Universidad Zaragoza
1542

Trabajo Fin de Grado

«Órdenes condicionadas»: Una propuesta lingüística para un nuevo modelo de oraciones condicionales destinada a los lenguajes de programación.

Autor/es

Miguel López Otal

Director/es

Mª del Carmen Horno Chéliz

Facultad de Filosofía y Letras.

2020

ÍNDICE

CAPÍTULO I: INTRODUCCIÓN

A lo largo de la historia, hemos sido capaces de idear numerosas invenciones técnicas que han facilitado nuestra vida. Desde los primeros utensilios para cultivar la tierra, pasando por las vacunas y otros avances médicos, hasta los modernos ordenadores, la inteligencia humana ha creado todo tipo de soluciones a los problemas que nos han rodeado desde los comienzos de nuestra especie.

Nuestra mente, sin embargo, no es una máquina creativa capaz de encontrar soluciones prácticas *ex nihilo*: siempre ha partido de la observación de la naturaleza que nos rodea, que le ha servido como fuente de inspiración. La gran versatilidad del ser humano es, de hecho, saber reconocer los puntos fuertes de los sistemas que observa en lo natural e intentar replicarlos para sus propios fines. Esto es lo que ocurre, por ejemplo, con algunos inventos de Leonardo Da Vinci, como sus prototipos de aviones que se asemejan a las alas de los pájaros; o con la invención de las primeras resinas para proteger los muebles de madera de ácaros o termitas, que surgieron de observar cómo, en troncos de árboles sumamente dañados por estos, ciertos nodos se mantenían intactos (precisamente porque tenían dicha resina de manera natural); o, de manera más reciente, con los sistemas de rádar, que han replicado (de manera más o menos consciente) la manera de trabajar de la de la ecolocalización de las ballenas jorobadas y de los murciélagos. En definitiva, podemos decir que la retroalimentación entre las ideas aportadas por el funcionamiento de los sistemas biológicos y nuestras soluciones «artificiales» ha sido constante en la historia de la tecnología.

De todos los sistemas naturales que pueden haber servido de base al ser humano para sus invenciones, vamos a centrarnos en uno que, al parecer, es exclusivo y definitorio de nuestra especie: se trata de nuestra capacidad para el lenguaje articulado. De acuerdo con Mendívil 2012, las lenguas son sistemas con un componente dual biológico-cultural[1]. Por un lado, los seres humanos tenemos la capacidad cognitiva de aprender y usar lenguas naturales. Esta capacidad (que Chomsky, décadas atrás, denominó «Facultad del Lenguaje») es de naturaleza cognitiva (biológica, por tanto) y determina los límites de la diversidad de las lenguas, a la par que es una propiedad común a toda nuestra especie. Por otra parte, sobre dicha base biológica habría todo un poso cultural, en constante evolución,

[1] Su definición como sistema biológico ha sido siempre objeto de dudas (tal y como se ve expuesto en Mendívil 2018), pero en el contexto de nuestro trabajo lo consideraremos no obstante como tal.

que sería el que configuraría las distintas lenguas que hablamos, como el inglés o el japonés (que, sin embargo, como podemos ver, parten de una misma base biológica). Esta combinación biología-cultura estaría presente en cada individuo de la especie humana, y sería lo que se denominaría en el terreno de la Biolingüística como *lengua-i*. Con esta visión, defendemos la idea de que el lenguaje es un sistema biológico (aunque con un componente cultural cuanto menos importante).

El lenguaje, como sistema natural, presenta interesantísimas propiedades que, si son bien estudiadas, podrían ser replicadas para todo tipo de invenciones que mejorarían nuestro día a día. En este trabajo nos gustaría recalcar la importancia del componente biológico del lenguaje (y de la disciplina que lo estudia, la Biolingüística) para poder inspirar soluciones en un terreno que, *a priori*, parece rechazar toda intervención que se aleje de lo meramente técnico o matemático: hablamos del mundo de la Informática. Somos conscientes de que, de manera general, parece que hay una marcada y arbitraria dicotomía entre las Humanidades (donde residiría, al parecer, el estudio del Lenguaje) y la Informática; hasta tal punto que la colaboración entre ambos terrenos parece imposible. Toda aportación teórica al mundo de los ordenadores por parte de humanistas corre el riesgo de ser rechazada por sus expertos como «poco precisa en lo técnico» o «irrelevante» para sus avances. No obstante, en la práctica esto no ha sido, en absoluto, cierto: no solo existe un floreciente campo en el procesamiento del lenguaje natural en los ordenadores (que alimenta la tecnología detrás de los motores de búsqueda y la Inteligencia Artificial, entre otros, y donde el conocimiento lingüístico es imprescindible), sino que además la propia base de la Informática debe en gran parte su existencia al terreno del lenguaje.

A día de hoy, casi todos los programas que manejan las computadoras están escritos en lenguajes de programación, una lengua especial creada específicamente para comunicarse con las máquinas y que, en muchos casos, es una imitación de las lenguas humanas (para así facilitar la intercomprensión hombre-máquina). Estos lenguajes de programación, que imitan el inglés (pues surgieron principalmente en EE. UU.), adolecen del fallo de haber sido creados por ingenieros que tenían un conocimiento intuitivo pero no preciso de los mecanismos del lenguaje humano. Por este motivo, muchas veces pueden tener ciertos problemas de naturalidad o falta de expresividad.

En este trabajo presentaremos una solución, desde el estudio de la Biolingüística, que busca resolver un problema técnico asociado a una de las estructuras más comunes en estos lenguajes de programación: las llamadas *oraciones condicionales*. Comenzaremos, en el

capítulo II, con una descripción de dos mecanismos del lenguaje natural (las órdenes –entendidas como *actos de habla directivos*– y las oraciones condicionales). Estos serán interesantes para resolver un problema de los lenguajes de programación, expuesto en el capítulo III. Finalmente, en el capítulo IV planteamos una solución a ese problema técnico, aplicando el conocimiento presentado en el capítulo II. El trabajo terminará con unas conclusiones y la bibliografía consultada.

CAPÍTULO II: LOS ACTOS DE HABLA DIRECTIVOS, LAS ORACIONES CONDICIONALES Y LAS «ÓRDENES CONDICIONADAS» EN LAS LENGUAS NATURALES

En este capítulo, nuestro objetivo es explicar las denominadas «órdenes condicionadas», que expresan la noción de «orden que está sujeta a que sucedan unas condiciones determinadas». Para ello, hemos considerado oportuno hablar por separado de dos estructuras básicas en las lenguas humanas: en primer lugar, comenzaremos con una presentación, desde la Semántica y la Sintaxis, de cómo los hablantes damos órdenes a nuestros interlocutores; en segundo lugar, pasaremos a describir la tipología de las oraciones condicionales en español. El capítulo terminará con una tipología de esas «órdenes condicionadas».

1. Los actos de habla directivos

Hablemos en primer lugar de la tipología de los llamados *actos de habla directivos*. En el lenguaje cotidiano, cuando queremos que una persona haga algo por nosotros, se lo comunicamos a nuestro interlocutor por medio de unos enunciados especiales que realizan ese cometido. Dichos enunciados reciben el nombre de *actos de habla directivos*, y son un subtipo de los *actos de habla* (Austin, 1975; Searle & Searle, 1986[2]). Los *directivos* son aquellos actos de habla en los que, al pronunciar unas palabras, buscamos que nuestro interlocutor haga una acción por nosotros (en otras palabras, se trata de una *orden*). Es un acto de habla orientado hacia nuestro interlocutor y que trata sobre acciones. Las órdenes, entendidas como actos de habla directivos, solo pueden tener efecto cuando las condiciones pragmáticas que las rodean son las adecuadas[3]. Esto se relaciona con el hecho de que, en el habla cotidiana, al dar órdenes no esperamos que estas surtan efecto en todas las

[2] Entender los enunciados como *actos de habla* supone entender que, cuando hablamos, en ocasiones, además de hacer una mera descripción o aseveración objetiva de la realidad se realizan acciones concretas. Estos *actos de habla* están sujetos a que sucedan una serie de condiciones pragmáticas concretas que, si y solo si se cumplen, permitirán que el acto de habla sea exitoso. En eso se parecen a los denominados Enunciados o Predicados Realizativos como *Yo os declaro marido y mujer*, que solo puede surtir efecto si esas palabras son pronunciadas por un oficiante en una ceremonia nupcial; si cualquiera de esas condiciones no se dan, el acto de habla fallará.

[3] Requieren que sucedan esas condiciones pragmáticas, igual que ocurre con el resto de tipos de actos de habla.

circunstancias posibles[4]. Estas condiciones pragmáticas, que determinan el que una orden pueda o no ser llevada a cabo por otra persona, reciben el nombre de *condiciones de éxito* y, como su propio nombre indica, determinan el *éxito* (mayor o menor) de una orden para ser cumplida[5]. Una de esas *condiciones de éxito*, que es además esencial en estos actos de habla, es la de respetar la jerarquía entre emisor e interlocutor. Esta información debe ser explicitada lingüísticamente en el discurso (ya sea en la propia oración con el acto de habla o en otra ubicada en otro punto de la conversación; Garrido Medina, 1999). Los *actos de habla directivos* (y, en general, todos los tipos de *actos de habla*) no se sujetan, por otro lado, a una propiedad semántica que sí se da en enunciados descriptivos: en estos últimos, los hablantes pueden (y suelen) intentar verificar si lo dicho en ellos es verdadero o falso; para ello, comparan lo que se afirma en el enunciado con la realidad del contexto comunicativo que les rodea, viendo a ver si lo dicho se ajusta o no a lo que en esta realidad palpable se observa. Los rasgos que permiten verificar la verdad (o falsedad) de un enunciado se llaman *condiciones de verdad* (hablaremos más sobre ellas en párrafos posteriores), y son una propiedad inherente a muchos enunciados. Los *actos de habla directivos*, sin embargo, por su semántica, no tienen *condiciones de verdad*, pues una orden no puede juzgarse como verdadera o falsa. En su lugar, solo se puede determinar si una orden tiene o no éxito en su realización, de allí que se hable de *condiciones de éxito*.

Los actos de habla directivos pueden construirse de una multitud de formas. Existen para ello, como ocurre en otros actos de habla, procedimientos tanto léxicos como oracionales[6]. En los oracionales, tenemos fórmulas realizativas[7] del tipo «Te ordeno», «Te solicito»… No obstante, aquí nos va a interesar principalmente un procedimiento de tipo léxico: el conjugar los verbos con modo imperativo. Este es un modo verbal que presenta

---

[4] Sabemos, por ejemplo, que si damos una orden a una persona desconocida es más que probable que este individuo decida no cumplirla.

[5] En la teoría general de los Actos de Habla, estas condiciones de éxito son fundamentales: en el clásico ejemplo de «Yo os declaro marido y mujer», sus condiciones son que las palabras estén pronunciadas en un contexto pragmático adecuado (ser su emisor un cura, en una iglesia…); de lo contrario, no se cumplirá.

[6] También hay formas que lo son por pragmática: una forma como «¿Podrías abrir la ventana?» no es una mera pregunta, sino una orden indirecta cuyo significado está determinado por la pragmática de la oración.

[7] Se trata de verbos y fórmulas fijas que, por su propia semántica, están sujetos a *condiciones de éxito* y no de *verdad*, de allí que puedan ser utilizados para crear Actos de Habla. Esto incluye fórmulas como «Yo prometo…», en la que con la mera presencia del verbo «prometer» ya expresa el contenido semántico de una promesa.

una forma sintética[8], y cuya conjugación es más reducida que su equivalente en indicativo o subjuntivo: solo cuenta con dos personas verbales[9] (2ª persona del singular y del plural[10]).

(Tú) Ven
(Vosotros) Venid[11]

Hay ciertas formas en modo subjuntivo (como la subordinada sustantiva en la oración «Nos dice *que vengamos*») tiene también un carácter imperativo, si bien no pertenecen propiamente al paradigma. La cuestión es controvertida en este sentido[12], pero no obstante el subjuntivo sí que se usa notablemente para construir la negación de las formas en imperativo («No vengas» frente a «Ven»[13]). Además, el imperativo tampoco se puede usar en las cláusulas subordinadas, donde deben ser también suplidas por una forma en subjuntivo («Dile que venga» frente a «Ven»).

Las formas en imperativo, semánticamente, se asemejan a las formas exhortativas y tiene un carácter apelativo (como las interrogativas; *Nueva Gramática*, 2009). Este modo, *a priori*, no tiene tiempo verbal: si bien a veces se le asocia con el presente de indicativo, realmente es «atemporal» (quizá con un cierto valor de «acción a ser realizada en el futuro[14]»), de allí que Garrido Medina (1999) nos diga que el imperativo «alterna con futuros de indicativo y formas del subjuntivo en la expresión de órdenes y peticiones» (pág. 3910). Por otro lado, las formas en imperativo no tienen un sujeto agente, en el sentido de aquel referido al emisor de la oración, sino que el sujeto que pueda existir en ellas se refiere al interlocutor, a la persona que se desea que haga la acción. Al ser el imperativo, por otro lado, el modo preferente para construir actos de habla directivos, posee un rasgo en común

---

8 Frente a una construcción de tipo analítica, como «has venido», una forma sintética es aquella compuesta por una sola palabra, como «vienes».

9 Respecto a la ausencia de tiempos verbales, se debe recalcar que, por ejemplo, no tiene sentido que existan formas en imperativo en 1ª persona. Tal y como dice Garrido Medina (1999) citando a Bosque: «no puede uno darse una orden a sí mismo» (pág. 3916).

10 Adviértase de estas dos formas aquí arriba expresadas que el imperativo solo puede expresar sintáctica y semánticamente órdenes dadas a una 2ª persona, ya sea del singular o del plural, además de mandatos que sean de carácter afirmativo (las órdenes negativas se expresan en modo subjuntivo: «No vengas»).

11 La Nueva Gramática hace referencia a ciertas formas que podrían formar parte del paradigma del imperativo, si bien su inclusión es controvertida: entre ellas, destacamos la 1ª persona del plural de presente de indicativo («vayamos»).

12 Garrido Medina (1999) establece que la confusión entre las formas de imperativo, subjuntivo e infinitivo (forma que no hemos tratado aquí por brevedad expositiva) para expresar órdenes se daba ya en latín.

13 No se puede construir una negación de una forma en imperativo con la mera anteposición de un adverbio de negación: «*No ven». En su lugar, se tiene que recurrir al subjuntivo.

14 Según Garrido Medina (1999): el imperativo «se refiere a acciones que no han tenido lugar ni están teniendo lugar» (pág. 3910); es decir, tiene inherentemente un cierto valor de futuro.

con estos: no están sujetos a *condiciones de verdad*, sino de *éxito*[15]. También en consonancia con estos, debe expresarse la jerarquía que existe entre emisor e interlocutor (esta es, como hemos comentado arriba, una *condición de éxito* del imperativo). Tal y como establece Garrido Medina (1999): «La oración imperativa introduce […] la representación del oyente y del hablante, y presenta explícitamente la relación de que el hablante le solicita al oyente la realización de la acción» (pág. 3918); es decir, esa relación hablante-oyente debe estar formalizada en una oración imperativa.

2. Las oraciones condicionales

Hagamos ahora una breve digresión para hablar sobre la oración condicional en castellano. Esta consiste en una construcción sintáctica que permite expresar la noción de que lo dicho en el enunciado es verdadero solo si se cumplen una serie de condiciones a las que está sujeto. Presenta habitualmente la forma de una oración compleja, compuesta por dos oraciones simples, que reciben respectivamente el nombre de *prótasis* y *apódosis*. La prótasis, que suele estar encabezada por la conjunción condicional «si»[16], nos dice cuál es la situación que está condicionada; es decir, aquello que de cuyo cumplimiento depende el que lo que expresa la apódosis (la segunda parte de la oración condicional) sea o no cierto:

«Si llueve, → Prótasis
me cojo un paraguas» → Apódosis

En el ejemplo de arriba, la situación de coger un paraguas depende en exclusiva de que llueva. Si no llega a cumplirse esa condición, el sujeto, en principio, no cogerá un paraguas (aunque sí que podría llegar a suceder, no obstante). Esta manera de construir las oraciones condicionales es típica en lenguas como el español y el inglés (con la conjunción «*if*»), aunque no es la única. Tal y como establece Montolío (1999):

> «La denominación 'construcciones condicionales' constituye la etiqueta unificadora bajo la que se engloba un nutrido conjunto de estructuras sintácticas notablemente diferentes entre sí[17]» (pág. 3647)

[15] No tiene sentido responder, ante una orden como «Hazte la cama», con un «¡Eso es falso!». Más bien, esta orden tiene que atenerse a si, de acuerdo con sus condiciones de éxito, será o no realizada.

[16] Pueden existir otras muchas conjunciones condicionales, aunque con diferente contenido semántico, como «cuando», que marca temporalidad a la par que condicionalidad.

[17] Zaefferer (1989) alude al hecho de que, en ciertas lenguas, existen otros mecanismos alternativos para marcar la condicionalidad, tales como afijos morfológicos específicos o la presencia de ciertos elementos

En una oración condicional en español se da una relación de *entrañamiento*[18] (concepto de semántica lógica) entre prótasis y apódosis: de la verdad de la prótasis se deduce la verdad de la apódosis[19]. La semántica que rodea al hecho de que una prótasis pueda o no ser verdadera es una cuestión compleja, hecho que no impedirá sin embargo que abordemos su explicación.

Para ello, necesitamos hacer una explicación desde los principios de la Semántica Composicional[20]. Presentaremos, en primer lugar, el concepto de *Mundo Posible*. En Semántica, cuando queremos valorar si un determinado argumento (por ejemplo, «Hoy es Navidad») es verdadero o es falso, podemos remitirnos al momento presente o actual y comprobar si, en efecto, las condiciones que se articulan en la actualidad afirman (o desmienten) lo dicho en el enunciado. Dicha organización de condiciones que permiten demostrar la verdad (o falsedad) de un argumento se denomina «Modelo», y define un conjunto de propiedades que están vinculadas con el mundo real. Esta vinculación entre modelo y realidad se denomina *extensión*. Así pues, por ejemplo, en la oración «Hoy es Navidad» podemos afirmar que lo dicho es falso puesto que, en nuestro conocimiento del mundo, dicho día se celebra el 25 de diciembre y, por el contrario, nos encontramos en una fecha distinta[21]. No obstante, esto no quiere decir que un enunciado no pueda ser verdadero en otros contextos. Podemos perfectamente intentar demostrar la verdad o falsedad de un argumento a partir de otros momentos diferentes del actual, no solo en el plano temporal, sino también en el locativo, de realidad (si se trata de ficción), etc. Es aquí donde entran los *Mundos Posibles*: se trata de *modelos* alternativos, distintos de los del momento actual, que describen un estado de cosas diferente. Cada Mundo Posible, por el fenómeno de la extensión, está relacionado con un conjunto de propiedades de la realidad, pero no necesariamente con las del momento presente o incluso con las de nuestra realidad.[22] Un determinado argumento puede ser falso en el momento presente pero, cuando está asociado

---

léxicos (o construcciones perifrásticas) que indican condición; al igual que el posible uso de un orden concreto de palabras.

[18] Relación extraída de la Lógica: establece que, en la relación entre dos argumentos, *A* y *B*, *A* entraña a *B* si de la verdad de la primera se da también la verdad de la segunda. No obstante, si *A* es falsa, *B* no tiene por qué ser falsa (puede ser verdadera).

[19] Nota: la falsedad de una prótasis no implica necesariamente que la apódosis tenga que ser también falsa (puede ser verdadera al margen de que la prótasis sea falsa). La relación lógica de entrañamiento tiene en cuenta precisamente este hecho, circunstancia referida en la nota al pie anterior.

[20] Disciplina encargada de estudiar la Semántica de las construcciones sintácticas.

[21] Nota al lector: este argumento puede perder su validez si, en efecto, estas líneas son leídas entre turrones y comidas familiares.

[22] Un universo de ficción como la Tierra Media en *El señor de los anillos* constituye un Mundo Posible, con sus propiedades de verdad relacionadas con dicho mundo irreal.

a un mundo posible determinado, ser verdadero. De nuevo con el ejemplo de «Hoy es navidad», podemos afirmar que si dicho enunciado lo asociamos a un mundo posible en el que, entre otras cosas, la fecha es la del 25 de diciembre, el argumento será cierto. Estos ejemplos no se limitan a la temporalidad, sino que un mundo posible puede perfectamente hablar de un modelo en el que se den todo tipo de situaciones hipotéticas e incluso completamente irreales. Por ejemplo, en:

«Sócrates puso su remix de dubstep en la recepción oficial de Erich Honecker»

Hablamos de un modelo o mundo posible que, *per se*, es completamente ilógico, pues en él el dignatario de la Alemania del Este Erich Honecker y el filósofo griego clásico Sócrates conviven en un mismo período histórico, en el que además también está de moda el género musical del *dubstep*. Esta situación, más que ser hipotética, es completamente irreal e insostenible. La verdad de este argumento quedaría completamente negada en la mayor parte de los casos, pero existe la remota (aunque no por ello implanteable) posibilidad de que haya un mundo posible (aunque sea en el terreno de una ficción literaria político-filosófica) en el que sí se den esos hechos y permitan afirmar la validez de lo argumentado.

Esta semántica de los Mundos Posibles es perfectamente aplicable al análisis de las prótasis de las oraciones condicionales. Este modelo puede servir para explicar de qué manera se puede determinar, a nivel de lo semántico, la verdad o falsedad de una prótasis. Esto se debe a que esta construcción sintáctica, desde el momento en el que está encabezada por una conjunción «si», está refiriéndose a una situación o evento que se localiza en un plano irreal o hipotético, alejado del momento presente[23]. Por este motivo, para demostrar la verdad de esta no se puede recurrir a comprobarla con las circunstancias del momento en el que se enuncia[24], sino a aquellas que se encontrarían en dicho plano irreal en el que se inscribe. Ese plano irreal correspondería, en Semántica Composicional,

[23] Escandell (2004) nos habla de construcciones intensionales, un tipo de estructuras sintácticas cuyo cometido es el de poner al interlocutor en un plano diferente de aquel del momento actual. Una oración condicional sería una construcción de este tipo, pues ubica al interlocutor en una situación irreal e hipotética.

[24] Según Montolío (1999), hay una creencia generalizada que, cuando se intenta probar la verdad de una prótasis, lo que se hace es intentar analizar el grado de veracidad de lo dicho con respecto a la situación real que es referida por la prótasis. Esto en realidad es una falacia pues, si bien es una consideración que parece válida para la gran mayoría de las oraciones condicionales, no sirve para su gran totalidad, puesto que sigue habiendo un pequeño grupo de ellas en las que este hecho no procede. Por este motivo, Montolío establece que la relación de entrañamiento que existe entre prótasis y apódosis no se da «entre dos fenómenos existenciales, sino entre dos actos de habla»; visión que vemos compatible con la noción de Mundo Posible (un Mundo Posible que estaría vinculado a la prótasis y de la que dependería su verdad o falsedad).

con el de un Mundo Posible. De este modo, bajo el paraguas de esta teoría, toda prótasis de una oración condicional se vincularía con su propio Mundo Posible, que determinaría la verdad de lo argumentado. El conjunto extensional de propiedades de ese Mundo Posible, que demostrarían el grado de verdad de la prótasis con la que va vinculado, recibe el nombre de *condiciones de verdad* (a las que nos referimos en párrafos anteriores, en contraposición con las *condiciones de éxito*).

3. Las órdenes condicionadas

Hemos hablado hasta el momento de dos construcciones sintácticas y semánticas de gran interés: los *actos de habla directivos* (con el modo imperativo en cabeza) y las oraciones condicionales. Nuestro interés en presentarlas de manera independiente y en describirlas con relativa profundidad radica en el hecho de que ambas se fusionan en una circunstancia concreta, cuando quieren expresar un contenido semántico específico, que es el que nos interesa especialmente para este trabajo. Esta fusión sucede por una limitación inherente a los actos de habla directivos. El imperativo, como forma preferida para construir estos actos de habla, sirve a la perfección para expresar órdenes que queremos que nuestro interlocutor realice inmediatamente; sin embargo, esta forma falla a la hora de expresar la siguiente noción semántica: el conseguir que el interlocutor realice esa acción no ahora mismo, sino en otro momento y, más concretamente, si (y solo si) se da una determinada condición. Podríamos dar a este tipo de órdenes el nombre de «órdenes condicionadas» (puesto que están sujetas a que ocurra una determinada condición). La Nueva Gramática de la Lengua Española nos establece que, en el caso de querer expresar el deseo de que alguien realice una acción en un tiempo distinto del actual, es común hacer uso de adverbios de tiempo (u otras formas temporales) que acompañen a la forma en imperativo: «Ven mañana» o «Entrega esta carta el 5 de enero»[25]. Aunque el uso de estas formas temporales logra romper, al menos parcialmente, el carácter de inmediatez que es inherente al imperativo, no es un mecanismo suficiente para lograr ir más allá y poder dotar de un sentido de condicionalidad a las órdenes.

[25] Esta misma obra alude a la existencia de formas contrafactuales como «Haber estudiado», que no son propiamente verbos en imperativo pero que tienen su misma carga semántica. No las consideraremos aquí por la dificultad en su clasificación y porque, además, realmente solo tienen un valor de tipo temporal (referido además a un tiempo pasado).

Hemos de adelantar que no existe ninguna forma verbal sintética que exprese ese concepto de «órdenes condicionadas»[26]: por este motivo, la lengua ha recurrido a tomar el modo imperativo y «enriquecerlo» por medio de un añadido lingüístico[27], cuya presencia junto al imperativo permite expresar ese contenido semántico. En español, esta construcción consiste en una oración condicional (véase, prótasis) que acompaña a una forma verbal en imperativo (que sería su apódosis). Sería del siguiente modo:

«Si llueve, ← Prótasis
coge un paraguas» ← Apódosis (en imperativo)

Esta prótasis transmite a la forma en imperativo su noción semántica de Mundo Posible, aquella que está asociada a las prótasis de las oraciones condicionales. Podemos explicar, en este sentido, que en lo semántico (y olvidando por el momento la estructura sintáctica de la prótasis) una orden condicionada consistiría en una forma verbal en imperativo cuyo cumplimiento solo es factible si se ubica en un Mundo Posible que satisfaga sus exigencias. Frente a otro tipo de enunciados, sin embargo, el cumplimiento de lo dicho en esta forma en imperativo no está sujeta a unas *condiciones de verdad*, como sí lo están otro tipo de oraciones condicionales. Si bien es cierto que la apódosis en imperativo puede ocurrir solo y solo si se da la verdad de la prótasis (verdad que es demostrable por el Mundo Posible que tiene a ella ligado), la verdad de la prótasis no puede conducir a la verdad de la apódosis, por el simple hecho de que, semánticamente, es imposible que una forma en imperativo sea verdadera o falsa. De un imperativo solo se puede decir si ha tenido o no ha tenido éxito (recuérdese la explicación anterior al respecto de los actos de habla directivos). Por tanto, en las «órdenes condicionadas» tendremos que hablar de *condiciones de éxito* en lugar de *condiciones de verdad*. Esto se debe al hecho de que una orden, como ya hemos comentado, es un tipo de *acto de habla* (teoría de Austin y Searle), por lo que no cabe en ellos una descripción aseverativa de sus palabras sino que en ellas tienen más importancia las condiciones de éxito que conducen a que el acto descrito pueda o no ser realizado[28].

---

[26] El latín contaba con una forma morfológica de imperativo con valor de futuro, si bien solo expresaba ese valor temporal además del de orden. Además, se perdió en su paso al castellano.

[27] Si se nos permite esa expresión.

[28] Las «órdenes condicionadas», por tanto, no se ajustan a la teoría general de las oraciones condicionales, pues, como habíamos comentado antes, lo importante en estas formas es realmente la forma en imperativo, mientras que la prótasis que lo acompaña no es más que una especie de armazón sintáctico que expresa un contenido semántico adicional sobre dicha orden y la pone en un plano diferente del actual.

Vamos ahora a hablar de una propiedad sintáctica y semántica muy importante en las oraciones condicionales, que tiene también su reflejo en las «órdenes condicionadas». Los enunciados condicionales pueden mostrar, en lo sintáctico, variación de tiempo y modo de sus verbos en la prótasis y la apódosis. Ello responde a la necesidad semántica de expresar el siguiente contenido: la probabilidad (mayor o menor) de que pueda materializarse lo expresado en la prótasis. Existen, en este sentido, tres tipos principales de oraciones condicionales en español:

- Condicionales reales (tipo 1): expresan condiciones que, siempre que se cumplan, conducen a que ocurra una determinada circunstancia. Suele tratarse de máximas o verdades absolutas, y tanto su prótasis como su apódosis tienen verbos conjugados en modo indicativo.
    - «Si llueve, se moja el suelo»
    - «Si cantaba, todos se reunían en torno a él»[29]
- Condicionales hipotéticas (tipo 2): en este tipo, la prótasis expresa una situación que, aunque es improbable que ocurra a primera vista o en primera instancia, siempre hay una posibilidad de que pueda suceder. Tanto prótasis como apódosis presentan verbos en modo subjuntivo.
    - «Si me tocara la lotería, me compraría una casa grande»
- Condicionales irreales (tipo 3): también llamadas *contrafactuales* (Montolío, 1999), este tipo de condicionales expresan situaciones que es imposible que ocurran, ya por su irrealidad o porque ha pasado ya el momento en el que podrían haberse cumplido. Los hablantes suelen usarlas, con carácter retrospectivo, para lamentarse de situaciones que no pueden ser ya resueltas. Presentan en su prótasis y apódosis tiempos verbales en subjuntivo, pero con formas compuestas en la prótasis.
    - «Si hubiera sacado bien el curso, no habría suspendido»

De estos tres tipos de condicionales solo dos de ellas, las reales (tipo 1) y las hipotéticas (tipo 2), pueden ser aplicadas a las «órdenes condicionadas». El matiz semántico que estas prótasis aportan a la forma en imperativo a la que acompañan es el de

[29] Este ejemplo está basado en el pasado, pero tiene la misma naturaleza semántica que la oración condicional en presente de indicativo. De este modo, aunque la condición expresada en esta oración no puede ya cumplirse en el momento actual, en tiempos previos siempre que ocurría se cumplía lo dicho en la apódosis.

probabilidad, probabilidad de que pueda (o no) expedirse una orden a alguien. El hecho de que las condicionales de tipo 3, las «contrafactuales», no puedan utilizarse en las «órdenes condicionadas» es por una mera cuestión de significado: no tiene ningún sentido dar una orden cuyo punto de referencia es un contexto que nunca va a suceder[30]. De este modo, oraciones del tipo «*Si hubiera llovido, recoge la ropa» son, por este motivo, agramaticales.

Adviértase que, a pesar de que las prótasis presentan formas verbales distintas, el imperativo de la apódosis se mantiene con ambos tipos de condicional igual. Así lo vemos en este esquema, en el que además detallamos qué significado aporta el tipo de prótasis a cada una de ellas:

- «Si te llama el director, atiende a lo que te diga» → (Alta probabilidad de que se cumpla la condición y, por tanto, que se lleve a cabo la orden)
- «Si llegara la pandemia al pueblo, aléjate de ahí» → (Menor posibilidad, aunque no imposible)

Finalmente, querríamos hablar aquí de las oraciones condicionales cuyas prótasis son complejas. Podemos encontrarnos ante «órdenes condicionadas» cuyo cumplimiento está sujeto no solo a una condición sino a dos o más:

«Si el niño se porta bien y hace los deberes, dale golosinas»

En la oración de arriba, la orden en la apódosis solo puede llegar a ser efectiva si se cumple no solo una sino dos condiciones de la prótasis: que el niño se comporte bien y que haga los deberes. El incumplimiento de cualquiera de esas premisas impediría que el niño recibiera su recompensa. Otro caso similar es el siguiente:

«Si el niño hace deporte o termina los deberes, déjale jugar a la consola»

Esta oración condicional también cuenta con dos condiciones en su prótasis, pero su contenido semántico es algo distinto: da igual que se cumpla solo una de las condiciones y la otra no (aunque si las dos se cumplieran, incluso mejor); el resultado de que una sea exitosa es que el niño logrará el premio.

Estos dos tipos de oraciones que hemos expresado arriba son, tipológicamente, oraciones condicionales complejas, y aquí las estamos viendo desde la perspectiva de las

[30] En términos de Semántica Composicional y de la teoría de los *Actos de Habla*, esta forma en imperativo no puede probar sus condiciones de éxito en un Mundo Posible que es irreal.

«órdenes condicionadas». En lo sintáctico, el análisis de este tipo de prótasis puede resultar algo complejo[31] y controvertido.

No obstante, su caracterización desde el punto de vista semántico puede resultar un poco más accesible. Hemos de tratar cada una de las condiciones que integran esa prótasis compuesta como ligada a un Mundo Posible distinto, individual para cada una de ellas. Esa prótasis constituiría, por tanto, un conjunto de *n* mundos posibles. Es esencial conocer, por otro lado, cuál es la relación sintáctica que se establece entre esas condiciones de la prótasis: si hay conjunción copulativa o disyunción. Pueden darse casos en los que incluso se junten ambos tipos de relaciones sintácticas en una misma prótasis: «Si el niño hace los deberes y riega las plantas, o si hace algo de ejercicio, déjale ver la televisión[32]».

La ligazón sintáctica que se da entre esas condiciones de la prótasis establece, en el terreno de la Semántica, el tipo de relación lógica que existirá entre los Mundos Posibles individuales de esa prótasis. Si la unión es copulativa, la relación será de *Intersección de conjuntos*; si no, será de *Disyunción*.

Estos dos son conceptos que provienen de la *Teoría de Conjuntos*, una rama de las Matemáticas y de la Lógica que estudia los *conjuntos* (colecciones de elementos, *a priori*, del mismo tipo) y las relaciones que se establecen entre ellos. Aplicado a este contexto de lingüística (y considerando cada Mundo Posible de la prótasis como un *conjunto extensional* de propiedades e individuos), podemos aplicar esa teoría de conjuntos al análisis de cómo se vinculan en lo semántico esos conjuntos extensionales de significado entre sí.

Primero, hablaremos de qué ocurre cuando hay Intersección de conjuntos. En este caso, esa intersección se aplica a aquella situación en la que una prótasis expresa dos (o más) condiciones que tienen que ser verdaderas para que conduzca (o *entrañe*) a la verdad de la apódosis. Aquí, cada uno de los mundos posibles que conforman la prótasis son, individualmente, un conjunto extensional, como hemos dicho más arriba. La Intersección de conjuntos se da cuando, en dos o más conjuntos, algunos elementos que pertenecen a un conjunto están también presentes en los otros conjuntos (ello no implica que los conjuntos

---

[31] Y pueden complicarse aún más, porque haciendo uso del mecanismo de la recursión en el lenguaje podemos potencialmente añadir más y más condiciones a la prótasis, *ad infinitum*: «Si el niño hace los deberes, y come sano, y hace ejercicio, y va a visitar a su abuela, y se porta bien con los otros niños, y no dice palabrotas, y se limpia los zapatos, y hace su cama, y ordena su habitación, y ayuda a sus padres, y comparte sus juguetes…, entonces (y solo entonces) el pobre niño podrá jugar a la consola».

[32] Prótasis donde, para que el niño vea la televisión, puede hacer un poco de ejercicio, o hacer los deberes, pero donde si no riega las plantas entonces no podrá tener ese momento de ocio.

compartan entre sí todos sus elementos, ni mucho menos, sino solo un subconjunto común de ellos). Trasladado al terreno de las oraciones condicionales, desde la Semántica, la intersección entre esos mundos posibles (entendidos como *conjuntos extensionales*) se da cuando se comparten ciertas condiciones de verdad (condiciones de éxito, en las «órdenes condicionadas») entre varios de esos mundos. Por ejemplo, en la siguiente oración:

«Si Carlos prepara una tarta y María prepara un gazpacho, haremos la fiesta»

Hay un mundo posible que se vincula al quehacer diario de Carlos y otro al de María. Esos mundos posibles, vinculados a personas distintas, pueden contener muchas otras propiedades extensionales aparte de las mencionadas en la prótasis: Carlos, aparte de hacer la tarta, puede haber hecho yoga o escrito una novela ese mismo día; María, además de preparar el gazpacho, puede haber compuesto una sinfonía. Lo relevante en ambos casos, sin embargo, es que se dé esa intersección entre esos dos Mundos Posibles, el de Carlos y María, de donde entre todas las actividades que hacen esas personas pueda coincidir que haya en las dos al menos una que permita que se cumpla la consecuencia de celebrarse la fiesta.

Un poco distinto es el caso de una situación de Disyunción, aunque no es del todo diferente. Esta relación lógica se da en aquellas prótasis en las que la verdad (o éxito) de la apódosis se da a partir de la verdad de al menos uno de los Mundos Posibles expresados en la prótasis. Podemos poner el mismo ejemplo anterior para explicarlo, pero adaptado a esta otra situación:

«Si Carlos prepara una tarta o María prepara un gazpacho, haremos la fiesta»

En este caso, es posible que en el Mundo Posible de Carlos no esté la acción de «preparar una tarta»; no obstante, con tal de que en el de María esté el de «preparar un gazpacho», es suficiente para que se pueda hacer la fiesta[33]. De cada uno de esos Mundos Posibles, entendidos como conjuntos extensionales, es suficiente con que este demuestre la verdad del enunciado con el que está ligado. Aunque el resto de enunciados puedan ser falsos, el único enunciado verdadero da el conjunto de toda la prótasis como verdadero:

«Si vas al cine o al parque, te castigaré sin salir un mes»

[33] Por supuesto, si en esos Mundos Posibles no se da ni la acción de preparar una tarta ni un gazpacho, la fiesta no se celebrará.

CAPÍTULO III: EL PROBLEMA TÉCNICO

Tal y como anticipamos en la Introducción, una buena descripción de los sistemas biológicos puede ayudar a mejorar los problemas técnicos de nuestro día a día, y aquí concretamente en los lenguajes de programación. Si en el capítulo anterior hemos estudiado a fondo aquellas construcciones de la lengua natural a las que hemos denominado «órdenes condicionadas», en este capítulo vamos a aplicarlas en el terreno de la informática, para estudiar un problema técnico que existe en los lenguajes de programación. En el capítulo siguiente propondremos una manera de mejorar dicho problema técnico por medio de estas órdenes condicionadas.

En este capítulo, haremos primero una descripción general de los lenguajes de programación, para después hablar sobre cómo se aplica el concepto de «orden condicionada» en estos lenguajes. El capítulo acabará con el planteamiento de un problema técnico, surgido a causa de estas «órdenes condicionadas», y cuya solución plantearemos en un capítulo posterior.

1. Los lenguajes de programación

Nuestro problema tiene que ver con los llamados «lenguajes de programación». Estos son unos lenguajes formales, inventados por los ingenieros informáticos, que sirven para poder comunicarnos con los ordenadores y decirles qué tareas tienen que hacer. Esto es muy relevante para el funcionamiento de un ordenador, pues todo aquello que vemos ocurrir detrás de una pantalla sucede porque ha sido escrito, de antemano, en un lenguaje de programación. Por ejemplo, cuando buscamos un archivo en Windows, hay un código detrás, escrito en lenguaje de programación, que accede al disco duro y lo escanea de arriba abajo hasta encontrar el archivo con el nombre que hemos buscado. El hacer clic en un botón gráfico de búsqueda esconde toda una maquinaria detrás, que está escrita en un lenguaje de programación.

A pesar de que los lenguajes de programación son el medio por el que se crean los programas que usamos en un ordenador, en realidad la máquina no es intrínsecamente capaz de entender ese lenguaje (al menos, no como nosotros entendemos una lengua humana[34]). Un lenguaje de programación es, de hecho, un constructo artificial que sirve a los humanos para interactuar y operar un ordenador, pero lo que un ordenador en sí

[34] En un ordenador no hay un módulo en su interior dedicado a entender el inglés o cualquier otra lengua, como sí hay, por el contrario, una unidad de DVD o puertos USB.

entiende es el denominado *lenguaje máquina,* que es radicalmente distinto del nuestro. El *lenguaje de programación* es una especie de «punto medio» en la comunicación entre máquina y humano: el humano debe ceder parte de la expresividad de su lengua materna para comunicarse en un lenguaje formal que, entre otras características, cuenta con una gramática muy limitada (un rasgo imprescindible para la intercomprensión con la máquina), pero con suficientes similitudes con la lengua humana, a pesar de sus diferencias, como para que no resulte demasiado complejo para el humano. Para entender en qué consisten los lenguajes de programación y cómo se relacionan con el denominado lenguaje máquina, debemos conocer los orígenes del ordenador.

Las primeras máquinas autómatas (similares a un ordenador, pero aún lejanas a este) consistían en una serie de mecanismos que estaban especializados en realizar una única tarea, que no podía ser cambiada una vez se fijaban las especificaciones de los circuitos de la máquina. Si se deseaba hacer otra tarea distinta, se tenía que construir otra máquina[35]. La llegada de los primeros ordenadores fue una auténtica revolución, pues supuso la introducción de unos circuitos que, por la manera en la que estaban construidos, permitían perfectamente que pudieran ser reprogramados para toda clase de usos, distintos entre sí.

Estos ordenadores, que cambiaron de forma y tamaño a lo largo de los años, tienen en su seno unos circuitos llamados *procesadores* o *CPU.*[36] Estos son el «cerebro» del ordenador, la «batuta» que rige su funcionamiento. El CPU consigue hacer tareas diferentes siguiendo una serie de instrucciones que le son especificadas por un programador. Los circuitos internos del procesador están especializados en interpretar dichas instrucciones y, de acuerdo con ellas, ejecutar acciones específicas. Aunque solo ofrecen de base una serie de acciones muy sencillas (por ejemplo, sumar dos números o comparar varios valores), si son utilizadas en conjunto pueden construir una tarea útil y compleja. La grandísima utilidad que ofrecen estos procesadores se ve, sin embargo, contrarrestada por el extraño formato en el que sus instrucciones están codificadas: estas consisten, individualmente, en secuencias ordenadas de *bits*, números que toman el valor exclusivo de 0 o 1, con una

---

35 Un ejemplo es la radio, cuyos circuitos internos solo podían servir para recibir la señal AM/FM del aire y transformarla en sonido (una proeza en sí misma); en cambio, se necesitaba fabricar otros circuitos, distintos e independientes, para recibir la señal de televisión que venía por una antena e instruir a los rayos catódicos de una televisión CRT para que la transformara en una imagen visible.

36 Por sus siglas en inglés, *Central Processing Unit*; *Unidad de Procesamiento Central.*

extensión variable o fija de estos[37]. Cada secuencia única de esos dígitos binarios constituye una instrucción diferente, que realiza una acción básica distinta y conforma el *lenguaje máquina*.

Los programadores encargados de gestionar estas máquinas no tardaron en asignar etiquetas identificativas y concisas a cada una de estas instrucciones, con nombres ilustrativos de lo que hacían. De esta manera, si por ejemplo la instrucción para sumar dos números en un CPU de Intel era la secuencia de dígitos binarios 0000000000000101, se le atribuyó la etiqueta «*add*»[38] para que su escritura fuera más sencilla. Es lo que se conoce como *lenguaje ensamblador*.

Durante muchas décadas, la programación de ordenadores se hizo por medio de *lenguaje ensamblador*[39] (es decir, prácticamente en las instrucciones que el CPU comprendía de forma nativa). Sin embargo, en los años cincuenta del siglo XX esta situación empezó a cambiar. Por un lado, empezó a crecer exponencialmente la complejidad de los programas a escribir. Por otro lado, ya que cada procesador era distinto y tenía sus propias instrucciones y lenguaje ensamblador, ello obligaba a los programadores a reescribir sus aplicaciones para cada CPU distinto en el que quisieran ejecutarlo.

Por este motivo, se empezó a barajar la idea de escribir programas en un lenguaje (consensuado por algún tipo de comité) que imitara y se acercara a la expresividad de las lenguas naturales; concretamente, el inglés[40]. Una primera propuesta[41] vino de la mano de Grace Hopper, almirante en la Marina americana, quien propuso esta idea mientras supervisaba el Departamento de Informática de la Universidad de Harvard. Aunque inicialmente su sugerencia fue rechazada, con el tiempo empezó a tener gran aceptación, culminando en la creación de los primeros lenguajes de programación.

Para conseguir que la máquina «comprendiera» el inglés y ejecutara órdenes escritas en esa lengua, se crearon aplicaciones (en sí programadas en *lenguaje ensamblador*) cuyo

---

[37] Hay procesadores que tienen un número fijado de bits para sus instrucciones (común en la arquitectura RISC, como en los procesadores de ARM) y otros que varían su extensión dependiendo de la instrucción (propio de los procesadores CISC, como los de Intel). Este es un tema del que no hablaremos aquí.

[38] «*Add*» significa «sumar» en inglés.

[39] El lenguaje ensamblador, aunque ha sido desplazado en gran medida por los lenguajes de programación, no ha caído en desuso: sigue teniendo su margen de empleo, aunque solo para aplicaciones críticas. Una parte del sistema operativo Microsoft Windows, por ejemplo, está escrito en ensamblador.

[40] La revolución informática tuvo lugar, sobre todo, en EE. UU; de ahí que el inglés fuera la lengua elegida.

[41] Ya en el siglo XIX, antes incluso de la invención propiamente dicha de los ordenadores, Ada Lovelace Byron (hija del famoso poeta británico) propuso un sistema similar cuando trabajó en la máquina diferencial de Babbage, un precursor del ordenador actual que nunca llegó a fabricarse.

cometido era el de transformar, previo paso de un análisis de tipo textual, un texto escrito en ese lenguaje de programación en las instrucciones que entiende directamente el procesador. Estas aplicaciones reciben el nombre de *compilador*[42]. Dado que ese análisis textual sería una operación compleja (sobre todo para los lentos ordenadores de la época), se optó por dotar a estos lenguajes de programación, basados en el inglés, de una gramática limitada. También se procuró que esta gramática no permitiera introducir ningún tipo de ambigüedad, para que así hubiera una correlación exacta entre texto en este lenguaje e instrucciones del procesador.

Las aportaciones de Grace Hopper dieron lugar a algunos de los primeros lenguajes de programación, como COBOL o FORTRAN, cuyos descendientes indirectos son lenguajes actuales como C++, Java o Python.

Aparte de simplificar la programación, estos lenguajes aportaron una ventaja clave: de un programa que se escribía una sola vez, era perfectamente posible crear versiones de esa misma aplicación para que se ejecutara en múltiples ordenadores. Esto era factible porque, al existir un compilador de un lenguaje específico para dos tipos distintos de CPU, dicho compilador podía en cada caso convertir el programa al formato de procesador específico de esa máquina.[43]

Los lenguajes resultantes adquirieron, en su gran mayoría, un carácter imperativo: el programador se comunicaba con la máquina aportándole órdenes que tenía que hacer.[44]

2. Las órdenes condicionadas en los lenguajes de programación

Ahora, debemos hablar de la «gramática» de los lenguajes de programación, aquella que se utiliza para escribir programas. La gramática de estos lenguajes cuenta con una serie de construcciones sintácticas que se asemejan a las estructuras de las lenguas humanas. Los programadores utilizan estas construcciones para elaborar sus programas. Cada una de esas

---

[42] Gran parte de la construcción de compiladores se dio gracias al *input* de las primeras teorías del lenguaje de Noam Chomsky, que surgieron por esta misma época.

[43] Un ejemplo son las últimas versiones del sistema operativo *Microsoft Windows* (originalmente lanzado en 1993 bajo el nombre de *Windows NT*), que fue escrito en el lenguaje C, gracias a lo cual la empresa Microsoft es capaz de trasladar este sistema operativo tanto a un ordenador de sobremesa como a una tableta o teléfono inteligente. Si bien arriba, en otra nota al pie, hemos aludido al hecho de que parte de este sistema operativo está escrito en *lenguaje ensamblador* (lo que impediría *a priori* su traslado a otros procesadores), la compañía norteamericana mantiene al mínimo las partes del sistema escritas en él: de este modo, para trasladar Windows a otra plataforma, Microsoft solo tiene que compilar de nuevo el código en C de su sistema operativo y reescribir la mínima porción que está programada en *ensamblador*.

[44] En el capítulo anterior, hemos hablado del modo imperativo precisamente por esta cuestión. Aunque existen otros modelos de lenguajes de programación, como los declarativos, en este trabajo solo nos interesan los de tipo imperativo.

estructuras es una abstracción de algún mecanismo interno de la máquina. Una de esas construcciones, y la que nos va a interesar aquí, es la de las «oraciones condicionales».

Esta es una de las estructuras más importantes de los lenguajes de programación y es prácticamente un universal en todas ellas[45]. Estructuralmente, se escriben así:

```
if (condición)          → Prótasis
{
  //ÓRDENES
  A EJECUTAR//          → Apódosis
}
```

Esta estructura está compuesta por una prótasis (introducida por la conjunción «*if*», «si» en inglés) en la que su condición se expresa entre paréntesis. La apódosis viene a continuación englobada entre dos llaves[46] y consta de unas instrucciones que se desea ejecutar en el caso de que se cumpla la condición de la apódosis.

Si relacionamos esta estructura de programación con la lengua natural, en lo semántico estas oraciones condicionales son más similares a las «órdenes condicionadas» que hemos introducido en el capítulo anterior que a otro tipo de condicionales. De este modo, una «orden condicionada» como la siguiente:

«Si llueve, recoge las hamacas»

Se vería representada en un lenguaje de programación de la siguiente manera:

```
if (llueve)
{
  Recoge hamacas
}
```

Toda la explicación que hemos dado en el capítulo anterior sobre el imperativo y las posibilidades semánticas de las «órdenes condicionadas» no es casual: los lenguajes de programación son, a nivel general, lenguas imperativas, que utilizan la orden como su modo de expresión principal. Un programa escrito en un lenguaje como C++ o Java consistirá, fundamentalmente, en órdenes que se le dan al ordenador. Dichas órdenes, aunque son muy distintas morfológicamente a un imperativo en una lengua como el inglés o el español, codifican semánticamente el mismo concepto.

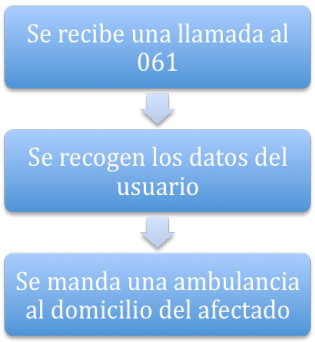
---

[45] En otras palabras, todo lenguaje de programación tiende a incluir en su repertorio sintáctico oraciones condicionales.

[46] En otros lenguajes de programación la apódosis puede no estar entre dos llaves; esto depende de la gramática de cada lenguaje.

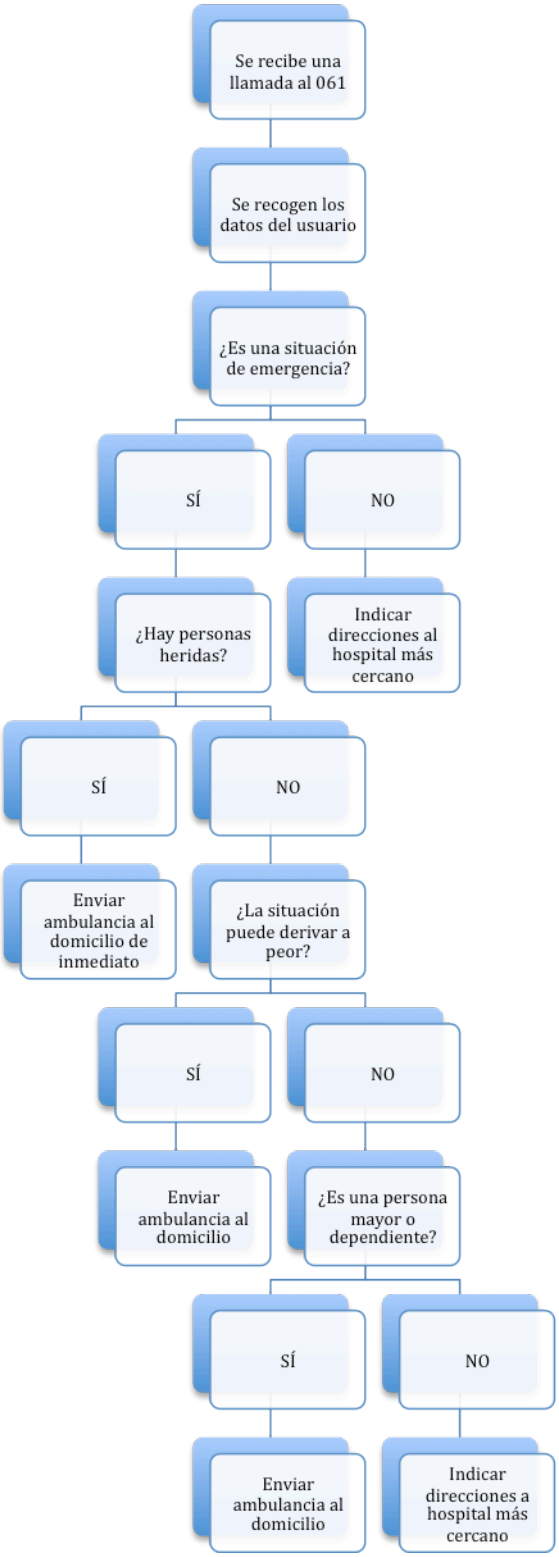
Dado el carácter imperativo de los lenguajes de programación, se entiende que es un hecho esencial el poder dar órdenes al ordenador no para que las ejecute inmediatamente, sino solo si se cumplen determinadas circunstancias. Esto permite dar a los programas una utilidad mayor que la de ejecutar una mera lista secuencial de órdenes. Esto resulta en la creación de programas que se amoldan a muchas circunstancias de uso y que dan respuestas adaptadas a cada situación que se presenta.[47]

Tanto en la vida real como en los programas informáticos, poder dar respuestas diferentes ante un contexto u otro es primordial. Lo podemos ver en el siguiente ejemplo cotidiano, que simularía un protocolo de actuación de unos servicios de emergencia. Cuando un trabajador del SAMUR atiende una llamada, en muy pocas ocasiones su modo de actuación es el de unos pasos secuenciales como los siguientes:



Con un esquema como el de arriba, el servicio de emergencias se quedaría sin ambulancias muy rápidamente y privaría de atención médica esencial a casos que sí lo necesitaran. Más bien, entonces, en un puesto de atención telefónica se seguiría un protocolo como el siguiente, en el que se actuaría de manera diferente de acuerdo con la situación que se presentara:

[47] Rara vez un programa consiste exclusivamente en una mera lista secuencial de comandos: generalmente, contiene muchas oraciones condicionales para realizar determinadas operaciones si se da un contexto específico de ejecución y otras en otro.

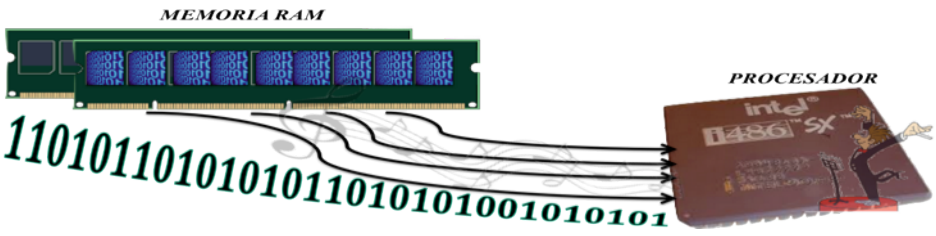


En la elaboración de un programa informático se sigue esta misma lógica: se necesitan, por ende, las «órdenes condicionadas». Sintácticamente, estas están codificadas por medio de oraciones condicionales, que hemos ilustrado en párrafos anteriores.

3. Cuando las órdenes condicionadas fallan. Un problema técnico que se debe solucionar.

Por norma general, toda construcción sintáctica en un lenguaje de programación tiene una correlación directa con una instrucción del CPU, a la que representa y abstrae. En el caso de las oraciones condicionales, estas codifican una serie de instrucciones muy especiales que reciben el nombre de *salto* o *rama* (*jump* o *branch*). Estas permiten dirigir la ejecución del programa a un bloque de instrucciones o a otro, saltándose así la mera secuencialidad de las instrucciones; pero tan solo condicionalmente. Es decir, solo saltan a un grupo de instrucciones si se cumplen unas condiciones determinadas. Los lenguajes de programación, para abstraer estas instrucciones, encontraron una manera efectiva de representarlas por medio de las oraciones condicionales de las lenguas naturales. Estas condicionales, aunque son una ingeniosa solución, pueden incurrir en un problema de rendimiento en los ordenadores; problema que deseamos abordar y explicar aquí.

Para comprender la razón de ese problema, sin embargo, debemos primero entender la vía por medio de la cual las instrucciones se transmiten al procesador. Las instrucciones (que, como ya sabemos, son en realidad conjuntos ordenados de dígitos binarios) están almacenadas en unos circuitos especializados llamados *memoria RAM*[48], interconectados directamente con el procesador[49], cuya función es la de almacenar los ceros y unos de los que se componen las instrucciones y los datos[50], y de transmitirlos al procesador cuando este los requiere. Las instrucciones, almacenadas en RAM, son recibidas por defecto secuencialmente por el procesador, una detrás de otra. Tras la terminación de una instrucción, llegará desde la memoria la que le sigue a continuación.

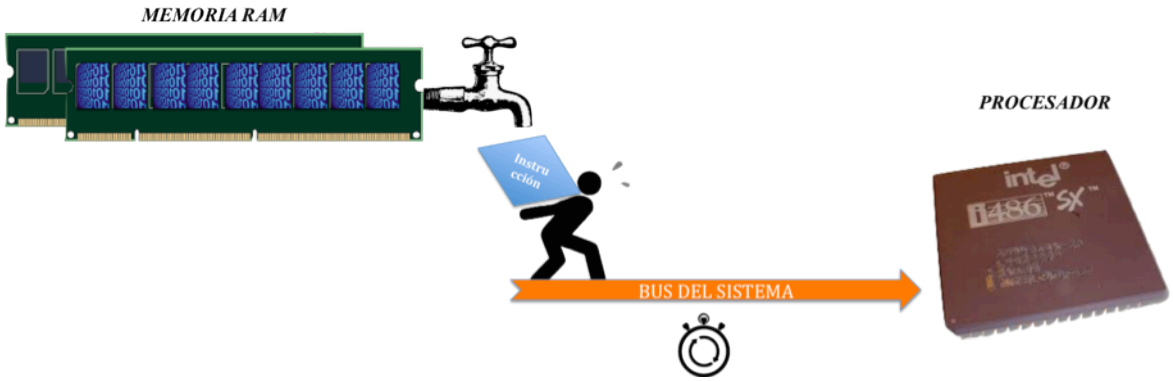


[48] Del inglés *Random Access Memory*; *Memoria de acceso aleatorio*.

[49] La configuración de este tipo de conexiones depende del tipo de ordenador, pero en todos los casos es un tipo de *bus*. Un *bus* (del latín *ómnibus*) es todo tipo de conexión o cableado que existe entre componentes de un ordenador en la placa base y que los liga entre sí, como ocurre en este caso entre procesador y memoria RAM.

[50] No solo las instrucciones, sino también los datos con los que trabaja un ordenador (ya sea un archivo de música, una imagen o incluso las líneas que estamos leyendo) están también codificados como datos binarios.

Este modelo de ejecución, que ha sido el estándar desde el principio de la Informática, adolece de un grave fallo: la conexión entre procesador y RAM es demasiado lenta, por lo que las instrucciones llegan a cuentagotas al procesador. Paradójicamente, cuanto más rápido es un procesador[51], más se agrava esta situación de lentitud, pues el procesador consigue terminar de ejecutar más rápidamente sus instrucciones pero debe quedarse a la espera de que le llegue, desde la memoria, la siguiente instrucción.



Ante este problema, que se fue agravando progresivamente, se ideó una solución a mediados de los años 90: proveer al procesador de una memoria auxiliar, conectada directamente con este, y de mayor velocidad que la memoria RAM, llamada *caché de instrucciones*. Esta es un tipo de memoria *caché*[52], una memoria rápida, con múltiples usos, cuyo cometido principal es tener disponible una copia de los datos (almacenados en la memoria principal) que el procesador necesita frecuentemente.[53] Las *cachés* suelen ser de tamaño reducido, por lo que solo almacenan una minúscula porción de información.

Respecto al *caché de instrucciones*, esta memoria almacena un grupo de instrucciones, consistente, por un lado, en aquella que el procesador va a ejecutar inmediatamente, además de un conjunto de las que están inmediatamente a continuación. El procesador, al ejecutar instrucciones, no recibirá estas desde la memoria RAM, sino desde la *caché*. De este modo, cuando el procesador tenga que disponer de la instrucción que necesita ejecutar en ese momento, podrá acceder a ella rápidamente, sin depender de un lento acceso a la memoria principal. Además, ya que las instrucciones que deberá ejecutar a

---

[51] En los años ochenta, con los primeros ordenadores personales, esto aún no era un problema, pues los procesadores eran muy lentos y, entre la ejecución de una instrucción y de otra, había suficiente tiempo para que llegara la siguiente desde la RAM. Conforme los CPU iban ganando en velocidad, sí que se fue convirtiendo en un problema grave.

[52] Es una palabra inglesa que, a su vez, es un préstamo del francés: el adjetivo *caché*, «escondido».

[53] De este modo, si el procesador requiere *x* datos cada pocos segundos, una *caché* los precarga desde la memoria principal para que el procesador tenga rápido acceso a ellos. Evita así, de este modo, un lento acceso a la memoria RAM.

continuación van a estar también disponibles de antemano en la *caché*, el tiempo de acceso se reducirá drásticamente. Cuando el procesador lea la última instrucción que se había precargado en la caché, esta cargará otro grupo de instrucciones desde la memoria principal, repitiéndose el proceso *ad infinitum*[54]. Este modelo funciona fundamentalmente gracias el hecho de que las instrucciones están almacenadas de manera secuencial (como secuencias de ceros y unos) en la memoria RAM.

Con esta solución, la situación de eficiencia de los procesadores parecía en principio haber sido resuelta. La memoria *caché*, que antes se instalaba como un set de chips opcionales en zócalos de la placa base del ordenador, empezó a integrarse nada más y nada menos que en el propio chip del procesador. Sin embargo, la incorporación de *caché de instrucciones* a los procesadores trajo un nuevo quebradero de cabeza, íntimamente ligado con las oraciones condicionales y las instrucciones de salto. Si bien la *caché* es efectiva cuando las instrucciones de un programa son secuenciales, la mayor parte de las aplicaciones tienen un gran número de instrucciones de salto[55], que rompen dicha secuencialidad. La *caché de instrucciones*, dadas estas instrucciones, no puede ciegamente cargar desde la memoria RAM el bloque de instrucciones contiguo a una instrucción de salto, pues la ejecución podría saltar a otra ubicación en la memoria sin previo aviso. Este problema, además, se ve agravado por el hecho de que el destino de una instrucción de salto no puede saberse de antemano hasta que no se ejecuta esa instrucción. Este es un problema grave, pues si la *caché* decide cargar un bloque de instrucciones que al final resulta no ser el requerido, deberá incurrir en un (lento) acceso a la memoria RAM para conseguir la información correcta. Ya que la mayor parte de los programas actuales contienen oraciones condicionales, esta situación podría poner en duda la viabilidad de la memoria caché.

Como solución a este problema, los fabricantes proveyeron a los procesadores de unos circuitos adicionales llamados *Predictor de ramas* (*Branch predictor* en inglés). Estos predicen, por medio de estadística, qué conjunto de instrucciones es más probable que se ejecute tras una instrucción de salto. Para ello, dejan que un bloque de instrucciones determinado se ejecute más de una vez y, en cada una de las ocasiones en las que se ejecute, se toma nota en una pequeña memoria de cuál ha sido el destino de salto preferido

---

[54] Para simplificar esta explicación, se ha obviado el hecho de que los procesadores suelen programar la ejecución de varias instrucciones simultáneamente por medio de una técnica llamada segmentación de instrucciones (*instruction pipelining*, en inglés).

[55] Recordemos que, en líneas anteriores, habíamos comentado que la presencia de bloques condicionales en los programas informáticos es imprescindible para la utilidad de estos.

al ejecutarse una oración condicional. De este modo, por medio de la estadística, este mecanismo es capaz de indicar a la *caché de instrucciones* qué bloque de instrucciones esta debería cargar preferentemente; hecho que hace aumentar drásticamente el rendimiento de los procesadores.

Esta solución, sin embargo, adolece del hecho de que está meramente basado en una técnica de recuento estadístico, de tal modo que solo tiene en cuenta cuáles han sido las instrucciones más comúnmente ejecutadas ante una instrucción de salto, de una manera rígida y meramente técnica. Este sistema no recurre, en cambio, a otras técnicas como podría ser, por ejemplo, hacer una discriminación y clasificación de cuáles pueden haber sido las condiciones que hayan llevado a que se ejecutara unas instrucciones frente a otras. Este hecho tiene una repercusión directa, y es que el *predictor de ramas* puede llegar a hacer recomendaciones equivocadas a la *caché de instrucciones*, en el sentido de que por estadística se asumiera que se iban a ejecutar unas determinadas instrucciones y que, al final, las condiciones de la instrucción de salto determinaron que se ejecutarían otras.

En el capítulo siguiente presentaremos una solución técnica, basada en el concepto de «órdenes condicionadas» de las lenguas naturales, que busca resolver esta carencia del *predictor de ramas* y complementar su funcionamiento.

CAPÍTULO IV: SOLUCIÓN TÉCNICA

Al escribir un programa informático, es habitual que un desarrollador siembre su código de oraciones condicionales, que vimos en el capítulo anterior. Lo hará con la perspectiva de que su programa, ante situaciones cambiantes que se dan durante la ejecución, será capaz de responder adecuadamente a distintos problemas que se puedan plantear.

En este sentido, escribir un programa no es muy diferente a establecer, en la vida real, un plan a seguir; por ejemplo, en una oficina de Correos. Tanto en una situación real como en una virtual (es decir, en la Informática), el utilizar oraciones condicionales resulta de gran utilidad para este cometido, pues es el mecanismo lingüístico por medio del cual se pueden plantear posibles obstáculos y sus soluciones en el marco de un plan de actuación. Debemos adelantar que estas oraciones condicionales serían en realidad «órdenes condicionadas», aquellas del tipo: «Si alguien trae una carta mal sellada, no le permitas enviarla». No obstante, nos referiremos a ellas de aquí en adelante como «oraciones condicionales» para mayor facilidad en la exposición.

En una situación real, es posible que los responsables de crear un plan de actuación tuvieran que tener en cuenta situaciones que, dentro de lo habitual, tienden a no ocurrir con demasiada frecuencia: en el ejemplo de una oficina postal, es más que improbable que un cliente quisiera facturar una jaula con unos periquitos vivos; eso no quiere decir, sin embargo, que ese hecho no pueda llegar a ocurrir. La oficina de Correos debería tener, por ende, un protocolo en el que se tengan en cuenta estas situaciones extremas. Si los delimitadores de ese plan de actuación decidieran tener en cuenta situaciones así, posiblemente también recurrieran a formularlas con oraciones condicionales, pero con aquellas del tipo 2 (las llamadas *hipotéticas*): «Si alguien trajera un animal vivo a Correos, avisa a la policía por posible maltrato animal».

El codificar situaciones de este tipo a través de prótasis hipotéticas es algo, como podemos ver, cotidiano en situaciones de la vida real. No obstante, si quisiéramos replicar esto en un lenguaje de programación, nos encontraríamos ante la limitación de que dichos lenguajes solo cuentan con oraciones condicionales de tipo 1 (las de tipo *real*). De este modo, en un programa que recreara el modo de actuación de una oficina de correos, tendrían la misma validez semántica y probabilísticamente una oración condicional que

comprobara si el cliente ha puesto bien el sello y otra en la que se pretendiera facturar un animal vivo. Esta parece ser, *a priori*, una limitación semántica bastante seria en los lenguajes de programación. No obstante, esto ha tenido, históricamente, una razón de ser: como explicamos en el capítulo anterior, los constructos de los lenguajes de programación constituyen en realidad abstracciones, que emulan el lenguaje natural, de algunas de las instrucciones que entiende directamente el procesador, y aquí las oraciones condicionales no son en realidad más que una efectiva metáfora de aquellas instrucciones llamadas de *salto* (o *jump*) en los procesadores. Para evitar redundancias, y ya que no hay ningún mecanismo interno en los CPU que permita tener en cuenta el grado de probabilidad de una condición determinada, se decidió implementar solo aquellas oraciones condicionales de tipo real. Proponer, así pues, prótasis de tipo hipotético para su uso en programación podría parecer, en principio, algo superfluo, pues no tiene, aparentemente, utilidad práctica en los procesadores.

No obstante, aquí queremos argumentar lo contrario: sí que sería factible (y, de hecho, resultaría de gran interés) el poder implementarlas en el seno de los actuales lenguajes de programación. Aunque no negamos las posibles reticencias a este respecto, en este capítulo vamos a exponer una situación técnica en la que la existencia de condicionales hipotéticas sería de gran utilidad y contribuiría a un mayor desarrollo de los lenguajes de programación.

En el capítulo anterior, hemos hecho una extensa descripción de la manera en la que las instrucciones son transmitidas, desde la memoria RAM, hasta el procesador. Con un breve apunte de historia de la Informática, hemos justificado el por qué de la existencia de una pequeña memoria auxiliar, la *caché de instrucciones*, como puente entre la memoria y el CPU. Habíamos, finalmente, destacado la incorporación de un mecanismo a los procesadores, llamado *Predictor de ramas* (*Branch predictor*), que solucionaba un problema serio (casi limitante) de eficiencia en la *caché*: la imposibilidad de saber de antemano el resultado de una condicional y, por ende, la probabilidad (constante, además) de que la caché precargara un conjunto de instrucciones equivocado. Este mecanismo funciona, como ya habíamos comentado, por medio del análisis, individual y de manera repetida, de cada oración condicional de un programa informático, mientras este se ejecuta. Se obtiene así una muestra estadística de qué conjunto de instrucciones es más probable que se ejecute tras cada condicional. Este sistema, aunque eficiente, adolece del hecho de ser puramente estadístico: se limita a apuntar el número de veces que se han ejecutado, ante

una condicional, una serie de órdenes frente a otras y, con esos datos, ofrece un porcentaje probabilístico a la *caché de instrucciones* para que esta tome la decisión de qué conjunto de instrucciones cargar preferentemente.

Es más que probable que el programador, quien conoce a fondo su programa, sepa a qué condiciones deberá enfrentarse su aplicación y que escriba instrucciones que respondan según esas situaciones. A la hora de planificar el funcionamiento de su programa, como ocurría en aquel ejemplo de una oficina de Correos mencionada arriba, es probable que el programador se plantee cuáles son las condiciones con mayor probabilidad de ocurrir en su código. Sin embargo, para que su aplicación sea lo más útil posible, también debería responder a situaciones extrañas e improbables que, sin embargo, pueden llegar a suceder y ante las que debería haber una respuesta acorde (como el ejemplo de la jaula de periquitos enviados a una oficina postal). Esto es aún más relevante en el contexto de programación porque, frente a la vida real, a un ordenador se le tiene que especificar, clara y explícitamente (y sin ningún tipo de ambigüedad) qué es lo que debe hacer y las situaciones a las que puede llegar a enfrentarse (una máquina no tiene ningún tipo de inferencia por contexto, al contrario que los humanos[56]). No obstante, el programador actual no cuenta con ningún mecanismo para expresar este concepto. Por ende, se encuentra ante la tesitura de poseer una valiosísima información sobre el funcionamiento de una parte de su programa que, sin embargo, no va a poder aprovechar de manera práctica[57]. Esto va a provocar que esa circunstancia de ejecución, de menor probabilidad, vaya a ser codificada por medio de una mera condicional real; esta condicional, durante la ejecución, va a ser tratada ciegamente por el *predictor de ramas* como una posibilidad de ejecución tan válida como otras condicionales que tienen una probabilidad más alta de suceder.

Ante este problema, en el contexto de este trabajo querríamos sugerir la introducción de oraciones condicionales hipotéticas en los lenguajes de programación, para

---

[56] Un ordenador podría asemejarse, en este sentido, a un útil robot que puede hacer cosas casi imposibles para nosotros, pero al que hay que decirle con total exactitud, punto por punto, cómo debe hacerlas (quizá, incluso, cómo debe mover las piernas para andar). Esta es una de las grandes «debilidades» de los ordenadores y la razón principal por la que se están dedicando incontables horas de investigación y recursos a recrear los mecanismos de la mente humana en las computadoras en el terreno de la Inteligencia Artificial.

[57] Sí que existe un mecanismo bastante aproximado, llamado «Excepciones», que podría parecer muy similar a la propuesta que vamos a presentar a continuación; sin embargo, no lo es. Este solo se aplica en los programas para expresar situaciones de error durante la ejecución, error que se busca solucionar y subsanar limpiamente. Tan solo busca interrumpir y abortar el proceso cuando en este ha ocurrido una «avería» irreparable. Como su propio nombre indica, se trata de «excepciones», algo que tiene *a priori* un carácter semántico similar a las condicionales de tipo hipotético. Sin embargo, estas «excepciones» no son solo internamente muy diferentes de la solución que vamos a proponer a continuación, sino que además contarían más bien como un subtipo, bastante limitado por otra parte, de un espectro más amplio representado por las condiciones hipotéticas.

así subsanar este problema. Estas formas condicionales podrían materializarse de una multitud de formas, pero aquí sugerimos la siguiente implementación:

```
ifw (Condiciones a ser comprobadas)        ← Prótasis
{
        //INSTRUCCIONES          ← Apódosis
        A SER EJECUTADAS//
}
```

Donde la palabra clave «ifw» sería una contracción de la forma habitual para las prótasis de las oraciones condicionales hipotéticas en inglés («***if*** *it* ***was/were***»)[58].

La utilización de estas condicionales hipotéticas en una aplicación permitiría al programador expresar la noción de es probable que la condición que es codificada con ella ocurra con poca frecuencia. Aunque no hay ninguna limitación para utilizar condicionales reales (de tipo «*if*») para este mismo cometido, internamente carecerían del funcionamiento que tendrían las condicionales de tipo «ifw» (funcionamiento que vamos a describir a continuación). Dicho funcionamiento consistiría en complementar la labor del *predictor de ramas* por medio de la aportación a este de información precisa (que el propio creador del programa habría aportado, de manera indirecta, al programar con estas condicionales) sobre la probabilidad de que un bloque de instrucciones pudiera o no ejecutarse. Con esta solución, el *predictor* pasaría de tratar ciegamente cada bloque condicional como equivalentemente capaz de ser ejecutado frente a otros y, en su lugar, este mecanismo eliminaría de su análisis todas aquellas prótasis que están introducidas por «ifw». Al ser completamente descartadas, dicho mecanismo tendría que hacer menos trabajo, lo que supondría una mejora en su rendimiento. Por supuesto, esto establece *a priori* un pequeño problema: aunque las condiciones hipotéticas son menos probables, y por tanto eliminarlas del análisis de predicción puede resultar ventajoso, podría ocurrir la circunstancia de que, en algún determinado momento, esa condicional sí que se cumpla. Ante ese hecho, el *predictor* habría ignorado completamente la posibilidad de que esa condicional pudiera siquiera suceder, por lo que esas órdenes no estarían cargadas en la *caché de instrucciones*:

---

[58] La modificación de la palabra clave «*if*» es aquí importante, ya que para delimitar esa prótasis como una de tipo hipotético no existe una vía «morfológica» en los lenguajes de programación para marcar el contenido de la prótasis (aquel que está entre paréntesis), como sí lo existe en una lengua como el español. Además, en el inglés se sigue un sistema relativamente similar al aquí sugerido, pues ni siquiera se recurre a una marcación morfológica *per se* (entendiéndose aquí como uso de formas con desinencias verbales) para denotar el carácter hipotético de una prótasis, sino que meramente se incorporan las palabras «*it*» y «*was/were*» a la conjunción condicional «*if*». Esto tiene unas repercusiones sintácticas y estructurales mucho mayores que las de una mera aposición de palabras, pero a efectos prácticos el recurso funciona así.

ello llevaría al procesador a cargar manualmente las instrucciones requeridas desde la memoria principal (un proceso que, como hemos recalcado en capítulos anteriores, es relativamente lento). No obstante, ya que las ocasiones en las que esto puede suceder son bastante limitadas, en comparación con el gran número de veces en las que el ahorro de tiempo sería un hecho palpable, las ventajas contrarrestarían con amplio margen a las desventajas[59].

Téngase en cuenta que, en el transcurso de este capítulo, hemos, por un lado, advertido la ausencia de condicionales hipotéticas en los lenguajes de programación pero, al mismo tiempo, no hemos comentado nada al respecto de las condicionales *contrafactuales* o *irreales*. Mientras que las del tipo 2 tienen una cabida en programación, las del tipo 3 no tienen más remedio que quedarse al margen en este sentido. Esto se debe al hecho de que, en primer lugar, las oraciones condicionales de los lenguajes de programación son en realidad «órdenes condicionadas», donde recordemos que solo podían existir, debido a la naturaleza semántica del imperativo, dos tipos de prótasis (las reales y las hipotéticas). Por otro lado, tampoco tendría mucho sentido plantear, en un lenguaje de programación, la existencia de instrucciones que se ejecutaran en circunstancias que no pueden suceder nunca (pues se refieren a hechos pasados o imposibles). Por este motivo, no hemos tenido en cuenta en nuestro análisis las condicionales irreales[60].

Hemos de adelantar que en la arquitectura de los procesadores actuales no existe todavía un mecanismo capaz de implementar esta funcionalidad: esta sería, por tanto, una solución técnica con vistas a posibles modificaciones posteriores en la ingeniería de los CPU. Han existido soluciones en algunos procesadores, como el Pentium 4 de Intel, que venían con instrucciones para precargar datos en la *caché* (algo que se conoce como *Software precaching* o *Precacheo por software*), pero los procesadores actuales de Intel ya no vienen, en principio, con esta funcionalidad (aunque técnicamente existe un mecanismo por medio del que se podría implementar esta solución). Por otra parte, existen ya propuestas de lenguajes de programación con condicionales hipotéticas: un ejemplo es N-

---

[59] Este hecho es comparable con un algoritmo de compresión de datos, Huffman, utilizado entre otros para el formato de imagen JPEG. Un algoritmo de compresión busca reducir de manera considerable el tamaño de un archivo, para que así ocupe menos espacio. El algoritmo Huffman, por su funcionamiento interno, tiende a reducir el tamaño de los datos, pero en algunas ocasiones puede hacerlos más grandes, al caer en cierta redundancia accidental. No obstante, dado que el número de datos que son reducidos es mucho mayor que aquellos con redundancia, esta diferencia se contrarresta.

[60] No obstante, tampoco habría que destacar en el futuro que, en el contexto de aprendizaje de las máquinas en Inteligencia Artificial, este tipo de condicionales expresen un contenido muy rico, experiencial, que facilite la formación de las máquinas en su toma de decisiones (lo mismo que ocurre con el ser humano).

Prolog, una versión del lenguaje Prolog. No obstante, estas ideas se han aplicado a los lenguajes de programación de tipo declarativo (basados en establecer sets de reglas que guían la ejecución de un programa), mientras que aquí lo aplicamos a otro tipo de lenguajes de programación: los imperativos, como C, Java, Python…, que se basan en la idea de transmitir órdenes al procesador; de ahí el nombre de «imperativos».

A pesar de estos problemas, consideramos que esta solución sería de gran efectividad. No solo aumentaría las capacidades de expresión semánticas de los lenguajes de programación, sino que además tendría un efecto real sobre el rendimiento de la máquina, un doble efecto (semántico y tecnológico) que se podría tener en cuenta. La programación es una actividad creativa, en la que se utilizan los recursos de los lenguajes de programación para encontrar soluciones prácticas a problemas, y el hecho de contar con una herramienta más en el set que además evite el encorsetamiento inherente a expresarse solamente con condicionales reales, podría suponer un paso adelante en el mundo de la programación.

CONCLUSIONES

En este trabajo, por un lado, hemos intentado hacer una descripción general de los lenguajes de programación desde el punto de vista de la Lingüística, señalando los puntos en común de estos con las lenguas naturales, al igual que las diferencias que existen entre ambas. Para ello, hemos hecho una introspección lingüística en ciertos mecanismos del lenguaje, el de los actos de habla directivos y las oraciones condicionales, pues guardan estos bastante similitud con los principales elementos de los llamados lenguajes de programación imperativos: órdenes y condicionales. Hemos tratado de ahondar en los rasgos sintácticos y semánticos de estas formas, recurriendo para ello a conceptos como los de *Mundos Posibles*, *condiciones de verdad y éxito*, a la filosofía del lenguaje de Austin y Searle (*Actos de habla*), nociones gramaticales sobre el modo imperativo y las oraciones condicionales.

Por otra parte, nos hemos atrevido a sugerir una posible vía de investigación que, si se desarrollara con más profundidad, sería una contribución más en la creciente colaboración entre Lingüística e Informática, tras una tradición de mutua ignorancia. Además, la propuesta que hemos presentado –la introducción de oraciones condicionales hipotéticas en los lenguajes de programación– podría ayudar en la continua evolución de estos, en un intento de desencorsetarlos y hacerlos más próximos a la expresividad humana, sin por ello perder la capacidad de comunicarse efectivamente con la máquina.

BIBLIOGRAFÍA

«Acto de habla» en *Diccionario de términos clave de ELE*. Instituto Cervantes (en línea: <https://cvc.cervantes.es/ensenanza/biblioteca_ele/diccio_ele/diccionario/actodehabla.htm >).

ALARCOS LLORACH, EMILIO (1995). *Gramática de la lengua española*. Espasa-Calpe (pp. 150-151, 376-379).

AUSTIN, JOHN L. 1975 *How to do things with words* Oxford university press [*Cómo hacer cosas con palabras*, Paidós, Barcelona, 2016 (pp. 85-100).]

CUSTER, HELEN (1993). *Inside Windows NT*. Microsoft Press.

ESCANDELL VIDAL, M. VICTORIA (2004). *Fundamentos de Semántica composicional*. Ariel (pp. 233-259, 301-304).

GARRIDO MEDINA, JOAQUÍN (1999). «Los actos de habla. Las oraciones imperativas» en *Gramática Descriptiva de la Lengua Española 3: Entre la oración y el discurso, Morfología*, coord. por Ignacio Bosque y Violeta Demonte. Espasa (pp. 3879-3928).

KUSSWURM, DANIEL (2014). *Modern x86 assembly language programming. 32 bit, 64 bit, SSE and AVX*. Appress.

MENDÍVIL-GIRÓ, JOSÉ LUIS (2012). «Lingüística histórica y teoría de la evolución: semejanzas, diferencias e implicaciones» en *XIII Jornadas de Lingüística*. Universidad de Cádiz (pp. 55-102).

MENDÍVIL-GIRÓ, JOSÉ LUIS (2018). «Naturaleza y cultura en el lenguaje = sintaxis y léxico en las lenguas» en *Actas do XIII Congreso Internacional de Lingüística Xeral*. Universidad de Vigo (pp. 607-614).

MONTOLÍO, ESTRELLA (1999). «Las construcciones condicionales» en *Gramática Descriptiva de la Lengua Española 3: Entre la oración y el discurso, Morfología*, coord. por Ignacio Bosque y Violeta Demonte. Espasa (pp. 3643-3737).

PENFOLD, R. A. (1994). *An Introduction to 68000 Assembly Language*, Bernard Babani (Publishing) (pp. 2-10).

PETZOLD, CHARLES (1999). *Code: The Hidden Language of Computer Hardware and Software*. Microsoft Press.

SEARLE, J. R., & SEARLE, J. R. (1969). *Speech acts: An essay in the philosophy of language* (Vol. 626). Cambridge university press.

VV. AA. (2009), *Nueva gramática de la lengua española 2. Sintaxis II*. Espasa (pp. 3118-3152, 3528-3557).

ZAEFFERER, DIETMAR (1991). «Conditionals and Unconditionals: Cross-linguistic and Logical Aspects» en *Semantic Universals and Universal Semantics*. Foris Publications (pp. 210-236).