Máquinas interactivas y simulaciones

Pío García

Universidad Nacional de Córdoba

Abstract. La interacción en computación es un fenómeno estudiado a partir de los años 70 del siglo pasado desde una perspectiva formal. Sin embargo no resulta una tarea sencilla evaluar el alcance de esta perspectiva para distintas áreas en donde la computación es relevante. En particular la discusión acerca de simulaciones computacionales podría enriquecerse a partir de la modelización de la interacción. **La reflexión filosófica acerca de simulaciones computacionales descansa en parte en una caracterización de los modelos computacionales implicados. La forma habitual de caracterizar a estos modelos es en términos de máquinas de Turing tradicionales. Este aspecto se manifiesta en argumentos como los de Margaret Morgan acerca de la diferencia entre las simulaciones como sistemas cerrados y los experimentos como sistemas abiertos. Una objeción que se le puede hacer a esta estrategia es que no considera a los tipos de simulación en donde los aspectos interactivos puedan ser relevantes. Así, por ejemplo, desde un punto de vista teórico, Peter Wegner ha sugerido que la ubicuidad del fenómeno de la interacción entre máquinas involucra un cambio en el paradigma de la computación tradicional. Robin Milner ha realizado una sugerencia similar, focalizándose más bien en fenómenos concurrentes – procesos que interactúan-. Hoare ha propuesto una forma de modelización de este tipo de procesos. A su vez pueden considerarse algunos de los trabajos de Turing acerca de la relación entre máquinas como una anticipación estas perspectivas. En el presente trabajo analizaremos algunas maneras de modelizar e interpretar la interacción y evaluaremos su eventual alcance para una filosofía de las simulaciones. Para esta última tarea tomaremos en particular las sugerencias de Vincent Schachter a propósito de los límites de la modelización del fenómeno de la interacción.**

Keywords: computación, concurrencia, simulaciones computacionales

Introducción

La reflexión filosófica acerca de simulaciones computacionales descansa, en gran medida, en una caracterización de los modelos computacionales implicados. De esta manera parecen entenderlo los filósofos como Guala (2002), Winsberg (1999, 2003), Parker (2009) y Morgan (2003) quienes evalúan semejanzas y diferencias entre simulaciones computacionales y experimentos. Así, Margaret Morgan (2003) sugiere que un experimento puede sorprender en un sentido genuino, al generar conocimiento, mientras que las simulaciones computacionales sólo pueden mostrar lo que ya está en sus modelos. En gran medida esta consideración parece apoyarse en un contraste entre sistemas que interactúan con algo más, como sería el caso de

los experimentos y aquellos sistemas cuyo comportamiento quedaría limitado a sus modelos como sería el caso de las simulaciones computacionales-.

En general esta perspectiva parece adecuada, sin embargo se podría objetar que no se consideran aquí tipos de sistemas computacionales en donde los aspectos *interactivos* puedan ser relevantes. Una forma directa de plantear esta objeción es a través de las llamadas simulaciones multi-agentes (Humphreys 2004) Y efectivamente se ha reconocido que hay aspectos distintivos, como alguna forma de emergencia, en este tipo de simulaciones que no coinciden con las simulaciones más tradicionales[1]. Pero hay una forma más general de plantear esta cuestión. El fenómeno de la interacción y las formas de modelarlo ha sido analizado en el campo de la computación desde una vertiente alternativa. Tony Hoare, Robin Milner, Carl Petri, Peter Wegner y Carl Hewitt entre otros, han estudiado- algunos desde los sesenta- de qué manera la interacción en computación podría ser vista como una situación más común de la que se cree. Muchas tareas habituales en donde hay intervención del usuario o comunicación entre programas y/o computadoras podrían ser vistas como parte de una situación en donde la interacción es relevante. Aunque las formas de modelizar este fenómeno difieren sustancialmente. Igualmente, difieren las interpretaciones de los informáticos acerca del alcance de su trabajo.

En este trabajo consideraremos aspectos generales de esta perspectiva de investigación. Como veremos, algunos evalúan el fenómeno de la interacción en computación como una forma distinta de hablar de una noción clásica de computación mientras que para otros implicaría la consideración de un sistema distinto - en cierto modo abierto o vinculado con algo más-. Por esta razón, aunque no será tema de este trabajo, la presente discusión podría ser, en principio, relevante para evaluar supuestos habituales en la filosofía de las simulaciones computacionales.

2 Interacción y máquinas de Turing

El problema de la interacción y la concurrencia en computación aparece de manera sistemática a partir de fines de los sesenta y principios de los setenta.

Milner publicó su primer trabajo sobre modelos concurrentes en 1973. Su interés, en esta época, estaba centrado en modelar procesos que fueran más allá de programas secuenciales descriptibles en términos de su mera entrada-salida. Hacia finales de los setenta este interés originario se cristaliza en una propuesta procedimental, a saber la construcción de un lenguaje que fuera un “cálculo para sistemas que se comunican”: CCS. En 1978 Hoare presenta otro lenguaje de programación denominado “comunicación entre procesos secuenciales” (CSP en inglés) que va tener un gran impacto en la comunidad de investigadores del área. En particular porque ha sido utilizado en la industria del software en tareas de especificación de programas y su posterior verificación formal.

Pero, ¿por qué surge y qué se entiende aquí por concurrencia o interacción? Comenzando con lo segundo, se entiende por concurrencia el fenómeno en donde hay una ejecución de procesos computacionales en simultáneo que pueden eventualmente interactuar. Así, por ejemplo si tenemos dos procesos computacionales sin un reloj común, el resultado de

la interacción puede ser incierto. Se supone entonces que si no hay un reloj global que controle todos los eventos, entonces *la conducta resultante* va a depender de las relaciones que se establezcan entre programas controlados por distintos relojes internos.

Carl Petri es uno de los primeros que hace notar la importancia de analizar “la noción de ejecución de una máquina como una secuencia de estados globales o como una secuencia de eventos ordenados por su ocurrencia temporal con respecto a un reloj” (Esparza 2010). Petri observa que es muy difícil - en realidad que no se pude- *implementar* un reloj global porque “la información debe viajar a una velocidad finita” y por tanto hay circunstancias en donde no hay ninguna parte del sistema que pueda “conocer” el estado de todos los componentes en un cierto momento. De esta manera Petri sugiere reemplazar las ejecuciones de procesos no secuenciales por un tipo de relación “causal” que dependa del observador. Esta propuesta de Petri se materializa luego en un lenguaje de programación para tratar con sistemas distribuidos[2].

Pero en términos más generales, las sugerencias que hace Petri suponen un cambio en el análisis de los procesos computacionales. Y este aspecto se vincula con la otra pregunta que hacíamos arriba. Al menos parte de la respuesta a la cuestión de porqué surge el problema de la concurrencia, la podríamos encontrar en el cambio que representó, en la actividad de los informáticos, pasar de la manipulación de una máquina - programa o computadora- a múltiples máquinas. Esto implicaba, entre otras cosas, tratar de entender arquitecturas cliente-servidor, sistemas distribuidos o el paralelismo entre procesos computacionales.

También a principios de los setenta, Carl Hewitt propone el modelo ACTOR para sistemas concurrentes con principios muy similares los lenguajes citados (CCS o CSP). De acuerdo con el propio Hewitt, éste se inspiró en la creciente complejidad de lenguajes de programación de alto nivel y en las dificultades que emergen a partir de la consideración de los aspectos físicos y prácticos de los sistemas computacionales.

Dichos aspectos prácticos- al menos los que involucraban comunicación entre máquinas- hicieron surgir problemas no previstos. La reacción inicial fue tratar de evitar o anular este tipo de circunstancias. Así, por ejemplo, cuando un proceso necesita recursos utilizados por otro, esta situación puede derivar en que un programa sea incapaz de cambiar de estado. Se requería, entonces, de rutinas para tratar con estos “puntos muertos” (*deadlocks*). Se suele señalar, también, que algunos de los primeros programas concurrentes fueron los “controladores de interrupciones”. La necesidad de manejar la información que venía de interfaces como el teclado, que podía interrumpir un proceso computacional, hizo que se desarrollaran recursos como los “espacios de memoria temporales” - “*buffers*” -que permitían guardar instrucciones hasta que el programa las pudiese procesar. También de esta forma podrían interpretarse la implementación en los sesenta por parte de Dijskstra de rutinas computacionales denominadas “semáforos” para controlar y eventualmente excluir procesos informáticos. Como dicen varios comentadores, no solo no existía un lenguaje para expresar algunos fenómenos que ocurrían en la comunicación entre sistemas sino que la misma perspectiva “secuencial” tradicional impedía ver aquí algo más que un error de diseño.

Pero, adoptar una perspectiva positiva y operacional no era una tarea sencilla y exigía al menos explicitar algunos supuestos generales. Cuando Milner construye su lenguaje CCS, considera que concurrencia -acción simultánea e independiente- y comunicación -interacción- son los aspectos fundantes de su lenguaje. En particular la concurrencia supone, como mínimo, que se deben distinguir entre "partes" que puedan interactuar. A estas partes se las suele llamar "agentes" en tanto uno está interesado en las 'acciones' que llevan adelante. A su vez estos agentes pueden comunicarse - recibir o enviar datos-, el cual es considerado un caso especial de interacción - esto es, cuando hay sincronización entre la entrada y salida de dos procesos-. Si nos centramos en esta relación de comunicación podemos discriminar entre grupos de agentes y considerar a algunos de ellos un tipo de "medio" o "contexto". Por supuesto que el modelizador puede cambiar de perspectiva y fijándose nuevamente en la relación de comunicación definir *otra* relación entre sistema y medio. Este cambio de perspectiva puede provocar que estados resultantes de alguna comunicación sean "impredecibles". Así, "la impredecibilidad es una característica ineludible de los sistemas concurrentes" (Schachter, 1999). Por esta razón no se intenta anular este aspecto sino operacionalizarlo.

Como bien lo señala Schachter, la "conducta del sistema (...) se convierte en un árbol de acciones internas y potenciales interacciones. Los nodos (de este árbol) son estados del agente y las "ramas" son las acciones (transiciones de un estado a otro) y los nodos con múltiples descendientes pueden ser estados donde el agente realiza elecciones no deterministas. Este árbol captura no sólo el conjunto de "ejecuciones" computacionales - las trazas- sino también la estructura de las elecciones posibles durante la computación. (...) expresa (...) la estrategia de un agente en su interacción con el medio" (Schachter 1999). O al menos es lo que se pretende.

Esta forma de ver la computación fue ganando cada vez más espacio. Así, por ejemplo, en los 80 del siglo pasado se construyeron algunos procesadores (como el transistor-computadora T9000) para lenguajes derivados del CSP (Occam) de Hoare. Es más, las especificaciones y verificación formal de parte de este procesador se hizo con CSP. Se esperaba que el paralelismo de este tipo de dispositivo físico pudiera ser aprovechado por lenguajes de programación concurrentes y así desarrollar más integradamente esta forma de manera de entender a la computación. Complementariamente, la intención explícita era optimizar el uso de recursos informáticos escasos a través del procesamiento en paralelo.

Además de los lenguajes, hay otra perspectiva que podría servir para nuestro objetivo de entender el alcance de la interacción computacional y es la forma en la cual se interpreta, más allá del formalismo, el fenómeno de la concurrencia.

4 Interpretaciones de la interacción

Las interpretaciones de la concurrencia se podrían dividir en aquellas que destacan la relevancia de las prácticas y aquellas que ponen el foco en una eventual extensión de la tesis Church-Turing. Una tercera alternativa, defendida por muchos, es vincular ambos aspectos.

Mucha de la discusión acerca de la concurrencia tiene que ver con las modificaciones en las maneras de manipular sistemas informáticos a partir del desarrollo tecnológico. Como vimos

en la sección anterior, este aspecto práctico aparece en el interés original por proponer modelos de interacción; pero igualmente se muestra en las controversias actuales acerca del alcance de dichos modelos.

Hoare y Schachter entre otros ven en la concurrencia un emergente de la actividad de los informáticos más que el resultado de un desarrollo teórico independiente. Es más, Milner, en el discurso a propósito del premio Turing, expresaba su convicción de que “las ideas correctas para explicar la computación concurrente vendrán sólo de la dialéctica entre los modelos de la lógica y la matemática y la apropiada ponderación de la experiencia práctica” (Milner 1993). Igualmente podrían citarse aquí las consideraciones antes señaladas por Petri acerca de los aspectos físicos de la computación. Así, el ámbito meramente formal y procedimental aparece constantemente cuestionado por los aspectos prácticos y empíricos de la informática.

Una interpretación más fuerte, es la que contrasta la concurrencia con la universalidad de la propuesta de Turing. Se ha sugerido que la concurrencia podría constituir una extensión de la tesis Church-Turing. Esta tesis constituyó una forma de extrapolar la invariancia extensional encontrada entre distintos formalismos de la computación. En este sentido se postula que dichos formalismos capturan la idea intuitiva de *procedimiento efectivo* o de *algoritmo*. Hoare y Milner han destacado que los modelos de concurrencia utilizados en lenguajes como CCS o CSP no constituyen una extensión de la tesis Church-Turing.

Sin embargo, se podría cuestionar esta conclusión de diferentes formas. Una primer forma -débil- sería señalando una diferencia entre los lenguajes que describen la concurrencia y las prácticas y fenómenos que la motivan. Como vimos, Milner ha manifestado su insatisfacción con los lenguajes de concurrencia efectivamente construidos puesto que no parecen dar cuenta de *todos* los aspectos relevantes del fenómeno al cual se refieren. Con una intención similar, Schachter ha destacado que la expresividad de las modelizaciones que tenemos de concurrencia no debería obstaculizar la búsqueda de una mejor caracterización del fenómeno que la motivó.

Pero habría una forma - más fuerte- de contrastar concurrencia y la tesis Church-Turing. Por un lado Hewitt ha cuestionado la propuesta de Turing tomando en cuenta una interpretación de lo que implica la modelización de sistemas en paralelo. Así, Hewitt sugiere que una forma de ver el modelo ACTOR es como el regreso a un modelo de computación en donde aspectos sociales o empíricos eran importantes. A fines del siglo XIX la computación era considerada como una actividad “social” en el sentido que involucraba computadoras humanas (habitualmente mujeres) trabajando de manera conjunta. Luego Turing presenta un modelo “psicológico e intrínsecamente individual” de la computación (Hewitt 1973). Los modelos de concurrencia, como el ACTOR, representarían, desde esta perspectiva, una vuelta a un modelo “social” de la computación.

En un sentido similar, Wegner sugiere que la práctica en computación en las últimas décadas ha tenido como resultado la emergencia de un tipo de máquina - las interactivas- que no sería como la propuesta por Turing. Wegner destaca que hay *tareas* - como manejar un auto- que son interactivas por naturaleza y cuya modelización no puede ser *caracterizada* en

términos algorítmicos. Un algoritmo, según Wegner, se ejecuta sin tomar en cuenta un medio, sino sólo sus instrucciones internas.

Por otro lado la interacción puede simplificar tareas en algunas situaciones o constituir “el único juego”- la única alternativa- en otras.

Pero, ¿cuál es el aspecto que haría a una máquina interactiva más expresiva, en principio, que una máquina Turing? En varios artículos lo que destaca Wegner es que la caracterización de una máquina de Turing supone que es un sistema *aislado* cuyo comportamiento depende sólo de una entrada fija. En una máquina interactiva está la posibilidad de que el sistema reciba *otro* tipo de entradas que modifiquen el comportamiento resultante. Schachter se expresa en un sentido bastante parecido a este al decir que “cuando se ejecuta una máquina de Turing, ésta está aislada del medio, esto es no recibe ninguna entrada, excepto la inicial y efectivamente una máquina de Turing está diseñada para asegurar este “aislamiento” del medio. Así se entendería el interés original de los informáticos de *evitar* la interferencia o la interacción. Pero, nos dice Schachter, hay algunos sistemas computacionales que han sido diseñados con un objetivo muy distinto. Los sistemas operativos, por ejemplo, son sistemas que por naturaleza suelen ser “reactivos”. Por esto, la conducta final no es ya el resultado de una computación en el sentido recién descripto de una máquina de Turing. De esta manera, el comportamiento generado puede no ser el resultado de una única entrada y puede “no haber un único resultado especificable al final”. En términos más generales, según Wegner, cuando una máquina toma en cuenta el medio para actuar en consecuencia se convierte, de alguna forma, en un *sistema abierto*.

5 Consideraciones finales

En el presente trabajo hemos tomado en consideración aspectos de la noción de interacción tal como lo han tratado aquellos que trabajan en concurrencia. Hemos presentado de manera muy esquemática algunas ideas a partir de los lenguajes desarrollados y de las interpretaciones propuestas.

Por un lado aparece la relevancia de las prácticas en computación tanto para el origen del trabajo en concurrencia como para la tensión actual entre lenguajes y fenómenos modelados. En este punto habría que señalar al menos dos aspectos: los desarrollos tecnológicos asociados con las máquinas físicas y las formas en las cuales se entienden, programan y manipulan las computadoras.

La tesis más fuerte acerca del alcance de la concurrencia es la propuesta por Wegner. Su sugerencia acerca de la primacía de las prácticas resulta interesante, más aún cuando llega a decir que una máquina interactiva podría ser considerada como un sistema abierto. Pero, no resulta sencillo evaluar el alcance de esta propuesta. Así por ejemplo, Prasse y Ritgen (1998) le han cuestionado a Wegner que cuando se formalizan, de una manera particular, estas ideas intuitivas, se puede mostrar la equivalencia extensional con otros formalismos que no van más allá de lo que propuso Turing.

De cualquier manera, en el caso de sistemas computacionales complejos como las simulaciones computacionales se podría decir que hay varias fuentes potenciales de interacción que necesitarían ser estudiadas con mayor detalle. Así por ejemplo, raramente se da el caso de que una simulación esté “completa” y “cerrada” desde un comienzo. En general el que trabaja con la simulación debe no sólo introducir los datos relevantes sino también trabajar en la calibración del programa mismo. Estos aspectos han hecho que filósofos como Winsberg hayan resaltado como insuficiente un esquema meramente deductivo que vaya del modelo al “resultado” de la simulación.

No obstante, esto no hace de una simulación un sistema abierto en el sentido que lo es un experimento. Intuitivamente se puede decir que en un experimento típico hay algún tipo de interacción con el sistema que se quiere estudiar. En el caso de la concurrencia hay una interacción, pero no del sistema estudiado sino del sistema con el cual se realiza el estudio. De cualquier manera, lo que queremos destacar aquí es que la concepción de los sistemas computacionales reales e implementados como cerrados no parece apropiada. Esto abre otra puerta a la discusión acerca de la forma en la cual se deben interpretar los resultados.

Más allá de estas interpretaciones, la concurrencia e interacción en informática aparecen como situaciones relevantes, más aún si consideramos que, ya sea por el uso de bases de datos distribuidas o de sistema operativos, muchos sistemas - en realidad la mayoría- son “reactivos”. Por lo cual el fenómeno de la concurrencia puede ser más común de lo que se piensa habitualmente. En palabras de Milner la concurrencia es “ubicua” (Milner 1993). En este sentido, la indagación de estos problemas podría ayudar a evaluar más cuidadosamente prácticas científicas en donde la computación desempeña un papel relevante como es el caso de las simulaciones computacionales.

Notas:

1 En los trabajos de la mayoría de aquellos que reflexionan acerca de simulaciones computacionales suele haber una advertencia de que no se considerarán simulaciones con agentes (cfr. Humphreys 2004).

2 Petri estaba preocupado por la realización física de las máquinas de Turing y este fue el tema de su disertación doctoral de 1962 (Esparza 2010).

Referencias

Winsberg. E. (1999). Sanctioning Models: The Epistemology of Simulation. *Science in Context*, 12, pp 275-292.

Winsberg, E (2003). Simulated experiments: Methodology for a virtual world. *Philosophy of Science* 70 (1):105-125.

Guala, F. (2002). Models, simulations, and experiments. *Model-based reasoning: Science, technology, values*, 59–74.

Morgan, M. S. (2003). Experiments without material intervention: model experiments, virtual experiments and virtually experiments. In H. Radder (Ed.), *The philosophy of scientific experimentation*. University of Pittsburgh Press.

Parker, W. S. (2009). Does matter really matter? Computer simulations, experiments, and materiality. *Synthese*, *169*(3), 483–496.

Milner, R. (1989) Communication and Concurrency, Prentice-Hall.

Milner, R (1993) Interaction, Turing Award, Communication of the ACM Vol.36, No.l

Prasse, M.: Rittgen, P. (1998). Why Church's Thesis Still Holds. Some Notes on Peter Wegner's Tracts on Interaction and Computability.. *Comput. J.*, 41, 357-362.

Wegner, P. (1997) Why interaction is more powerful than algorithms. CACM, 40, 80–91.

Hewitt, C.E., Bishop, P., and Steiger, (1973) R. A universal modular Actor formal- ism for artificial intelligence. In Pro- ceedings of the International Joint Conference on Artificial Intelligence. pp. 235-245.

Hoare, C.A.R. (1978) Communicating sequential processes. Commun. ACM 21, 666-677

Humphreys, P. (2004), *Extending Ourselves: Computational Science, Empiricism, and Scientific Method*, Oxford University Press

Schächter, V. (1999). How does concurrency extend the paradigm of computation?. *The Monist*, {82}. Esparza, J. A False History of True Concurrency: From Petri to Tools, Model Checking Software, Lecture Notes in Computer Science Volume 6349, 2010, pp 180-186