Retruco: un intérprete para TIMBA

Alvaro Frías Garay

FAMAF-Universidad Nacional de Córdoba, Córdoba, Argentina
alvaro.frias.garay@mi.unc.edu.ar

Abstract. Presentamos Retruco, un intérprete para TIMBA (Terrible Imbecile Machine for Boring Algorithms); un lenguaje de programación creado en Argentina durante la decada de 1980 y que fue utilizado en varios puntos del país principalmente en materias de ciclo básico, o primeros años de una carrera en Ciencias de la Computación, como herramienta para introducir a la programación estructurada. Analizamos también herramientas similares a TIMBA.
Se presenta una investigación respecto a los actores involucrados en su creación y que tuvieron contacto directo con el lenguaje además del contexto previo a la creación del lenguaje, y su uso posterior.

Keywords: Lenguaje · programación · intérprete · historia · enseñanza Ciencias Computación Argentina · programación desenchufada

1 Introducción

La preservación de los lenguajes se vuelve necesaria no solo en documentos con especificaciones, dado que esta nueva forma de literatura incorpora, además de los tradicionales emisor y receptor, de un tercer actor: la computadora [3]. Por lo que en este trabajo se hace un esfuerzo en reconstruir el contexto histórico donde el lenguaje se creó y utilizó y, sumado a la falta de un compilador/intérprete disponible al público, se presenta el intérprete Retruco, el cual sigue las especificaciones originales del lenguaje, adoptándolas a las tecnologías actuales.

2 Trabajos Relacionados

Dadas herramientas educativas para la introducción a la computación como son CARDIAC y Little Man Computer (LMC) que se originaron en la década de 1960 pero que comparten rasgos significativos con TIMBA, como ser herramientas básicas para enseñar conceptos de programación y computación, así como no contar con una computadora digital convencional para funcionar, ambas cuentan con una gran cantidad de simuladores implementados en computadoras actuales para implementar programas en dichos modelos [16] [17].

La principal contribución de este trabajo, al igual que los simuladores mecionados anteriormente de los lenguajes similares a TIMBA, es proporcionar un software para preservar el lenguaje y que sea posible utilizarlo hoy día.

3 Contexto Histórico

La historia de los lenguajes de programación en Argentina se inicia junto con la llegada al país de la computadora Ferranti Mercury II, también conocida como *Clementina*, la cuál contaba con un lenguaje de programación de alto nivel conocido como **AUTOCODE**, el cual era un lenguaje muy próximo al lenguaje matemático que servía para abstraerse de los detalles técnicos de la máquina en sí, dado que usaba un modelo de máquina ideal [9]. Posteriormente, debido a la poca capacidad expresiva del lenguaje, sobretodo en lo que concierne a cálculo de matrices, se creó el que puede ser considerado el primer lenguaje de programación argentino: **COMIC** (COMpilador del Instituto de Cálculo) [19].

Con la Dictadura cívico-militar de 1966 y tomando como punto de partida el trágico hecho de La Noche de los Bastones Largos, debido a los exilios de científicos e investigadores y la intervención militar a las universidades llevó a que la investigación que se aplicaba en computación en Argentina se viera pausada. Tomemos como ejemplos que el desarrollo de **COMIC** se viera detenido y nunca pudiera completarse, la falta de actualización en el Instituo de Cálculo, llevando a Clementina a una sobrevida y decadencia, y los escasos grupos de investigación durante la dictadura, que servían como apoyo a otras áreas como Economía o Física, y no sería hasta una década después que podría retomarse el curso de investigación en Ciencias de la Computación [1].

El documento más antiguo que se ha encontrado que menciona al lenguaje TIMBA es un cuadernillo de Computación para el CBC de la UBA, donde se especifica que el lenguaje fue creado por un equipo de trabajo de la Universidad Nacional de San Luis, dirigido por el profesor Ing. Hugo Ryckeboer [4] como lenguaje para la enseñanza de la programación.

Es importante resaltar el contexto de la computación a nivel mundial en dicha época. La década de 1980 resultó en el boom de la microcomputación, Companías como Apple e IBM en Estados Unidos comenzaban a comercializar las Apple II y PC, así como sus contrapartes inglesas Acorn y Sinclair vendían las Acorn Atom y ZX80 respectivamente, todas las computadoras listadas contaban con un intérprete de BASIC. En Argentina la clase media podía acceder a ellas aunque estaba más apuntado a hobbistas. Esto puede explicar la principal peculiaridad del lenguaje; el uso de un mazo de cartas españolas, elemento común que se puede encontrar en cualquier casa, como principal fuente de datos para el lenguaje, y que el UCP, también llamado ejecutor i.e. el elemento encargado de ejecutar las ordenes que se conforman a partir de las reglas del lenguaje, sea uno mismo, [6] como lo muestra la Figura 1. Aunque BASIC y TIMBA compartieron época los dos lenguajes se encontraron en veredas opuestas en cuestiones fundamentales de su construcción: primeramente BASIC fue pensado como un lenguaje de próposito general de alto nivel para las computadoras de tiempo compartido de Darthmouth [12], mientras que TIMBA fue creado como un lenguaje de enseñanza, con construcciones más limitadas aunque igualmente expresivas y un lenguaje del que no hace falta una computadora digital para escribir en él. Otra diferencia notable es que TIMBA se base en el paradigma de programación estructurada, el cual se inició en la década de 1950 con ALGOL,

Fig. 1. Una representación del UCP de TIMBA como figura en [5]

y hace uso de las estructuras principales de condición y repetición, mientras que BASIC, si bien tenía una estructura de for, también hacía uso de los go to y el if se usaba como un salto condicional, prácticas del lenguaje que ya para la época se consideraban dañinas [13].

TIMBA puede considerarse como un lenguaje enmarcado dentro de los *CS Unplugged*[1], es decir, un conjunto de actividades que se utilizan para enseñar conceptos de Ciencias de la Computación sin el uso de una computadora digital [11]. Es de notar que TIMBA es un precursor de dichas actividades dado que su formalización se dio en la década de 1990, por lo menos diez años después de la aparación del lenguaje argentino, aunque comparte el mérito con otros lenguajes unpluggled como lo son CARDIAC y Little Man Computer (LMC).

CARDIAC fue creada en los laboratorios Bell en 1969 como una herramienta para demostrar el funcionamiento interno de una computadora. Consistía de una cartulina con piezas móviles que representaba una computadora de arquitectura Von Neumann, con CPU, memoria, I/O y un set de instrucciones, contando con mnemónicos para escribir programas en lenguaje ensamblador [20] [14].

LMC presenta un paradigma similar a CARDIAC pero variando esta vez en que la computadora se presenta como una habitación con un hombrecito el cual cuenta con una calculadora de bolsillo para hacer cálculos y cajas de correo donde almacenar números, además de dos cajas donde recibe instrucciones mediante códigos de operación (opcodes) y donde deposita resultados. [15]

Retomando TIMBA, durante esa época el lenguaje se portó a la computadora PECOS en un proyecto educativo que involucraba hardware, software y material didáctico [7]. Se ha encontrado evidencias de un libro titulado "Programa Educativo Para Colegios Secundarios. Area Computación", creado por Viviana Rubinstein el cual contaba con un diskette de "TIMBA", posiblemente un intérprete del lenguaje para la computadora, aunque no se ha podido recuperar dicho diskette [18][2].

TIMBA continuó siendo usada en el CBC de la UBA por un par de años y también se incorporó como herramienta la Universidad Nacional de Río Cuarto.

Finalmente el lenguaje persistió en la Universidad Nacional de San Luis, hasta 2019 [6] como un pseudo-lenguaje simple que permitía la definición de algoritmos basado en el paradigma de la programación estructurada para la materia Programación I [2].

En la actualidad no hay referencias al lenguaje en el material de estudio de la materia [8].

4 Lenguaje e Intérprete

4.1 TIMBA

El lenguaje TIMBA está estructurado y tiene como única estructura de datos a la pila. Un programa típico de TIMBA se divide en dos: una parte de instrucciones al UCP y la otra encargada de definir las pilas que habrá al principio del

[1] En español, Computación desenchufada

[2] Agradecemos a Gustavo del Dago por aportar la referencia

programa y las cartas, si las hay, en cada una de ellas. Además de las pilas, el UCP en cualquier momento puede tener una carta en su "mano", una analogía de un registro, tomadas de una de las pilas, por lo que siempre comienza con la mano vacía.

El lenguaje cuenta con instrucciones operativas para mover cartas de la mano a las pilas y viceversa, e invertir la posición de la carta que se tiene en mano, pudiendo estar boca arriba o boca abajo. Cuenta también con instrucciones de condición (SI) y de repetición (MIENTRAS), ambas construcciones que pueden encontrarse en un lenguaje imperativo típico, contando los dos tipos de instrucciones nombradas anteriormente con proposiciones para obtener valores de verdad sobre la carta que se tiene en mano: la posición de la carta (boca arriba o boca abajo); el valor numérico de la carta comparándolo con un valor dado o con el valor de la carta que está en el tope de una pila; el palo de la carta, pudiendo hacer comparaciones similares a las del valor numérico: comparándolas con un palo dado o con el palo de una carta que está en el tope de una pila, así como también proposiciones sobre el estado de una Pila dada: si está o no vacía. Dadas las proposiciones anteriores estas pueden combinarse usando los operadores de Conjunción (Y) y de Disyunción (O).

El lenguaje puede generar errores e.g. tomar una carta de una pila vacía, invertir la carta que se tiene en mano cuando no hay una carta en mano, evaluar una proposición sobre la carta cuando no se tiene una carta o esta está boca abajo, etc.

Para una especificación completa del lenguaje se refiere a [5].

A continuación se presentan dos problemas resueltos en el lenguaje. El primero es un problema que sirve para mostrar todas las construcciones que posee el lenguaje. El segundo es un problema tomado de [4].

El primer problema es así: dada una pila de cartas llamada "PRINCIPAL" de la que se sabe que solo contiene cartas de palo Espada u Oro y, sin saber si están boca arriba o boca abajo, hay que repartir las cartas en dos pilas, llamadas "DE ESPADAS" y "DE OROS". En el programa se da una pila predeterminada aunque el programa es válido para cualquier pila de cartas que solo contengan cartas de palo de Oro o Espada en cualquier posición.

```
DEFINICION DE PROGRAMA
MIENTRAS LA PILA PRINCIPAL NO ESTA VACIA
    TOME UNA CARTA DE LA PILA PRINCIPAL
    SI LA CARTA ESTA BOCA ABAJO
        INVIERTA LA CARTA
    SINO
    NADA MAS
    SI LA CARTA ES DEL PALO ORO
        DEPOSITE LA CARTA EN LA PILA DE OROS
    SINO
        DEPOSITE LA CARTA EN LA PILA DE ESPADAS
    NADA MAS
```

```
REPITA

UCP EJECUTE CON LAS SIGUIENTES CARTAS
PILA PRINCIPAL TIENE 1 DE ORO ↑ - 1 DE ESPADA ↑ -
                2 DE ORO ↑ - 2 DE ESPADA ↑ -
                7 DE ORO ↑ - 7 DE ESPADA ↑
PILA DE OROS NO TIENE CARTAS
PILA DE ESPADAS NO TIENE CARTAS
```

Las sentencias de "DEFINICION DE PROGRAMA" y "UCP EJECUTE CON LAS SIGUIENTES CARTAS" sirven como identificadores de las secciones del programa mencionadas anteriormente.

El segundo problema dice así (sic):

> Dada una pila B con al menos 2 cartas extraer el tope y la base y generar una pila con ellas (las demás cartas deben quedar en el mismo orden en la pila B).

Una posible solución al problema se presenta a continuación.

```
DEFINICION DE PROGRAMA
TOME UNA CARTA DE B
DEPOSITE LA CARTA EN RESPUESTA
MIENTRAS LA PILA B NO ESTA VACIA
    TOME UNA CARTA DE B
    DEPOSITE EN AUXILIAR
REPITA
TOMA UNA CARTA DE AUXILIAR
DEPOSITE EN RESPUESTA
MIENTRAS LA PILA AUXILIAR NO ESTA VACIA
    TOME UNA CARTA DE AUXILIAR
    DEPOSITE EN B
REPITA

UCP EJECUTE CON LAS SIGUIENTES CARTAS
PILA B TIENE 1 DE ESPADA ↑ - 4 DE ESPADA ↑ -
                4 DE ORO ↑ - 4 DE BASTOS ↑ -
                4 DE COPA ↑ - 1 DE BASTOS ↑
PILA AUXILIAR NO TIENE CARTAS
PILA RESPUESTA NO TIENE CARTAS
```

El programa anterior es no trivial, y si bien es posible hacer un modelo mental del proceso de movimiento de las cartas en una pila, o usar una pila de cartas físicas, ambas alternativas se vuelven engorrosas, sin tener en cuenta además lo propenso a errores que es el ser humano en lo que refiere a realizar computaciones tan repetitivas y monótonas. Por esto, y teniendo en cuenta la masividad de las computadoras hoy día, es que Retruco sirve como una herramienta en el proceso de creación y ejecución de programas de TIMBA.

4.2 Retruco

El Intérprete que se ha desarrollado para TIMBA se lo ha llamado Retruco y ha sido implementado usando una máquina virtual (apodada TIMBA2000), una puesta en práctica similar a los intérpretes BASIC de los años '80, el cual cuenta con una serie de opcodes de largo variable, que siguen una relación uno a uno con las principales construcciones del lenguaje. Por ejemplo, la instrucción de Tomar una carta de una pila se define como el opcode OS donde S $\in \mathcal{N} \cup \{0\}$.

Como se vuelve engorroso escribir en opcodes[3] de la máquina virtual se desarrolló un prototipo de editor que sirve también como generador de opcodes. Como se muestra en la figura 2, el lenguaje es casi idéntico pero con pequeñas modificaciones al lenguaje original; las cartas que componen una pila se escriben por separado o pueden intercalarse, la división de sentencias no es mediante "," pero con saltos de línea, etc. Y debido a que se pensó al lenguaje como introductorio a la programación, y siguiendo el diseño de lenguajes como Scratch, se limitó la entrada del usuario solo mediante botones, y otras restricciones.

Además del editor anterior también se está desarrollando un compilador de TIMBA a la máquina virtual.

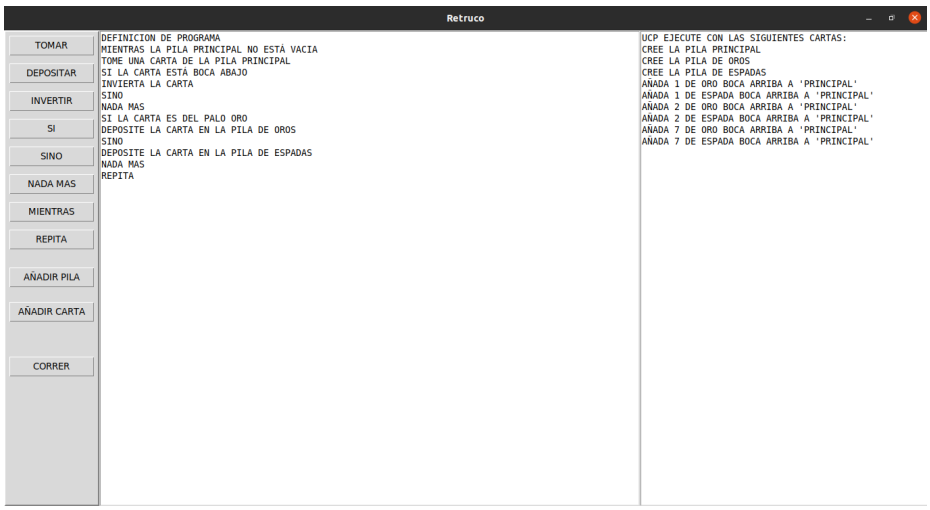

Fig. 2. El editor Retruco de TIMBA

El intérprete es Software de código abierto y se encuentra todavía en desarrollo; puede encontrarse en [10], junto con una explicación más detallada de los opcodes del lenguaje y el funcionamiento de la máquina virtual.

[3] Abreviatura en inglés para referirse a códigos de operación de la CPU

5 Conclusiones y trabajo futuro

La investigación histórica que se ha realizado nos muestra la capacidad en materia de investigación de la Computación y la producción de lenguajes que Argentina demostró a principios de los '60 con la llegada de *Clementina* que pudo habernos marcado como referentes en computación pero que por el traspié de la dictadura y las medidas para destruir el aparato científico nos llevó a perder esa oportunidad. TIMBA surge como un hito entre tantos otros, como la producción local de computadoras, de resurgimiento de la computación en nuestro país después de los oscuros sucesos de la dictadura, y es importante preservar y que sea accesible a todos y a todas no solo como una herramienta didáctica sino también como una muestra de nuestra cultura.

Debido a la alta modularidad que presenta la máquina virtual respecto a su frontend queda mucho trabajo por delante: el desarrollo de un compilador de TIMBA al código máquina de la máquina virtual, el uso de un game engine como frontend de la máquina virtual que sirva para gamificar y potenciar el uso didáctico del lenguaje, por solo nombrar algunos.

Finalmente, cabe remarcar que la existencia de un diskette perdido de TIMBA para PECOS incentiva su búsqueda y el posterior análisis de la implementación del intérprete de TIMBA que se hizo para PECOS.

Aunque en este trabajo no se han aplicado técnicas tendientes a preservar un intérprete antiguo de TIMBA propias de arqueología computacional [21], sí se ha podido preservar el lenguaje en sí mismo, haciéndolo disponible al público, así como también documentación, Testimonios y recopilación de diversos objetos materiales del legado cultural.

References

1. Historia de la informática en Latinoamérica y el Caribe: investigaciones y testimonios. Carvalho, Marcelo. Continuidad formal y ruptura real: la segunda vida de Clementina ISBN: 978-950-665-573-0
2. Mauricio R. Dávila.: Comparativa de Abordajes de Cursos Introductorios de Programación. Último acceso 8 de junio de 2021. https://core.ac.uk/download/pdf/234157119.pdf
3. Annette Vee.: Coding Literacy How Computer Programming Is Changing Writing. The MIT Press, (2017)
4. Computación Unidades 0 y 1. EUDEBA Editorial Universitaria de Buenos Aires (1985). Disponibilizado por Santiago Ceria. Último acceso 2 de Julio de 2021 https://archive.org/details/computacion-cbcunidad-0-1
5. Lenguaje TIMBA. Universidad Nacional de San Luis. Último acceso 8 de junio 2021 http://dirinfo.unsl.edu.ar/servicios/abm/assets/uploads/materiales/23005-timba.pdf
6. Ricardo Medel, Comunicación Personal, junio 2021.
7. Viviana Rubinstein, Comunicación Personal, junio 2021.
8. Programación I. Universidad Nacional de San Luis. http://proguno.unsl.edu.ar/material.html Página vigente al 09/06/2021

9. AUTOCODE un sistema de codificación para la computadora Mercury. Instituo de Cálculo Facultad de ciencias exactas y naturales. Universidad de Buenos aires. Último acceso 20 de Junio de 2021. https://elgranerocomun.net/IMG/pdf/El_Lenguaje_AUTOCODE.pdf
10. Página de GitHub del proyecto Retruco. https://github.com/qequ/retruco
11. Bell, Tim and Vahrenhold, Jan:. CS Unplugged—How Is It Used, and Does It Work?. Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday. Springer International Publishing. 2018
12. First Basic Instruction Manual. Darthmouth College Computation Center. Mayo 1964. Último acceso 1 de Julio 2021. https://www.dartmouth.edu/basicfifty/basicmanual_1964.pdf
13. Edgar Dijkstra: Go To Statement Considered Harmful. Último acceso 1 de Julio 2021 https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf
14. Drexel University. Computing and informatics. Cardiac. Último acceso 2 de Julio 2021. https://www.cs.drexel.edu/~bls96/museum/cardiac.html
15. The Little Man Computer. Último acceso 2 de Julio de 2021. https://www.kean.edu/~gchang/tech2920/http___professor.wiley.com_CGI-BIN_JSMPROXY_DOCUMENTDIRECTORDEV+DOCUMENTID&0471715425+DOCUMENTSUBID&1+PRFVALNAME&pdfs_ch06.pdf
16. Yurcik, William and Osborne, H. Febrero 2001. A crowd of Little Man Computers: visual computer simulator teaching tools - ISBN: 0-7803-7307-3. - doi: 10.1109/WSC.2001.977496.
17. CARDIAC Simulator. Último acceso 2 de Julio de 2021. https://www.cs.drexel.edu/~bls96/museum/cardsim.html
18. Catálogo Biblioteca ORT Sede Belgrano. Último acceso 3 de Julio de 2021. https://campus.belgrano.ort.edu.ar/biblioteca
19. Wilfred Oscar Durán Salvador. COMIC EL LENGUAJE DE PROGRAMACIÓN Y COMPILADOR DEL INSTITUTO DE CÁLCULO EN 1965. Ediciones del Domo.
    Último acceso 13 de Julio de 2021. http://edicionesdeldomo.altervista.org/
20. David Hagelbarger, Saul Fingerman. An instruction manual for CARDIAC. Bell Telephone Laboratories.
    Último acceso 13 de Julio de 2021.
    https://www.cs.drexel.edu/~bls96/museum/CARDIAC_manual.pdf
21. Creación de un ecosistema donde preservar el primer lenguaje y compilador argentino: Un caso de arqueología computacional. Gustavo del Dago. Proyecto SAMCA