIRO*, volume 1.

[Billard et al., 2006] Billard, A., Robins, B., Dautenhahn, K., and Nadel, J. (2006). Building robota, a mini-humanoid robot for the rehabilitation of children with autism. *the RESNA Assistive Technology Journal*.

[Birdwhistell, 1970] Birdwhistell, R. L. (1970). *Kinesics and Context. Essays on Body Motion Communication*. University of Pennsylvania Press, Philadelphia, USA.

[Bischoff and Graefe, 2004] Bischoff, R. and Graefe, V. (2004). Hermes, a versatile personal robotic assistant. *Proceedings of the IEEE*, 92(11):1759–1779.

[Bobrow, 1968] Bobrow, D. (1968). Natural language input for a computer problem solving system.

[Bobrow et al., 1977] Bobrow, D., Kaplan, R., Kay, M., Norman, D., Thompson, H., and Winograd, T. (1977). Gus: A frame-driven dialogue system.

[Breazeal, 2002] Breazeal, C. (2002). *Designing Sociable Robots*. The MIT Press.

[Breazeal et al., 2004] Breazeal, C., Brooks, A., Chilongo, D., Gray, J., Hoffman, A., Lee, C. K. H., Lieberman, J., and Lockered, A. (2004). Working collaboratively with humanoid robots. *ACM Computers in Entertainment*, 2(3).

[Breazeal and Scasselatti, 2000] Breazeal, C. and Scasselatti, B. (2000). Infant-like social interactions between a robot and a human caretaker. *Adaptive Behavior*, 8(1):49–74.

[Bunt, 1979] Bunt, H. (1979). Conversational principles in question-answer dialogus. In *Zur Theory der Frage*, pages 119–141. Gunter Narr Verlag.

[Cahn, 1990] Cahn, J. E. (1990). The generation of affect in synthesized speech. *Journal of the American Voice I/O Society*, 8:1–19.

[Calinon and Billard, 2004] Calinon, S. and Billard, A. (2004). Gesture recognition and reproduction for a humanoid robot using hidden markov models. AMI/PASCAL/IM2/M4 Workshop on Multimodal Interaction and Related Machine Learning Algorithms.

[Calinon et al., 2007] Calinon, S., Guenter, F., and Billard, A. (2007). On Learning, Representing and Generalizing a Task in a Humanoid Robot. *IEEE transactions on systems, man and cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298.

[Cassimatis et al., 2004] Cassimatis, N., Trafton, J., Bugajska, M., and Schultz, A. (2004). Integrating cognition, perception and action through mental simulation in robots. Technical report, Naval Research Laboratory.

[Chomsky, 1956] Chomsky, N. (1956). Three models for the description of language. *IRI Transactions on Information Theory*, 2(3):113–124.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structures*. Mouton, Berlin.

[Chomsky, 1965] Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

[Colby, 1975] Colby, K. M. (1975). *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. Elsevier Science Inc., New York, NY, USA.

[Cole et al., 1997] Cole, R., Mariani, J., Uszkoreit, H., Zaenen, A., and Zue, V. (1997). *Survey of the state of the art in human language technology*. Cabridge University Press.

[Davis, 1971] Davis, F. (1971). *Inside Intuition-What we know about Non-Verbal Communication*. McGraw-Hill Book Co.

[de Saussure, 1916] de Saussure, F. (1916). *Curso de lingüística general*. Losada, Buenos Aires, Argentina.

[Dominey et al., 2007] Dominey, P. F., Mallet, A., and Yoshida, E. (2007). Progress in programming the hrp-2 humanoid using spoken language. In *International Conference on Robotics and Automation (ICRA07)*.

[Ekman, 1999] Ekman, P. (1999). Basic emotions. *Handbook of Cognition and Emotion*, 3:45–60.

[Ekman and Friesen, 1969] Ekman, P. and Friesen, W. V. (1969). The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *Semiotica*, 1:49–98.

[Ferguson and Allen, 1998] Ferguson, G. and Allen, J. (1998). Trips: An intelligent integrated problem-solving assistant. In *Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 567–573.

[Fonga et al., 2003] Fonga, T., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and Autonomouns Systems*, 42.

[Furui, 2005] Furui, S. (2005). Speech recognitio technology in multimodal/ubiquitous computing environments. *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*.

[Gates, 2007] Gates, B. (2007). A robot in every home. *Scientific American*.

[Gockley et al., 2005] Gockley, R., Bruce, A., Forlizzi, J., Michalowski, M., Mundell, A., Rosenthal, S., Sellner, B. P., Simmons, R., Snipes, K., Schultz, A., and Wang, J. (2005). Designing robots for long-term social interaction. In *Proceedings of IROS 2005*, pages 1338 – 1343.

[Gorostiza and Salichs, 2006] Gorostiza, J. F. and Salichs, M. A. (2006). Implementación de una habilidad tts con expresión de emociones en un robot personal autónomo. In *XXVII Jornadas de Automática*, Madrid, España. Comite Español de Automática (CEA).

[Gorostiza and Salichs, 2008] Gorostiza, J. F. and Salichs, M. A. (2008). Sintetizador de voz con control de la entonación para interacción humano robot. In *4thRobocity2030*, Madrid, España.

[Gorostiza et al., 2006] Gorostiza, J. F., Salichs, M. A., Barber, R., Khamis, A., Malfaz, M., Pacheco, R., Rivas, R., Corrales, A., Delgado, E., and García, D. (2006). Multimodal human-robot interaction framework for a personal robot. In *RO-MAN 06: The 15th IEEE International Symposium on Robot and Human Interactive Communication*, Hatfield, U.K. IEEE.

[Green et al., 1963] Green, B., Wolf, A., Chomsky, C., and Laughery, K. (1963). Baseball: an automatic question answerer. pages 545–549.

[Grosz and Sidner, 1986] Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

[Haasch et al., 2004] Haasch, A., Hohenner, S., Hüwel, S., Kleinehagenbrock, M., Lang, S., Toptsis, I., Fink, G. A., J. Fritsch, B. W., and Sagerer, G. (2004). Biron - the bielefeld robot companion. In *Proceedings International Workshop on Advances in Service Robotics*, pages 27–32.

[Hersch et al., 2008] Hersch, M., Guenter, F., Calinon, S., , and Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467.

[HomeLab, 2007] HomeLab, P. R. (2007). Philips icat.

[Hüwel et al., 2006] Hüwel, S., Wrede, B., and Sagerer, G. (2006). Robust Speech Understanding for Multi-Modal Human-Robot Communication. In *15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06)*, pages 45–50.

[I.E.C., 1993] I.E.C. (1993). *Preparation of Control Chart for Control Systems: Sequential Function Chart (SFC) Standard*. IEC 61131-3.

[Ishiguro et al., 2002] Ishiguro, H., Miyashita, T., Kanda, T., Ono, T., and Imai, M. (2002). Robovie: An interactive humanoid robot - toward new information media support communications. In *Video Proceedings of IEEE Int. Conf. Robotics and Automation (ICRA)*.

[Ishiguro et al., 2001] Ishiguro, H., Ono, T., Imai, M., Maeda, T., Kanda, T., and Nakatsu, R. (2001). Robovie: an interactive humanoid robot. *Industrial Robotics*, 28:498–503.

[Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice-Hall, Inc., New Jersey, USA.

[Kanda et al., 2004] Kanda, T., Ishiguro, H., Imai, M., and Ono, T. (2004). Developement and evaluation of interactive humanoid robots. *IEEE*, 92(11):1839–1850.

[Kanda et al., 2002] Kanda, T., Ishiguro, H., Imai, M., Ono, T., and Mase, K. (2002). A constructive approach for developing interactive humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1265–1270.

[Kandinsky, 1912a] Kandinsky, W. (1912a). *De lo espiritual en el arte*. Paidós Ibérica, Barcelona, España.

[Kandinsky, 1912b] Kandinsky, W. (1912b). *La gramática de la creación*. Paidós Ibérica, Barcelona, España.

[Kidd and Breazeal, 2004] Kidd, C. and Breazeal, C. (2004). Effect of a robot on user perceptions. Sendai, Japan. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004).

[Klingspor et al., 1997] Klingspor, V., Demiris, J., and Kaiser, M. (1997). Human-robot-communication and machine learning. *Applied Artificial Intelligence*, 11(7/8):719–746.

[Kruijff, 2005] Kruijff, G.-J. M. (2005). Contextually appropriate utterance planning for ccg. In *9th European Workshop on Natural Language Generation*.

[Kruijff et al., 2007] Kruijff, G.-J. M., Zender, H., Jensfelt, P., , and Christensen, H. I. (2007). Situated dialogue and spatial organization: What, where... and why? *International Journal of Advanced Robotic Systems, Special Issue on Human and Robot Interactive Communication*, 4(2).

[Kruijff et al., 2006] Kruijff, G.-J. M., Zender, H., Jensfelt, P., and Christensen, H. I. (2006). Clarification dialogues in human-augmented mapping. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 282–289, New York, NY, USA. ACM.

[Larsson et al., 2004] Larsson, S., Berman, A., Bos, J., Grönqvist, L., Ljunglöf, P., and Traum, D. (2004). *TRINDIKIT 3.1 Manual*. Goteborg University, Goteborg, Suiza.

[Lauria et al., 2001] Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., and Klein, E. (2001). Training personal robots using natural language instruction. *IEEE Intelligent Systems*, 16:38–45.

[Lauria et al., 2002] Lauria, S., Bugmann, G., Kyriacou, T., and Klein, E. (2002). Mobile robot programming using natural language. *Robotics and Autonomous Systems*, 38:171–181.

[Lozano, 1982] Lozano, T. (1982). Robot programming. Technical report, Memo 698, MIT AI. También en Proceedings of the IEEE. Vol 71, Julio 1983, pp.821–841. Y en IEEE Tutorial on Robotics, IEEE Computer Society, 1986, pp.455–475.

[Luria, 1980] Luria, A. (1980). *Fundamentos de neurolingüística*. Toray Masson, S. A., Barcelona, Spain, 1 edition.

[Malfaz and Salichs, 2004] Malfaz, M. and Salichs, M. A. (2004). A new architecture for autonomous robots based on emotions. Lisboa, Portugal. Fifth IFAC Symposium on Intelligent Autonomous Vehicles.

[Martelli, 2006] Martelli, A. (2006). *Python in a Nutshell (In a Nutshell (O'Reilly))*. O'Reilly Media, Inc.

[McTear, 2004] McTear, M. F. (2004). *Spoken Dialogue Technology, Toward the Conversational User Interface*. Springer.

[Mori, 1970] Mori, M. (1970). The uncanny valley. *Energy*, 4(7):33–35.

[Morris, 1938] Morris, C. (1938). *Fundamentos de la teoría de los signos.* Paidós, Barcelona, España.

[Mortensen, 1972] Mortensen, C. D. (1972). *Communication: The Study of Human Interaction.* Mcgraw-Hill, Dortmound, Alemania and Rondebosch, Sudáfrica.

[Ochs et al., 1996] Ochs, E., Schegloff, E. A., and Thompson, S. A. (1996). *Interaction and Grammar.* Cambridge University Press, New York, USA.

[Ogata, 1993] Ogata, K. (1993). *Ingeniería de control moderna.* Prentice-Hall, México.

[Omrom, 2007] Omrom (2007). Necoro. aibo.pdf.

[Oudeyer, 2003] Oudeyer, P.-Y. (2003). The production and recognition of emotions in speech: features and algorithms. *International Journal Human - Computer Studies*, 59(1-2):157–183.

[Pallotta, 2003] Pallotta, V. (2003). Computational dialogue models. In *EACL 2003: 11th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

[Paul Watzlawick, 1967] Paul Watzlawick, Janet Helmick Beavin, D. D. J. (1967). *Pragmatics of Human Communication. A study of Interactional patterns, pathologies and paradoxes.* Tiempo contemporáeo, Buenos Aires, Argentina.

[Robins et al., 2003] Robins, B., Dautenhahn, K., Boekhorst, R., and Billard, A. (2003). Effect of repeated exposure of a humanoid robot on children with autism - can we encourage basic social interaction skills. In Proceedings of CWUATT.

[Salichs et al., 2006] Salichs, M. A., Barber, R., Khamis, A., Malfaz, M., Gorostiza, J., Pacheco, R., Rivas, R., Corrales, A., Delgado, E., and García, D. (2006). Maggie: A robotic platform for human-robot social interaction. In *Submitted to IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2006)*, Bankgok, Thailand. IEEE.

[Sawada et al., 2008] Sawada, H., Kitani, M., and Hayashi, Y. (2008). A robotic voice simulator and the interactive training for hearing-impaired people. *Journal of Biomedicine and Biotechnology*, 2008:768232.

[Schultz, 2004] Schultz, A. (2004). Cognitive tools for humanoid robots in space. In *Proceedings of the 16th IFAC Conference on Automatic Control in Aerospace*, Elsevier, Oxford.

[Searle, 1970] Searle, J. R. (1970). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

[Sebeok, 1966] Sebeok, T. A. (1966). *Style in language*. The MIT Press, Massachusetts, USA.

[Seneff and Polifroni, 2000] Seneff, S. and Polifroni, J. (2000). Dialogue management in the mercury flight reservation system. In *ANLP/NAACL 2000 Workshop on Conversational systems*, pages 11–16, Morristown, NJ, USA. Association for Computational Linguistics.

[Shannon, 1948] Shannon, C. E. (1948). A mathematic theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

[Sharma and Kunins, 2002] Sharma, C. and Kunins, J. (2002). *VoiceXML: Strategies and Techniques for Effective Voice Application Development with VoiceXML 2.0*. John Wiley and Sons, Inc.

[Shibata, 2007] Shibata, T. (2007). Mental commitment robot [paro].

[Skubic et al., 2004] Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., and Brock, D. (2004). Spatial language for human–robot dialogs. *IEEE Transactions on Systems, Man and Cybernetics. Part-C: Applications and Reviews*, 34.

[Sony, 2007] Sony (2007). Aibo.

[Steil et al., 2001] Steil, J., Heidemann, G., Jockusch, J., Rae, R., Jungclaus, N., and Ritter, H. (2001). Guiding attention for grasping tasks by gestural instruction: the gravis-robot architecture. In *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, page 1570–1577.

[Strobel et al., 2002] Strobel, M., Illmann, J., Kluge, B., and Marrone, F. (2002). Using spatial context knowledge in gesture recognition for commanding a domestic service robot. In *In Proc. 11th IEEE Workshop on Robot and Human Interactive Communication (RO-MAN'02*, pages 468–473.

[Technology and Services., 2010] Technology, L. V. and Services. (2010). Loquendo: Sociedad global - soluciones automáticas y tecnología vocal.

[Toptsis et al., 2004] Toptsis, I., Li, S., Wrede, B., and Fink, G. A. (2004). A multi-modal dialog system for a mobile robot. In *International Conference on Spoken Language Processing*, volume 1, pages 273–276.

[Trafton et al., 2006] Trafton, J. G., Schultz, A. C., Perznowski, D., Bugajska, M. D., Adams, W., Cassimatis, N. L., and Brock, D. P. (2006). Children and robots learning to play hide and seek. In *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 242–249, New York, NY, USA. ACM Press.

[Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*.

[Vygotsky, 2003] Vygotsky, L. (2003). *Pensamiento y Lenguaje*. Paidós Ibérica, Buenos Aires, Argentina.

[Wallace, 2000] Wallace, R. (2000). Artificial linguistic internet computer entity (a.l.i.c.e.).

[Wang, 2002] Wang, K. (2002). Salt: a spoken language interface for web-based multimodal dialog systems. In *7th International Conference on Spoken Language Processing (ICSLP-2002)*, pages 2241–2244.

[Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

[Winograd, 1972] Winograd, T. (1972). *Understanding Natural Language*. Academic Pr.