FLACSO
ÁREA EDUCACIÓN

NUEVOS DESAFÍOS EN EDUCACIÓN

Una mirada interdisciplinaria

Prólogo de Mario Carretero

Compiladores:
Bibiana Buenaventura
Julio Del Cueto
Emilia Di Piero
Cristian Parellada
Julia Pérez Zorrilla

Nuevos desafíos en educación. Una mirada interdisciplinaria/ Mario Carretero... [et al.]; compilado por Bibiana Margarita Buenaventura Rodríguez... [et al.] -1a ed. - Ciudad Autónoma de Buenos Aires: Flacso Argentina, 2018.

Archivo Digital: descarga y *online*
ISBN: 978-950-9379-41-1

1. Edcucación. 2. Pedagogía. 3. Psicología. I. Carretero, Mario II. Buenaventura Rodríguez, Bibiana Margarita, comp.
CDD 306.43

Corrección y edición: Tamara F. Tello Borisovsky
tamaratborisovsky@hotmail.com

Diseño de tapa e ilustración digital: Martín Alejandro Cabrera
cabreramartinalejandro@gmail.com

ISBN: 978-950-9379-41-1

Ayudas semánticas en la resolución de problemas de cálculo proposicional

Verónica D'Angelo[1]

Introducción

La escasez de ingenieros y el progresivo aumento del desinterés por las carreras técnicas, supone un factor de riesgo para el futuro desarrollo mundial y para la solución de problemas surgidos durante las últimas décadas como consecuencia del crecimiento demográfico. Actualmente, la UNESCO *Engineering Initiative* (2011) está trabajando con los Estados miembros, los socios internacionales y los expertos en programas para fortalecer la educación en ingeniería. Según el informe de UNESCO (2010), la tendencia hacia un aprendizaje más centrado en los estudiantes ha llevado al empleo de diversos enfoques, entre los cuales se destaca el Aprendizaje Basado en Problemas (PBL) (De Graaf y Kolmos, 2003; Kolmos, Dahms, y Du, 2010), una perspectiva que propone organizar y diseñar los planes de estudio alrededor de escenarios de problema que son considerados centrales.

El Plan Estratégico de Formación de Ingenieros (PEFI) lanzado en Argentina en noviembre de 2012, impulsado por la Secretaría de Políticas Universitarias, tiene como objetivo general incrementar la cantidad de graduados en Ingeniería, basándose en tres ejes:

Uno de ellos, la mejora de los indicadores académicos, que comprende cuatro objetivos específicos: generar vocaciones tempranas, incrementar la retención en el ciclo básico, incrementar la retención en el ciclo de especialización e incrementar la graduación. Si bien, un porcentaje del abandono se produce por las altas posibilidades de empleo que surgen para los alumnos avanzados, esto solo se aplica a los alumnos que han logrado alcanzar las etapas finales de la carrera.

Este trabajo de tesis, apunta a mejorar la retención en el ciclo básico de las carreras universitarias que incluyen *algorítmica* como contenido inicial, incorporando lo que denominamos *ayudas semánticas* en la resolución de problemas específicos de *cálculo proposicional*.

1 Analista de Sistemas Informáticos. Lic. en Ciencias de la Educación UNR (en curso); Maestría en Psicología Cognitiva y Aprendizaje FLACSO/UAM (en curso). Profesora adjunta de Informática Aplicada, y Auxiliar de Investigación de Programación Estructurada y Programación I, Universidad Abierta Interamericana.

Consideramos que, si bien los incentivos económicos son importantes, hay factores de deserción ligados a sesgos comunes en educación, producto de diferencias psicológicas, culturales y epistemológicas de un paradigma científico que restringe implícitamente el ingreso a un sector acotado de la población, supuesto poseedor de capacidades naturales para la algorítmica y la matemática. No obstante, por ser la *escritura*, el *cálculo infinitesimal* y la *algorítmica* innovaciones culturales recientes —desde un punto de vista evolutivo— no es plausible que su dominio dependa de habilidades heredadas genéticamente, ni que deba desarrollarse en contextos de crianza. Dado que los *procesos psicológicos superiores avanzados* (Vygotski, 1978), correspondientes a las *funciones tipo 4* en la taxonomía de Rivière, no son universales sino *conscientes* y *voluntarios* (Rivière, 1999; Vygotski, 1978), requieren de formas de interacción específicas en contextos de enseñanza formal diseñados especialmente para tal fin. En el diseño de dichas interacciones para la enseñanza de ciencias formales, la Psicología Cognitiva del Aprendizaje está en condiciones de realizar importantes contribuciones.

Sin embargo, el sesgo parece ser bidireccional: por un lado, desde el sector tecnológico, que se presenta a sí mismo como vocero de novedades y director de cambios en la educación, se oculta la contradicción entre el aumento de los avances tecnológicos y la disminución del interés genuino por el conocimiento de sus bases de producción. Parecería que cuanto más cómodas resultan las interfaces de aprendizaje, más superficiales son los saberes que se transmiten, más separados están de las disciplinas básicas que les dieron origen y más lejos está el alumno de percibir lo aprendido como propio. Por otro lado, desde las ciencias humanas, en particular desde la psicología, los técnicos se perciben como portadores de reduccionismo, asociados al procesamiento de la información, a temidas metáforas hombre-máquina, a una revolución cognitiva que devino *cognitivismo*[2], al positivismo, al utilitarismo; a recuerdos que obstaculizan el diálogo interdisciplinar. Y ante el problema de la escasez de ingenieros, que es un problema de falta de interés *de las personas* —no de falta de artefactos— lejos de darse una respuesta interdisciplinar, se dio una respuesta tecnológica. Nuevas TIC[3] para nuevos saberes. Un reciclado cómodo pero peligroso.

La tecnología siempre avanzó más rápido que el conocimiento del ser humano, pero eso no justifica tratar los problemas humanos con soluciones Tec. Haber comprendido que la *programación estructurada* ayuda a reflejar la complejidad de la relación entre módulos globales y particulares pero no refleja el modo en que los programadores piensan los programas (Hoc, Green, Samurçay, y Gilmore, 1990) que el *pluralismo epistemológico* representa un avance aún en las ciencias de la computación (Turkle y Papert, 1990), que los diagramas de bloque implican una menor *carga cognitiva* que los de flujo (Shneiderman, Mayer, McKay, y Heller, 1977), que el análisis de ejercicios resueltos implica un aprendizaje similar o mejor que la resolución misma (Sweller, 1988; 1994), han sido descubrimientos realizados por psicólogos —junto con equipos interdisciplinares—.

Este trabajo es una invitación a otros profesionales de la tecnología a acercarse a la Psicología del aprendizaje, para trabajar juntos en las soluciones de los problemas mencionados al inicio; así como a considerar la incorporación de contenidos de las ciencias formales en la formación de los psicólogos, aquellos con orientación al psicoanálisis. Los problemas que hoy enfrentamos necesitan, además, una psicología de la conciencia.

2 "Podría escribirse un ensayo absorbente sobre la historia intelectual del último cuarto de siglo intentando averiguar qué sucedió con el impulso originario de la revolución cognitiva, cómo llegó a fraccionarse y tecnicalizarse. [...] algo que sucedió muy temprano fue el cambio de énfasis del 'significado' a la 'información', de la construcción del significado al procesamiento de la información" (Bruner, 1990).
3 Tecnologías de la Información y la Comunicación.

Si tomamos como ejemplo el cálculo proposicional, base de la algorítmica, la psicología cognitiva ha mostrado que las reglas de la deducción tradicionales no coinciden con las del pensamiento humano, sensibles a los contenidos y al contexto. La oposición tradicional entre teorías sintácticas versus teorías semánticas dio lugar a un cambio en el énfasis de la solución de los problemas, pasando de la sintaxis a la semántica, acentuando la importancia del contenido de las representaciones por sobre las reglas lógicas (Cheng y Holyoak, 1985, 1989; Cheng P., Holyoak, Nisbett, y Oliver, 1986; Cosmides, 1989; Johnson-Laird, 1975).

Marco teórico

Si bien la investigación en Psicología de la programación surge en los años setenta y continúa hasta nuestros días (ESP - *Empirical Studies of Programmers*, 2002/2017; Hoc, Green, Samurçay, y Gilmore, 1990; *Psychology of Programming Interest Group*, 1987/2017; Sajaniemi, 2008; Soloway y Spohrer, 1989; Weinberg, 1971), el acercamiento entre Informática y Psicología se ha producido paulatinamente. La mayoría de los estudios se iniciaron gracias a informáticos que desarrollaron un enfoque normativo de lo que consideraban los conceptos de programación más poderosos, hasta la obra de Weinberg (1971) quien postula que, en cualquier enfoque de la programación, debería incorporarse un punto de vista psicológico.

Sin embargo, considerar los procesos psicológicos humanos desde la teoría informática, puede resultar insuficiente, si se tiene en cuenta las grandes disparidades entre los objetivos de las Ciencias Informáticas y los de las Ciencias Humanas/Sociales. Por ejemplo, las métricas utilizadas para estructurar programas, no tienen por qué ser coherentes con las estructuras de las representaciones mentales utilizadas por los programadores. Mientras, los defensores de la programación estructurada adoptaron rígidamente la deducción *top-down* —método cartesiano— como estrategia de enseñanza (Wirth, 1976), los cambios en el paradigma en la psicología del razonamiento, específicamente el área del estudio de la deducción, mostraron que las personas tienen grados de incertidumbre tanto en las premisas como en las conclusiones y rechazan la lógica binaria como marco normativo (Evans, 2012). La evidencia experimental en psicología de la programación muestra que los programadores no abordan los programas de un modo (*de arriba hacia abajo*) sino que adoptan estrategias oportunistas, en ocasiones *bottom-up* y en ocasiones *top-down* (Hoc y Nguyen-Xuan, 1990).

Pensamiento computacional

Tras reiterados cambios de paradigmas desde los primeros lenguajes de programación hasta los actuales, antes de que pudiera resolverse este desencuentro entre teorías de la programación y teorías del aprendizaje, irrumpió el imperativo del pensamiento computacional acompañado de una serie de sugerencias, por parte del sector empresario, que se transformaron en políticas públicas en educación y agregaron más confusión de la que ya había. El "pensamiento computacional" pasó a ser considerado una habilidad fundamental en nuestro tiempo, que debe formar parte del currículum en la escuela media (Google, 2013). "Pensamiento computacional", "Uso de dispositivos", "Comunicación con los pares electrónicamente", son algunas de las habilidades básicas sugeridas a partir de los cinco años según los estándares ICSES (*International Computer Sciences Education Standards*).

Por otra parte, la programación de ordenadores dejó de ser una asignatura específica de la ingeniería informática para ser, además, parte del currículum secundario y primario. Se argumenta que los niños deben formarse para ser productores en vez de consumidores, pero dicha formación consiste precisamente en aumentar el consumo de aplicaciones en línea. Es así que nos preguntamos: ¿Qué cualidades especiales distinguen a las TICs para aprender a programar de las TICs en general? ¿Qué habilidades y qué hábitos pretende desarrollar la iniciativa de instrucción masiva presentada por las empresas a los ministerios de educación? (Code.org, s.f.; Google, 2013). ¿Conducirán efectivamente a un aumento del interés por las áreas formales y la ingeniería? ¿Qué relación existe entre el uso de multimedia y el pensamiento computacional? ¿Qué relación existe entre la "escasez de ingenieros" y la "escasez de programadores"? A continuación se consideran las cuestiones que atañen a la docencia en el nivel superior con una mirada retrospectiva hacia el origen del planteo generalista de programación para todos.[4]

El término *pensamiento computacional* fue utilizado por primera vez por Seymour Papert (1980) para referir a las ideas que pueden construirse mientras interactuamos con las computadoras como *extensiones de la mente* pero popularizado por Janette Wing con otro sentido. Para Wing, no solo los profesionales informáticos deberían saber utilizar habilidades como "abstracción, descomposición y recursividad" —y otros ítems agregados por Google (2013)[5]— sino que todas las personas deberían hacerlo, y aprenderlo desde la infancia (Wing, 2009). El salto es demasiado importante como para no despertar una actitud, cuanto menos, cautelosa, que lleve a efectuar investigaciones para lograr una mejor comprensión teórica antes de implementar programas de desarrollo a escala (Grover y Pea, 2012). No obstante, varios países ya han implementado experiencias de modificación del currículum para introducir contenidos de pensamiento computacional a edades más tempranas (Azar, Gabriela, 2014; Bargury, y otros, 2012; Grgurina, 2014; Kalelioğlu, *A new way of teaching programming skills to K-12 students*: Code.org, 2015; Lockwood y Mooney, 2017; Syslo y Kwiatkowska, 2013).

Nativos programadores

A pesar de que los sitios web, sobre pensamiento computacional y las aplicaciones TIC "facilitadoras" del aprendizaje de la programación están disponibles en Internet, y que los alumnos ingresantes a Ingeniería en Sistemas las conocen y utilizan, las dificultades que encuentran en la comprensión de los problemas algorítmicos, del cálculo proposicional booleano, del análisis matemático y el álgebra, son las mismas de antaño, o acaso hayan aumentado. Por un lado, la retórica del pensamiento computacional aumentó el nivel de abstracción e idealización de la labor del cómputo. Y por otro lado, las *app* para programar no son abstractas en absoluto sino entretenidas. La suposición de que los ingresantes a la universidad tendrán una mayor capacidad de comprensión si se exponen tempranamente a los medios puede ser errónea o insuficiente.

4 Algunas cuestiones vinculadas a los intereses del sector empresario en la formación de programadores se desarrollan en D'Angelo, V. (en prensa) La programación de ordenadores. Reflexiones sobre la necesidad de un abordaje interdisciplinar. *Revista Iberoamericana de Ciencia, Tecnología y Sociedad*. Consultado el 20 de diciembre de 2017 en <http://www.revistacts.net/>.

5 Google no solo considera las habilidades mentales sino también los productos asociados con la solución de problemas computacionales. Consultado el 14 de diciembre de 2017 en <https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview>. Para una guía de conceptos ver <https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview>. Consultado el el 14 de diciembre de 2017. .

Esto pone a los alumnos en una situación de desconcierto, cuando habiendo programado decenas de eventos en entornos visuales no son capaces de comprender la proposición lógica que subyace detrás de ese evento o de trasladar ese conocimiento a problemas de mayor envergadura. Las pruebas de escritorio a papel y lápiz les resultan demasiado lentas. Pero solo a esa velocidad es posible estar en contacto con la propia construcción del concepto a medida que ocurre. Cuando los profesores proponen resolver ejercicios con escritura *manual*, están apelando a la utilización de un recurso altamente sofisticado que aporta *feedback* sobre la transformación *mental* (no solo visual) que se está realizando, cuando el número de variables que deben manipularse es elevado. Incluso cuando los ordenadores portátiles se utilizan para tomar notas, conducen a un procesamiento más superficial y conceptualmente más pobre ya que la velocidad de escritura impide reformular la información en los propios términos (Mueller y Oppenheimer, 2014).

El nuevo alumnado ejerce una presión involuntaria que apresura la práctica digital y la eliminación de la distancia entre tiempo de análisis y tiempo de implementación, subestimando la carga cognitiva que implica la manipulación de la aplicación misma y el hecho de que la percepción visual no garantiza la apropiación, (Knobelsdorf, Isohanni, y Tenenberg, 2012) ni la interacción. Sobre una distinción entre *aprendizaje activo*, *constructivo* o *interactivo* ante los medios véase (Chi M. T., 2009; Chi y Wylie, 2014).

Considerando que muchos jóvenes que optan por carreras de Tecnología de la Información (TI) lo hacen motivados por su interacción previa con el ordenador en el hogar bajo el lema *nativos digitales*,[6] que contribuyó a la naturalización de las consecuencias del *consumo de medios* a edades tempranas, no debe sorprender que la mayoría de los alumnos que ingresan al ciclo universitario, hayan experimentado e incluso automatizado rutinas con aplicaciones visuales de programación, antes de haber comprendido lo que significa resolver un problema. Las respuestas se dan "sin pensar", esto es, antes de advertir en qué consiste realmente el problema a resolver, de identificar las variables, de tomar conciencia de la información que se tiene delante.

La programación de computadoras se presenta como una competencia importante para el desarrollo de habilidades de resolución de problemas, pero se confunde el *pensamiento* con la *habilidad* de operar un artefacto. El uso del artefacto no conduce *per se* a un progreso en el pensamiento. Algunos estudios realizados con alumnos de primaria muestran, por ejemplo, que la programación en la plataforma Scratch[7] no causó diferencias significativas en las habilidades de resolución de problemas de los estudiantes de primaria. Solo hay un aumento no significativo en la media del factor de "confianza en sí mismos en su habilidad para resolver problemas" (Kalelioğlu, 2015; Kalelioğlu y Gülbahar, 2014).

6 Prensky, M. (2001). *Digital Natives, Digital Immigrants Part 1. On the Horizon*, 9(5), 1-6. Consultado el 10 de diciembre de 2017 en <https://doi.org/10.1108/10748120110424816>.

7 Scratch es una aplicación visual interactiva para diseñar animaciones y juegos inserta en una red social. Por las características de algunos de sus objetos incrustados, que representan estructuras de iteración, decisión y secuencia, se lo suele considerar un lenguaje de programación.

¿Es posible enseñar a pensar en este contexto?

Históricamente, la enseñanza de programación en la universidad y en las tecnicaturas estuvo ligada al objetivo de analizar y resolver problemas en un espacio distante y en un tiempo diferido de la implementación, ya que los problemas que surgen en la implementación —compilación, depuración— son de una naturaleza distinta a los que surgen durante el análisis de la situación problema en el mundo real, por tanto, conducen a desarrollar habilidades diferentes.

Durante el surgimiento de las primeras carreras terciarias en las décadas de los setenta u ochenta, cuando no existía la formación universitaria para la computación, el título ofrecido era el de Analista Programador y le seguía el de Analista de Sistemas. La concepción que se tenía de las incumbencias del título incluía el *análisis* en primer lugar. Por las mismas peculiaridades de la labor, no se concebía a un programador que no supiera analizar.

Con la masificación de los productos de *software* y el aumento del tamaño de los programas, el tiempo invertido en modificaciones por depuración o por actualización de versiones supera ampliamente al tiempo invertido en diseño. Y las habilidades necesarias para la depuración, no coinciden con las de análisis y resolución de problemas. La formación para la resolución de problemas es muy diferente a la adquisición de habilidades de depuración propias del mantenimiento del *software*.[8]

La importancia de esta afirmación reside en que contribuye a esclarecer la diferencia entre dos propuestas de formación actualmente vigentes, una orientada hacia la ciencia de la computación en sentido amplio y a despertar vocaciones universitarias, otra dirigida a la programación de pequeñas aplicaciones de escaso valor agregado en la cadena del *software*, más cercano al consumo que a la producción.[9] La confusión relativa a estas iniciativas proviene de que todas se presentan ante los medios como *la* solución al problema futuro de la escasez de profesionales del *software* (Code.org, s.f.; Google, 2013).[10]

El análisis de los intereses que subyacen a tales propuestas y sus consecuencias para la educación, no se tratarán en este trabajo. Pero es importante mencionar que las alternativas difieren en un punto esencial: *la edad* a la cual los niños deben iniciarse en la manipulación de medios de pantalla. Mientras un sector sostiene el ya clásico "cuanto antes mejor", el otro afirma, con fundamento en la teoría de la computación, que los contenidos fundamentales y trasversales de la formación para la ingeniería en general y para las ciencias de la computación en particular, no están asociados, ni dependen, ni mejoran necesariamente con el uso de dispositivos:

> "La *ciencia* de la computación trata fundamentalmente sobre algoritmos, recetas para resolver problemas y realizar tareas. De la misma manera en que los niños pueden aprender acerca de los dinosaurios sin hacer excavación de huesos y sobre los planetas y el espacio sin observar los telescopios, el núcleo intelectual de la informática no depende de las máquinas para su presentación. Al igual que

8 Para el mantenimiento se necesitan, sobre todo, habilidades de *depuración*. Esta consiste en el proceso de corrección de errores. "La depuración es una de las partes más frustrantes de la programación". "Por desgracia, hay evidencia de que las destrezas para la depuración son innatas en el ser humano. Ciertas personas son buenas para ella; otras no." Schneiderman (1980) cito en Pressman, R. (2010). *Ingeniería del Software* (Sexta ed.). Méjico: McGraw Hill.

9 Sobre este punto puede verse "Dos maneras de aprender a programar" en D'Angelo, V. (en prensa).

10 Consultado el 14 de diciembre de 2017 en <https://www.youtube.com/watch?v=m1UnzPnJe6E>. Y consultado el 6 de agosto de 2017 en <https://www.youtube.com/watch?time_continue=27&v=VpDDJPWVn5-Q>.

> con esas otras temáticas, un enfoque basado en narraciones, actividades y materiales cotidianos puede ser más vívido y atractivo que los enfoques que hacen de las computadoras un fetiche. (...) Proponemos que los principios de la adquisición del lenguaje deberían ser aplicados a la enseñanza de las ciencias matemáticas, y revisar cómo estos principios se han aplicado previamente a la enseñanza de la lectura y escritura. (...)El núcleo intelectual de la ciencia de la computación puede ser presentado a los niños incluso en situaciones donde no hay computadoras. (...) Muchas de las ideas nucleares de la ciencia de la computación se introducen mejor sin computadoras". (Fellows, 1991)

Proponen un cambio de perspectiva: pensar el aprendizaje como basado en problemas (De Graaf y Kolmos, 2003), situar al alumno en contextos cotidianos, priorizar el significado a la información en un recorrido desde la semántica hacia la sintaxis en dirección opuesta a la enseñanza tradicional (Bell, Rosamond, y Casey, 2012; Fellows, 1991; Fellows y Parberry, 1993; Rapaport, 2015). Aquí algunos ejemplos:

Figura 1: la red de ordenación ubicua. En Tokyo (a).

Figura 2: en India (b)

Figura 3: en Japón (c). Fuente: (Bell, Rosamond, y Casey, 2012)

Figura 4: algoritmo de búsqueda dicotómica (d). Fuente: crumpluggod.org

Otra respuesta

Tal vez, sea necesario indagar con mayor detenimiento cuáles son los verdaderos factores que conducen a la pérdida del interés por las carreras técnicas, de modo que las decisiones acerca del currículum escolar no se tomen solo desde el punto de vista empresarial con énfasis en el acceso a los medios

digitales sino, desde una perspectiva que aborde el mismo proceso educativo. La solución de problemas implica distinguir fases, como la de *representación del problema* en estado actual o punto de partida y un tiempo de desarrollo en cada una de ellas (Carretero y Asensio, 2008; Holyoak y Morrison, 2005; Newell y Simon, 1970; Pérez Echeverría, 2008; Polya, 1757).

Según Chi (1989), la mayoría de las teorías sobre resolución de problemas se han focalizado en el proceso de compilación (Anderson, 1987), relegando a un segundo plano el subproceso de construcción de la representación, sin embargo, propone prestar atención a esta instancia, ya que se han observado diferencias individuales en el rendimiento de los alumnos asociadas a diferencias en el modo de construcción de la representación del problema (Chi, Bassok, Lewis, Reimann, y Glaser, 1989; Renkl, 1997). Varios estudios muestran que aquellos alumnos que *invierten más tiempo en explicarse a sí mismos* aquello que intentan comprender, son más eficaces en la resolución de problemas y en la adquisición de nuevos conceptos (Aureliano, Tedesco, y Caspersen, 2016; Chi, Bassok, Lewis, Reimann, y Glaser, 1989; Kastens y Liben, 2007; Mitchell, Mertz, y Ryant, 1994)

En la solución de problemas algorítmicos en particular, si bien las fases no son estrictamente secuenciales, sino recursivas (Pennington y Grabowski, 1990), hemos notado que la comprensión del problema en su dominio, suele confundirse con la fase de diseño del algoritmo. Una diferencia entre novatos y expertos en cuanto al abordaje de los problemas suele ser que la automatización de soluciones, el uso de esquemas y la rapidez que manifiestan los expertos, los lleva a presentar el problema completo identificándolo con su fase de cómputo (que es solo una fase del mismo), salteando una etapa del proceso y resolviendo de forma más rápida, eficiente pero poco útil a la enseñanza.

Los alumnos, en cambio, necesitan más tiempo de elaboración en la fase declarativa, durante la cual, deberían probar modos de explicar los problemas en sus propios términos, utilizando sus saberes previos. A menudo, los profesionales que dictan asignaturas procedimentales, suelen acelerar el proceso induciendo a los estudiantes a una automatización prematura, confundiendo "ejercicios" con "problemas". Sobre esta distinción, ver Pérez Echeverría y Pozo, 1994, cito en (Carretero y Asensio, 2008).

Sabemos sin duda, que el cálculo proposicional no evolucionó por selección natural. De los dos mil quinientos años (2500 años) de lógica clásica solo los últimos ciento setenta (170 años) han sido testigos de la lógica binaria. El álgebra booleana fue introducida por George Boole en su primer libro *El análisis matemático de la lógica* en (1847), ha sido fundamental para el desarrollo de la electrónica digital e integra cualquier lenguaje moderno de programación, sin embargo no podemos afirmar, como sugiere Boole en *El lenguaje del pensamiento* (1854), que sea un reflejo del modo en que razonan los seres humanos. Autores contemporáneos asumen que las personas rechazan la lógica binaria como un sistema normativo viable (Evans, 2012). El esfuerzo de Boole consistió en simplificar y sistematizar la lógica clásica aristotélica suponiendo que sintetizaba las "operaciones mentales" que le daban sustento. Ese recorrido "desde el lenguaje hacia el cálculo", aunque no podamos precisar qué tipo de esquemas pone en juego, está comandado por la expresión verbal, que es además, el recurso más utilizado en la argumentación. Desde el origen de nuestra cultura, el surgimiento de la razón no implicó ruptura con el pensamiento literario (Vernant, 1962), más bien los sistemas explicativos argumentativos surgieron a partir de los sistemas explicativos narrativos y entrelazados con ellos (Márgico, 2011).

Ayudas semánticas

Como actividades exploratorias para el diseño experimental, hemos efectuado dos pruebas de razonamiento proposicional. La presentación de un problema en formato narrativo con contenido proposicional subyacente y una variación del test de Wason, respectivamente.

Actividad 1: sistemas explicativos argumentativos

Durante la primera prueba, se proyectó a alumnos ingresantes al ciclo básico de Ingeniería en Sistemas Informáticos un capítulo de la Serie *The Big Ban Theory* (1) y se les entregó una explicación formato narrativo (2), a partir de la cual pudieran extraer reglas de inferencia.[11] Nos basamos en los siguientes supuestos: i) Cualquier soporte de la actividad que les aportara *feedback* sobre los pasos que iban siguiendo o que contribuyera a extender el tiempo que dedicaban a la representación del problema para sí mismos, sería facilitador de la tarea. ii) Si bien en su aplicación las reglas lógicas no dependen del contenido para la validez de sus premisas, en las prácticas de enseñanza, las operaciones de conjunción, disyunción, condicional, se comprenden mejor cuando se ejemplifican mediante un esquema psicológico: la *situación disyuntiva*, la *situación conjuntiva*, la *toma de decisiones* en situaciones concretas.

Ejemplo: fragmento de la actividad 1: Sheldon y sus amigos deben decidir en qué cine verán una película. Han representado en el plano todos los cines de la ciudad, los restaurants cercanos, los horarios y otras condiciones que consideran importantes. En uno de ellos proyectan la película a las 7:20, en otro a las 7:40, en otro a las 8:10, y en el cuarto a las 8:45. Según Sheldon, dos de los cines quedarían eliminados porque no tienen *máquinas de granizado*. Leonard no coincide, piensa que los cines tachados son aceptables porque tienen *proyección digital* y *sonido surround* (envolvente). Cada uno de los personajes expresa sus condiciones para ir al cine, la postura más inflexible es la de Seldon (...)

p simboliza "El cine es aprobado por Leonard"
r simboliza "El cine es aprobado por Howard"
s simboliza "El cine es aprobado por Sheldon"
x simboliza "El cine es aprobado por los cuatro"
Si se cumplen *p* Y *q* Y *r* Y *s* ENTONCES *s* se cumple *x* (donde *x* sería una conclusión).

Figura 5: Tabla de verdad de situaciones posibles

p	q	r	s	x
V	V	V	V	V
V	V	V	F	F
V	V	F	V	F
V	V	F	F	F
V	F	V	V	F
V	F	V	F	F
V	F	F	V	F
V	F	F	F	F
F	V	V	V	F
F	V	V	F	F
F	V	F	V	F
F	V	F	F	F
F	F	V	V	F
F	F	V	F	F
F	F	F	V	F
F	F	F	F	F

Situación ideal. Los cuatro de acuerdo

Situación actual

En la actividad completa se exponen detalladamente las propuestas de cada personaje y las tablas de verdad asociadas.

11. *El problema irresoluble*. Consultado el 14 de diciembre de 2017 en <https://www.youtube.com/watch?v=W9iaqd6o2C7&t>

Actividad 2: variación del Test de Wason

• **Variación 1:**
i). Cada tarjeta representa una premisa. De un lado, el sujeto y del otro, el predicado.
ii). Argumento vinculado a la justicia.
iii). Se les pide a las personas que imaginen estar en una discusión con el que afirma que "Todos los buenos van al cielo" y se propongan refutar ese argumento.

• **Variación 2:**
i). Se presenta la discusión entre el **AFIRMADOR** y el **NEGADOR**. El primero, asegura que "Todos los buenos van al cielo", el segundo dice que no es verdad. Se les pregunta a las personas quién tiene razón y qué tarjetas lo confirman.

• **Materiales**: una hoja con la siguiente especificación:

Cada tarjeta representa una afirmación. De un lado está el sujeto y del otro el predicado.

Sujeto	Sujeto	Predicado	Predicado
TODOS LOS BUENOS	TODOS LOS GLOBOS	VAN AL CIELO	VAN AL INFIERNO

El AFIRMADOR sostiene que TODOS LOS BUENOS VAN AL CIELO, es decir, que Cada vez que una tarjeta dice TODOS LOS BUENOS de un lado, del otro lado dice VAN AL CIELO.

El NEGADOR sostiene que no es cierto.

Responde

A) ¿Quién crees que tiene razón?
B) ¿Cuál es el número mínimo de tarjetas que deberías dar vuelta para demostrarlo?

Respuesta

A) .....

B) .....

Conclusiones

Si bien, se trató de una exploración previa a la toma de decisiones sobre el diseño experimental de la tesis, algunos alumnos reportaron que la actividad 1 les había resultado "fácil", no tuvieron problemas para resolverla. Mientras que en el test de Wason, aunque se trató de una variación con contenido semántico, manifestaron gran dificultad en comprender la consigna y en pensar la respuesta. Es posible que con una temática vinculada a esquemas de permiso u obligación, la tarea fuera facilitada. No obstante, parece claro que la narración es el vehículo preferido del argumento y este del razonamiento.

Bibliografía

Anderson, J. (1987). Skill Acquisition: Compilation of Weak-Method Problem Solutions. *94*(2), 192-210.

Aureliano, V., Tedesco, P., y Caspersen, M. (2016). Learning programming through stepwise self-explanations. *Information Systems and Technologies (CISTI), 2016 11th Iberian Conference on*, 1-6.

Azar, Gabriela. (2014). Diseño Curricular para la Nueva Escuela Secundaria. Ciclo Básico. 2014-2020. En G. Azar (Ed.). Ciudad Autónoma de Buenos Aires.

Bargury, I. Z., Muller, O., Haberman, B., Zohar, D., Cohen, A., Levy, D., y Hotoveli, R. (2012). Implementing a New Computer Science Curriculum for Middle School in Israel. *Proceedings of Frontiers in Education Conference* , 1-6.

Bell, T., Rosamond, F., y Casey, N. (2012). Computer Science Unplugged. En H. L. Bodlaender, R. Downey, F. V. Fomin, y D. Marx (Edits.), *The Multivariate Algorithmic Revolution and Beyond. Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday* (págs. 398-456). Berlín: Springer. doi:10.1007/978-3-642-30891-8.

Boole, G. (1847). *The Mathematical Analysis of Logic, Being an Essay Towards a Calculus of Deductive Reasoning.* Cambridge: Henderson y Spalding.

Boole, G. (1854). *The Laws Of Thought On Wich Are Founded The Mathematical Theories Of Logic And Probabilities.* Cambridge: University Press.

Bruner, J. (1990). El estudio apropiado del hombre. En J. Bruner, *Actos de significado. Más allá de la revolución cognitiva* (págs. 19-46). Madrid, España: Alianza.

Carretero, M., y Asensio, M. (Edits.). (2008). *Psicología del Pensamiento.* Madrid: Alianza.

Cheng, P. W., y Holyoak, K. J. (1989). On the natural selection of reasoning theories. *Cognition* , *33*, 285-313.

Cheng, P., y Holyoak, K. (1985). Pragmatic Reasoning Schemas. *Cognitive Psychology*, *17*, 391-416.

Cheng, P., Holyoak, K., Nisbett, R., y Oliver, L. (1986). Pragmatic versus Syntactic Approaches to Training Deductive Reasoning. *Cognitive Psychology*, *18*, 293-328.

Chi, M. T. (2009). Active-Constructive-Interactive: A Conceptual Framework. *Topics in Cognitive Science*, *1*(73-105). doi:10.1111/j.1756-8765.2008.01005.x

Chi, M. T., y Wylie, R. (2014). The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes. *Educational Psychologist, 49*(4), 219-243. doi:10.1080/00461520.2014.965823

Chi, M., Bassok, M., Lewis, M., Reimann, P., y Glaser, R. (1989). Self Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science, 13*, 145-182.

Cosmides, L. (1989). The logic of social exchange: Has natural selection shaped how humans reason? Studies with the Wason selection task. *Cognition, 31*, 187-276.

D'Angelo, V. S. (en prensa). La programación de ordenadores. Reflexiones sobre la necesidad de un abordaje interdisciplinar. *Revista Iberoamericana de Ciencia, Tecnología y Sociedad*. Consultado el 20 de diciembre en <http://www.revistacts.net/>.

De Graaf, E., y Kolmos, A. (2003). Characteristics of Problem-Based Learning. *International Journal of Engineering Education, 19*(5), 657–662.

ESP - Empirical Studies of Programmers. (2002/2017). Consultado el 20 de diciembre de 2017 en <https://www.interaction-design.org/literature/conference_series/esp>.

Evans, J. S. (2012). Questions and challenges for the new psychology of reasoning. *Thinking and Reasoning, 18*(1), 5-31.

Fellows, M. (1991). Computer science in the elementary schools. En N. Fisher, H. Keynes, y P. Wagreich (Edits.), *Proceedings of the Mathematicians and Education Reform Workshop* (págs. 143 -163).

Fellows, M., y Parberry, I. (1993). SIGACT trying to get children excited about CS. *Computing Research News, 5*(1), 7.

Google. (2013). *Exploring Computacional Thinking/ CT Overview*. Consultado el 20 de diciembre de 2017 en <https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview>.
Grgurina, N. B. (2014). Computational thinking skills in dutch secondary education: exploring pedagogical content knowledge. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* , 173-174 .

Grover, S., y Pea, R. (2012). Computational Thinking in K–12: A Review of the State on the Field. *Educational Researcher, 42*(1), 38-43.

Haak, S. (1978). 'Philosophy of Logics'. En S. Haak, *Philosophy of Logics* (págs. 1-10). Cambridge: Cambridge University Press.

Hoc, J. M., y Nguyen-Xuan, A. (1990). Language Semantics, Mental Models and Analogy. En J. -M. Hoc, T. R. Green, R. Samurçay, y D. J. Gilmore (Edits.), *The Psychology of Programming*.

Hoc, J. -M., Green, T. R., Samurçay, R., y Gilmore, D. J. (1990). Part 1: Theoretical and Methodological Issues. En J. -M. Hoc, T. R. Green, R. Samurçay, y D. J. Gilmore (Edits.), *Psychology of Programming*.

Hoc, J. -M., Green, T. R., Samurçay, R., y Gilmore, D. J. (Edits.). (1990). *The Psychology of Programming*. San Diego, CA: Academic Press.

Holyoak, K., y Morrison, R. (Edits.). (2005). *The Cambridge Handbook of Thinking and Reasoning*. New York: Cambridge University Press.

International Computer Sciences Education Standards. (s.f.). Consultado el 1 de Mayo de 2017 en <https://docs.google.com/spreadsheets/d/1SE7hGK5CkOlAf60EnqkoDPr8OOSdyGZmRnROhr0XHys/edit#gid=2118360934>.

Johnson-Laird, P. (1975). *Mental Models. Towards a Cognitive Science on Language, Inference and Consciousness*. Cambridge: Cambridge University Press.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200-210. doi:10.1016/j.chb.2015.05.047

Kalelioğlu, F., y Gülbahar, Y. (Enero de 2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics in Education, 13*(1), 33-50.

Kastens, K., y Liben, L. (2007). Eliciting Self-Explanations Improves Children's Performance on a Field-Based Map Skills Task. *Cognition and Instruction, 25*(1), 45–74.

Knobelsdorf, M., Isohanni, E., y Tenenberg, J. (Noviembre de 2012). The Reasons Might Be Different - Why Students and Teachers Do Not Use Visualization Tools. *Proceedings - 12th Koli Calling International Conference on Computing Education Research, Koli Calling 2012*, 1-10. doi:10.1145/2401796.2401797

Lockwood, J., y Mooney, A. (Marzo de 2017). Computational Thinking in Education: Where. *ResearchGate*.

Mársico, C. (2011). Ejes para pensar lo griego. En C. Mársico (Ed.), *Polythrýleta. Sistemas explicativos y mutación conceptual en el pensamiento griego* (págs. 13-41). Buenos Aires: Rhesis.

Mitchell, N., Mertz, K., y Ryant, R. (Abril de 1994). Learning Through Self-Explanation of Mathematics Examples: Effects of Cognitive Load. *Paper presented at the Annual Meeting of the American Educational Research Association*.

Mueller, P. A., y Oppenheimer, D. M. (23 de Abril de 2014). The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking. *Psychological Science, 25*(6). doi:10.1177/0956797614524581

Newell, A., y Simon, H. (Febrero de 1970). Human Problem Solving. *American Psychologist, 26*(2), 145-159.

Papert, S. (1980). *Mindstorms*. New York: Basic Books.

Pennington, N., y Grabowski, B. (1990). The Tasks of Programming. En *Psychology of Programming* (págs. 45-62). Academic Press.

Pérez Echeverría, M. P. (2008). Solución de Problemas. En M. Carretero, y M. Asensio (Edits.), *Psicología del Pensamiento* (págs. 199-218). Madrid: Alianza.

Polya, G. (1757). *How to solve it. A New Aspect of Mathemtical Method* (Segunda ed.). New Yersey: Princeton University Press.

Prensky, M. (2001). Digital Natives, Digital Immigrants. *On the Horizon, 9*(5).

Psychology of Programming Interest Group. (1987/2017). Consultado el 20 de diciembre en <http://www.ppig.org/>.

Rapaport, W. J. (2015). *Philosophy of Computer Science*. The State University of New York.

Renkl, A. (1997). Learning from Worked-Out Examples: A Study on Individual Differences. *Cognitive SCience, 22*(1), 1-29.

Rivière, Á. (1999). Desarrollo y educación: el papel de la educación en el "diseño" del desarrollo humano. En M. Belinchón, A. Rosa, M. Sotillo, y I. Marichalar, *Ángel Rivière. Obras Escogidas* (Vol. III, págs. 203-242). Buenos Aires.

Sajaniemi, J. (Mayo de 2008). Psychology of Programming: Looking into Programmers' Heads. *Human Technology. An Interdisciplinary Journal on Humans in ICT Environments, 4*(1), 4-8.

Shneiderman, B., Mayer, R., McKay, D., y Heller, P. (Junio de 1977). Experimental Investigations of the Utility of Detailed Flowcharts in Programming. *Communications of the ACM, 20*(6), 373-381 . doi:10.1145/359605.359610

Soloway, E., y Spohrer, J. (Edits.). (1989). *Studying the Novice Programmer*. New York: Psychology Press.

Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science, 12*, 257-285. Consultado el 20 de diciembre de 2017 en <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.459.9126&rep=rep1&type=pdf>.

Sweller, J. (1994). Cognitive Load Theory, Learning Difficulty, and Instructional Design. *Learning and Instruction, 4*, 293-312. Consultado el 20 de diciembre de 2017 en <http://coral.ufsm.br/tielletcab/Apostilas/cognitive_load_theory_sweller.pdf>.

Syslo, M. M., y Kwiatkowska, A. B. (Febrero de 2013). Informatics for all high school students: A computational thinking approach. *Proceedings of the 6th international conference on Informatics in Schools: Situation, Evolution, and Perspectives*.

Turkle, S., y Papert, S. (1990). Voces y estilos en la cultura computacional (citar). En S. Turkle, y S. Papert, *Pluralismo epistemológico*.

UNESCO. (2010). *Engineering: Issues, Challenges and Opportunities for Development*. UNESCO Publishing.

UNESCO. (2011). *UNESCO Engineering Initiative*. Consultado el 20 de diciembre de 2017 en <http://www.unesco.org/new/en/natural-sciences/science-technology/engineering/unesco-engineering-initiative/>.

Vernant, J. P. (1962). Del mito a la razón (Capítulo VII). En J. P. Vernant, *Los orígenes del pensamiento griego* (págs. 334-364). Barcelona: Paidós.

Vygotski, L. S. (1978). *El desarrollo de los procesos psicológicos superiores*. Barcelona: Crítica.

Weinberg, G. (1971). *The Psychology of Computer Programming*. New York: Van Nostrand.

Wing, J. (Marzo de 2006). Computational Thinking. *Communications of the acm M, 49*(3), 33-35.

Wing, J. (Julio de 2009). Computational thinking and thinking. *Philosophical Transactions of the Royal Society, 366*, 3717–3725.

Wirth, N. (1976). *Algorithms and Data Structures*. Canada: Pearson Education.