Extensiones de Thoth para la simulación de autómatas de pila y máquinas de Turing

César García-Osorio, Javier Jimeno-Visitación, Iñigo Mediavilla-Sáiz
Dpto. de Ingeniería Civil
Universidad de Burgos
Avda. Cantabría s/n, 09006, Burgos
cgosorio@ubu.es, javierjimenov@gmail.com, imediava@gmail.com

Resumen

En este recurso se presenta la nueva funcionalidad añadida a Thoth [2], una herramienta para el aprendizaje de la teoría de autómatas y lenguajes formales, escrita en Java, y por tanto multiplataforma. La versión previa de Thoth estaba principalmente centrada en los lenguajes regulares y los autómatas finitos, aunque incluía también alguna funcionalidad relacionada con los lenguajes independientes del contexto y la generación de tablas de análisis descendente.

En la nueva versión presentada en este recurso se han añadido funcionalidades consistentes en la creación del diagrama de transición y la simulación de autómatas de pila y máquinas de Turing. El estudiante puede diseñar estos modelos de autómata y validar interactivamente su funcionamiento. Una vez que el autómata ha sido creado es posible ver paso a paso el proceso de reconocimiento del lenguaje. Existen distintas variaciones al modelo básico de máquina de Turing, en esta versión de Thoth se han implementado los siguientes tipos de máquinas de Turing: binarias, multi cinta, multi cabeza, multi pista, cinta limitada, y máquinas escribir/mover. Además de estas nuevas funcionalidades, la interfaz de usuario presenta novedades importantes. La más notable es la posibilidad de hacer y deshacer operaciones de diseño del autómata (incluyendo los autómatas finitos incluidos en la versión previa de la herramienta). También, el usuario puede ampliar o reducir los diagramas de transición que representan los autómatas.

Frente a otras herramientas existentes Thoth presenta como ventajas la cuidada interfaz gráfica, la integración de distintos modelos de autómata en una única utilidad y la internacionalización, dado que permite elegir entre cuatro idiomas: español, inglés, francés y alemán.

En el futuro esperamos mejorar este recurso incluyendo otros tipos de autómatas y añadiendo la posibilidad de reutilizar máquinas de Turing como módulos de otras máquinas de Turing.

1. Introducción

Sin duda una de las mejores formas de hacer la adquisición de conocimientos atractiva al alumno es mediante el uso de herramientas interactivas. Esto es especialmente cierto cuando los conocimientos a transmitir son los de teoría de autómatas y lenguajes formales, conocimientos habitualmente considerados por los alumnos como demasiados complejos y teóricos.

La herramienta que comentamos en este artículo permite al alumno la simulación de los dispositivos de la teoría de autómatas y lenguajes formales. La versión previa de la herramienta estaba centrada fundamentalmente en aquellos formalismos y algoritmos relacionados con los lenguajes de tipo 3 (lenguajes regulares), el tipo de lenguaje más simple de la jerarquía de Chomsky. La nueva funcionalidad presentada en este artículo permite la simulación de los autómatas de pila y las máquinas de Turing, dispositivos relacionados con el reconocimiento de los lenguajes de tipo 2 (lenguajes independientes del contexto) y lenguajes de tipo 0 (lenguajes gramaticales) de dicha jerarquía.

En la sección que sigue vamos a dar un rápido repaso a los principales conceptos de la teoría de lenguajes y autómatas, centrándonos especialmente en los conceptos de autómata de pila y máquina de Turing. A continuación, daremos un resumen de la funcionalidad previa. Seguimos con la descripción de la nueva funcionalidad. Terminamos con la comparación de Thoth con otras herramientas y con las conclusiones y nuestra previsión de trabajo futuro.

2. Conceptos teóricos

La versión actual de THOTH facilita el aprendizaje de cinco conceptos fundamentales: gramáticas, expresiones regulares, autómatas finitos y máquinas de Turing. De éstos, la funcionalidad relativa a los dos últimos constituye la novedad en esta su última versión. A continuación vamos a dar una breve descripción de cada uno de estos conceptos. Para un tratamiento en profundidad pueden consultarse los libros de Alfonseca [1] o Kelley [3].

2.1. Gramáticas

Como las gramáticas para los lenguajes naturales, las gramáticas formales proporcionan las reglas que permite construir las sentencias que constituyen el lenguaje. Desde el punto de vista formal, un lenguaje no es más que un conjunto de *cadenas* (*sentencias*) construidas utilizando una secuencia de *símbolos* o *caracteres* (*palabras*) de un *alfabeto* (*vocabulario*).

Las reglas de una gramática formal se llaman *producciones*; son reglas de reescritura que indican como una cadena puede ser transformada en otra. El lenguaje generado por la gramática esta formado por todas las cadenas que pueden derivarse del *axioma de la gramática* (un símbolo especial que inicia el proceso generativo) aplicando una secuencia de transformaciones definidas por las producciones de la gramática.

2.2. Expresiones regulares

Las expresiones regulares son otra forma de describir lenguajes. En concreto, son una sintaxis para describir un tipo especial de lenguajes denominados *lenguajes regulares*.

Los lenguajes regulares (o lenguajes de tipo 3) son los lenguajes más sencillos de la jerarquía de Chomsky para lenguajes. Estos lenguajes pueden ser descritos también por gramáticas, pero las expresiones regulares son una notación más compacta y por tanto preferida. Hay tres operadores que pueden utilizarse con las expresiones regulares: el carácter '|' indica *selección*, el carácter '*' indica *repetición* o *cierre* y el carácter '·' indica *concatenación* (con frecuencia este operador no se hace explícito, las expresiones regulares concatenadas simplemente se ponen una a continuación de la otra).

Por ejemplo, la expresión regular (1|3)·(0·0)*·0 representa el lenguaje formado por todas las cadenas que comienzan por '1' o '3' y que van seguidas de un número impar de ceros (los dos ceros se pueden repetir un número arbitrario de veces, o no aparecer en absoluto, y van seguidos de un cero adicional).

2.3. Autómatas finitos

Son un dispositivo con un número finito de *estados* y de *transiciones* entre ellos. Las transiciones están etiquetadas con una carácter e indican como evolucionar al siguiente estado a partir del actual y dependiendo del siguiente carácter de la entrada. Para seguir una transición el siguiente carácter de entrada tiene que coincidir con el carácter que etiqueta la transición.

Los autómatas finitos pueden representarse mediante un grafo con nodos representando los estados y arcos representando las transiciones. Si siguiendo los arcos etiquetados con los símbolos de una palabra, podemos empezar en el *estado inicial* y finalizar en un *estado final*, la palabra se dice es reconocida por el autómata. Al conjunto de palabras para las que puede hacerse lo anterior, se le denomina lenguaje reconocido por el autómata finito.

La clase de lenguajes reconocidas por los autómatas finitos es exactamente la misma que la clase de lenguajes que se pueden describir mediante expresiones regulares.

2.4. Autómatas de pila

Los autómatas de pila, como su nombre índica, tienen una pila en la que pueden acumular símbolos, difieren de los autómatas finitos en que: i) pueden usar el tope de la pila para decidir que transición seguir, ii) pueden manipular la pila cada vez que siguen una transición. Ahora, con el uso de la pila, los autómatas de pila pueden reconocer lenguajes independientes del contexto; una clase de lenguajes más amplia que la de los lenguajes regulares. Como los autómatas finitos, los autómatas de pila pueden ser representados mediante un grafo. Los arcos ahora están etiquetados con: el carácter que tienen que leer de la entrada, el carácter que tiene que estar presente en el tope de pila, y los símbolos que se van a acumular en la pila si se sigue la transición.

Formalmente, un autómata de pila se define como una 7-tupla (Σ, Γ, $\mathcal{Q}$, Z, q_0, f, $\mathcal{F}$), donde:

- Σ es un *alfabeto de entrada*.
- Γ es un *alfabeto de pila*.
- $\mathcal{Q}$ es un alfabeto de estados.
- $Z \in \Gamma$ es el *símbolo inicial* de pila.
- $q_0 \in \mathcal{Q}$ es el *estado inicial*.
- f es la *función de transición*:
$$f: \mathcal{Q} \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}(\mathcal{Q} \times \Gamma^*)$$
- $\mathcal{F} \subset \mathcal{Q}$ es el conjunto de *estados finales* o estados de aceptación.

En la Figura 1 puede verse el diagrama de transición de autómata de pila capaz de reconocer el lenguaje $a^n b^n$ (las cadenas formadas por una secuencia de n aes seguidas de una secuencia de n bes).

La clase de lenguajes reconocidas por los autómatas de pila es exactamente la misma que la clase de lenguajes que se pueden describir con las *gramáticas independientes del contexto*. Esta clase de lenguajes se corresponde con los lenguajes de tipo 2 de la jerarquía de Chomsky.

2.5. Máquinas de Turing

Las máquinas de Turing son los autómatas más potentes de los tres que hemos mencionado. En realidad no están limitadas al reconocimiento de lenguajes y pueden resolver cualquier problema cuya solución pueda representarse de forma algorítmica.

Podemos pensar en un máquina de Turing como un dispositivo con una cabeza lectora/escritora que se mueve sobre una cinta infinita de dónde leer y dónde hace anotaciones. La cabeza, después de leer un símbolos y sustituirlo por otro (o dejarlo como esta), puede moverse a la izquierda o a la derecha.

Formalmente, una máquina de Turing se puede definir, como en el caso de los autómatas de pila, como una 7-tupla (Σ, Γ, b, $\mathcal{Q}$, q_0, $\mathcal{F}$, f), donde:

- Γ es el alfabeto de elementos llamados: *símbolos* o *caracteres de pila*.
- $\Sigma \subseteq \Gamma - \{b\}$ es un alfabeto de *símbolos de entrada*.
- b es un símbolo especial de Γ ($b \in \Gamma$ y $b \notin \Sigma$).
- $\mathcal{Q}$ es un alfabeto de *estados*.
- $q_0 \in \mathcal{Q}$ es el *estado inicial*.
- f es la *función de transición*:
$$f: \mathcal{Q} \times \Gamma \rightarrow \mathcal{Q} \times \Gamma \times \{\mathcal{L}, \mathcal{R}, S\}$$
donde: $\mathcal{L}$ representa que la cabeza de lectura escritura se moverá la izquierda ($\mathcal{L}$eft), $\mathcal{R}$ representa que la cabeza se moverá a la derecha ($\mathcal{R}$ight) y S que la máquina detendrá su funcionamiento (Stop).
- $\mathcal{F} \subset \mathcal{Q}$ es el conjunto de *estados finales* o *estados de aceptación.*

En la Figura 2 puede verse el diagrama de transición de una máquina de Turing capaz de reconocer el lenguaje Ia^pD donde $p=2^n$, $n>0$ (es decir, las secuencias de una potencia de dos de aes, que comienzan con un delimitador izquierdo I, y finalizan con un delimitador derecho D).

La clase de lenguajes reconocidas por las máquinas de Turing es exactamente la misma que la clase de lenguajes que se pueden describir mediante *gramáticas* (cualquier tipo de gramática). Esta clase de lenguajes se corresponde con los lenguajes de tipo 0 de la jerarquía de Chomsky.

Si a las máquinas de Turing, en el proceso de reconocer lenguajes, se les impone la restricción de utilizar una cinta de tamaño finito, la clase de lenguajes que reconocen coincide con la clase de lenguajes que se pueden describir con las gramáticas dependientes del contexto. Esta clase de lenguajes se corresponde con los lenguajes de tipo 1 de la jerarquía de Chomsky.

3. Resumen de la funcionalidad previa

En la versión previa Thoth implementaba distintos algoritmos que tenían que ver fundamentalmente con los lenguajes regulares, aunque también con gramáticas independientes del contexto. A continuación se enumera la funcionalidad aportada:

- *Gramáticas*: eliminación de producciones no generativas y de símbolos no terminables, no alcanzables y anulables; eliminación de recursividad a izquierdas; factorización por la izquierda; obtención de la forma normal de Chomsky; algoritmos de análisis LL(1) y de Cocke-Younger-Kasami; si la gramática es regular, obtención del autómata finito.
- *Expresiones regulares*: obtención del autómata finito a partir de la expresión regular (usando tres algoritmos distintos);

determinación de si una palabra pertenece al lenguaje descrito por la expresión regular; determinación de la equivalencia entre dos expresiones regulares.

- *Autómatas finitos*: obtención del autómata a partir de la expresión regular; dibujo y simulación de autómatas finitos; obtención de la expresión o gramática regular que define el lenguaje reconocido por el autómata; comprobación de la equivalencia entre dos autómatas; eliminación del determinismo; minimización de autómatas.

4. La nueva funcionalidad de Thoth

En esta versión de Thoth se han corregido pequeños bugs de la versión previa y ha habido un rediseño completo de la interfaz gráfica. Un cambio muy importante ha sido la utilización de la potente librería de dibujo de grafos JGraph[1], que permite mejorar la experiencia del usuario permitiendo la incorporación de mejoras como el zoom y las operaciones de deshacer y rehacer. Estas mejoras afectan también a la edición de los autómatas finitos.

Otra novedad de esta versión es la posibilidad de genera un documento PDF en la que se muestre paso a paso la secuencia de configuraciones por las que pasa un autómata de pila o una máquina de Turing en el proceso de reconocer un lenguaje (o en el caso de la máquina de Turing en el proceso de efectuar cualquier otro tipo de computación). Este documento puede ser impreso por el alumno para ser utilizado como referencia en las tutorías o para incorporarlo a las notas tomadas en clase.

Los añadidos, propios de los nuevos autómatas considerados, se explican a continuación.

4.1. Simulación de autómatas de pila

El primer paso, cuando trabajamos con los autómatas de pila, es definir su funcionamiento mediante la definición de su diagrama de transiciones, que podemos editar con las herramientas de dibujo. Podemos insertar estados y conectarlos mediante transiciones. Cada vez que realizamos la conexión de un par de estados se nos pregunta por el símbolo que deberá leerse de la entrada, el símbolo que deberá estar presente en el tope de pila, y la secuencia de símbolos que debe apilarse a continuación (estos se apilarán de izquierda a derecha, es decir al final, el último símbolo de la secuencia, será el símbolo de tope de pila).

Una vez definido el diagrama de transiciones para el autómata de pila, tenemos distintas opciones disponibles (ver Figura 3). Podemos simular su funcionamiento, observando paso a paso, como se van modificando el estado actual, la entrada por leer y los contenidos de la pila. Cuando hacemos esto, podemos avanzar o retroceder en el proceso. O seleccionar las configuraciones a las que llegamos para mostrar una traza que nos muestra toda la secuencia de configuraciones que nos ha permitido llegar a la seleccionada. Esto es especialmente útil en presencia de situaciones de indeterminismo, en las que podemos tener más de una posible configuración actual.

En vez de una ejecución paso a paso, podemos omitir la visualización de las configuraciones intermedias, haciendo que se muestre directamente la configuración final alcanzada. Seleccionando esta, como se explico anteriormente, podemos ver toda la secuencia de configuraciones que se han seguido para alcanzar esa configuración final.

También, podemos comprobar de golpe la pertenencia o no de un conjunto de palabras al lenguaje reconocido por el autómata de pila (ver Figura 4).

Por último, podemos detectar los estados que dan lugar a situaciones de indeterminismo. Con esta opción los estados en los que se da esta situación se sobre-iluminan en amarillo.

4.2. Simulación de máquinas de Turing

Como en el caso de los autómatas de pila, el primer paso para trabajar con máquinas de Turing, será la definición del diagrama de transiciones. Las opciones de dibujo son idénticas. Y lo mismo las operaciones que podemos llevar a cabo una vez tenemos el diagrama de transiciones definidos. Una diferencia importante es, sin embargo, la posibilidad de elegir distintos tipo de máquinas de Turing. El modelo básico se complementa con otros, fruto de restringir o aumentar las capacidades del modelo básico.

Las variantes de máquinas de Turing implementadas son:

[1] http://www.jgraph.com/jgraph.html.

- Máquina de Turing convencional. Este es el modelo básico. Las transiciones están constituidas por tres elementos: el símbolo que se lee de la cinta, el símbolo que se escribe en la misma, y la operación a realizar (mover la cabeza a la izquierda, a la derecha, o detener la ejecución).
- Maquina de Turing binaria. Es una restricción de la anterior en la que el alfabeto de entrada sólo esta constituido por dos símbolos: 0 y 1.
- Máquina de Turing con cinta limitada. La cinta de entrada sólo es infinita en una dirección.
- Máquina de Turing "escribir o mover". Aunque se trata de una restricción de la que hemos denominado "modelo convencional", este fue el modelo de máquina de Turing originalmente propuesto por Alan Turing [4]. En este modelo no es posible escribir un nuevo símbolo y mover la cabeza al mismo tiempo. La transición o bien escribe un símbolo, o bien mueve la cabeza.
- Máquina de Turing multipista. Estas es una ampliación al modelo básico en la que la cinta esta dividida en *n* pistas, cada una de ellas puede contener símbolos. La decisión de que hacer a continuación se basa en el conjunto de los *n* caracteres leídos.
- Máquina de Turing multicinta. En esta otra ampliación, en vez de una única cinta, tenemos *n* cintas, cada una con su correspondiente cabeza de lectura/escritura. Por tanto un paso básico involucra leer *n* símbolos, escribir *n* símbolos y llevar a cabo *n* operaciones de movimiento.
- Máquina de Turing multicabeza. En este modelo, la cinta es única, pero *n* cabezas se mueven sobre ella. Las transiciones son idénticas a las de la máquina de Turing multicinta, la diferencia sólo se hará aparente cuando estemos simulando su funcionamiento.

En la Figura 5 pueden verse los tipos de transiciones para estas variantes de máquinas de Turing.

5. Otras herramientas

Existen otras herramientas que como Thoth proporcionan la posibilidad de simular autómatas de pila y máquinas de Turing. La mayoría de ellas sólo ofrecen una u otra funcionalidad. Thoth frente a ellas tiene la ventaja de integrar en una única aplicación estos dos tipos de autómatas junto con los finitos y la restante funcionalidad relativa a lenguajes regulares y a la generación de tablas de análisis sintáctico predictivo descendente.

En Internet podemos encontrar multitud de pequeños applets que implementan el modelo básico de máquina de Turing. Como curiosidad se puede encontrar incluso un vocabulario XML para definir máquinas de Turing y una hoja de estilo XSLT que es capaz de simular su funcionamiento[2]. Asimismo, son numerosos los applets para la simulación de autómatas de pila. La mayoría de los cuales adolecen del mismo problema, la interfaz gráfica no es cómoda de usar. Aquí sólo vamos a enumerar aquellos simuladores que por sus características nos ha parecido más destacables:

- Visual Automata Simulator[3]: Escrita en Java, y por tanto multiplataforma. Proporciona simulación tanto de autómatas finitos como de máquinas de Turing. El diseño de los diagramas es pesado ya que obliga constantemente a ir dando nombres a los estados. El modelo de máquina que simula se puede calificar de escribir/mover, pero en su representación gráfica las acciones de escribir y mover están asociadas a los nodos del diagrama. Su característica más destacada es la disponibilidad de acciones del tipo "mover hasta encontrar *a*" o "mover hasta que sea distinto de *a*". Además, permite la invocación a sub-máquinas, lo que facilita el diseño modular de máquinas de Turing.
- Visual Turing[4]: No se trata de una aplicación multiplataforma, sólo esta disponible para windows. Su uso es muy pesado. Tenemos que estar constantemente seleccionando la operación que queremos realizar, lo que nos obliga a movernos constantemente del área de dibujo al área de menús. Sólo proporciona un tipo básico de máquina de Turing.
- Visual Turing Machine[5]: Su interfaz gráfica es bastante rígida, obliga a definir primero el vocabulario. Sólo permite el diseño y

[2] http://www.unidex.com/turing/utm.htm.
[3] http://www.cs.usfca.edu/~jbovet/vas.html.
[4] http://cheransoft.com/vturing/.
[5] http://sourceforge.net/projects/visualturing/.

simulación de máquinas de Turing escribir/mover.

- Tuatara Turing Machine Simulator[6]: Escrita en Java. Tiene una interfaz gráfica bastante cuidada. Es la única de las herramientas analizadas que incorpora funciones de deshacer/rehacer, como hace Thoth. La modificación de transiciones es un poco engorrosa al forzarnos a utilizar operaciones independientes para símbolo leído y acción. Su uso no es demasiada intuitivo y no es fácil de empezar a usar sin haberse leído primero el manual. Sólo implementa la máquina de Turing escribir/mover.
- Automaton Simulator[7]: Permite el diseño y simulación de autómatas finitos, autómatas de pila y máquinas de Turing. Como en otras, su punto débil es la facilidad de uso.

Merece una mención especial JFLAP[8]. Esta herramienta es sin duda muy superior a las previamente citadas. Al igual que Thoth, integra la simulación de autómatas finitos, autómatas de pila y tanto máquinas de Turing convencionales como multicinta. Frente a JFLAP, Thoth ofrece una forma más flexible de interaccionar con los elementos de dibujo. Además, Thoth proporciona máquinas de Turing que no están presentes en JFLAP, como son las máquinas de Turing multipista, multicabeza, escribir/mover, binaria y cinta limitada.

En su última versión, de enero de este mismo año, incorpora funcionalidad, que antes estaba presente sólo en Thoth, por ejemplo el algoritmo de Cocke-Younger-Kasami y la posibilidad de distribuir los estados del autómata de acuerdo a un layout. Pero hay que reconocer que sus algoritmos de layout son mucho más sofisticados que el presente en Thoth. Ésta es sin duda una de las cuestiones que tendrá que ser tenida en cuenta en versiones posteriores de Thoth.

6. Conclusiones y trabajo futuro

Hemos presentado una herramienta que incorpora en una única aplicación distintos algoritmos de la teoría de autómatas y lenguajes formales. Permite trabajar con gramáticas independientes del contexto, expresiones regulares, autómatas finitos, y ahora, en la nueva versión, con autómatas de pila y máquinas de Turing. Algunas de las variantes de máquinas de Turing no las hemos encontrado en ninguna otra herramienta.

Este año se ha empezado a utilizar Thoth en las clases de laboratorio de la asignatura de Autómatas y Lenguajes Formales de la Ingeniería en Informática de Gestión de la Universidad de Burgos. La experiencia ha sido positiva. Se explicaba el funcionamiento tanto de Thoth como de JFLAP, y se daba al alumno la libertad de elegir aquella herramienta con la que se sentía más cómodo. Al final, más de la mitad de los alumnos que acudían regularmente a las clases de laboratorio, optaron por usar Thoth.

En el futuro esperamos añadir otros tipos de autómatas, tales como las máquinas secuenciales de Moore y de Mealy. Un añadido mucho más interesante, a la hora de facilitar el diseño modular de máquinas de Turing, será la posibilidad de reutilizar máquinas de Turing como módulos de otras máquinas de Turing. También esperamos incorporar algún modo de importar y exportar los autómatas y máquinas de Turing a formatos definidos por otras herramientas, como JFLAP. Por último, creemos conveniente añadir a Thoth algún mecanismo de distribución automática de estados, para facilitar la visualización de los diagramas de transición, cuando estos tengan un alto número de estados.

La herramienta descrita en este artículo puede descargarse de la siguiente página web: http://pisuerga.inf.ubu.es/cgosorio/THOTH/.

Referencias

[1] Alfonseca, M., Alfonseca, E., Morrión, R. *Teoría de Autómatas y Lenguajes Formales*, McGraw Hill, 2007.

[2] García Osorio, C., Arnáiz Moreno, A., Arnáiz González, A. Enseñanza asistida de teoría de autómatas y lenguajes formales mediante el uso de THOTH. *XIII JENUI*, 425-432, Teruel, 2007.

[3] Kelley, D. *Automata and Formal Languages: An Introduction*. Prentice Hall, 1998.

[4] Turing, A. Intelligent Machinery, en *Collected Works of A.M. Turing: Mechanical Intelligence*, Ince, D.C. (editor), Elsevier Science Publishers, 1992.

[6] http://sourceforge.net/projects/tuataratmsim/.
[7] http://ozark.hendrix.edu/~burch/proj/autosim/.
[8] http://www.jflap.org/.

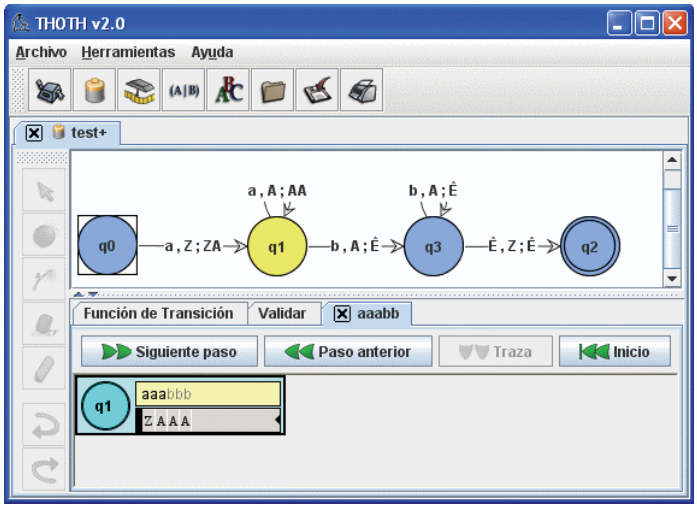


Figura 1. Thoth mostrando un autómata de pila y la ejecución paso a paso.

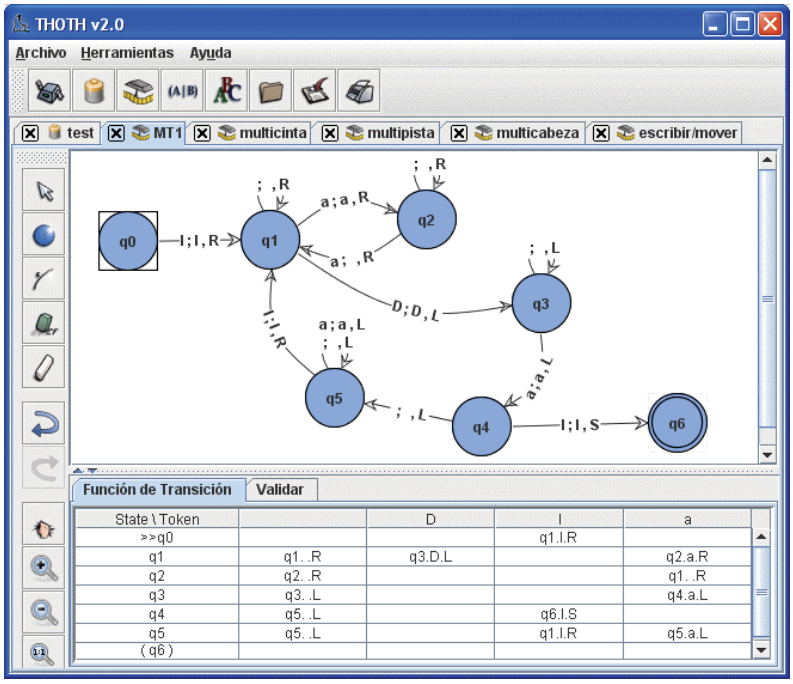


Figura 2. Thoth mostrando el diagrama y función de transición para una máquina de Turing.

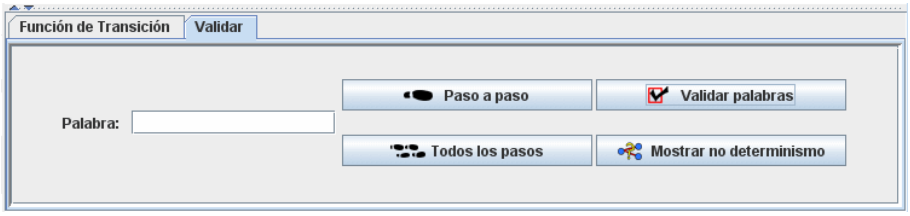


Figura 3. Las opciones disponibles una vez que el autómata/máquina ha sido creado.

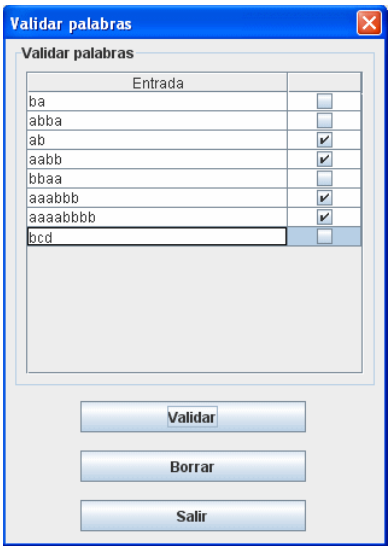
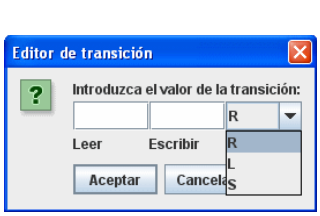
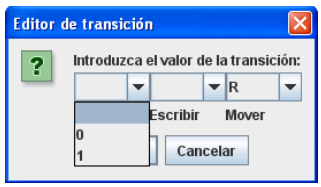


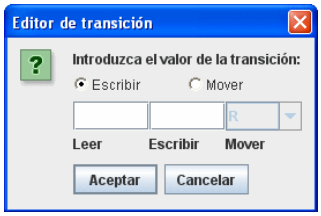
Figura 4. Comprobación simultánea de qué palabras pertenecen al lenguaje reconocido por el autómata de pila la Figura 1 (cadenas de la forma $a^n b^n$).



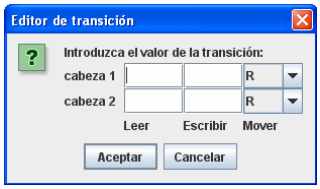
(a) Convencional



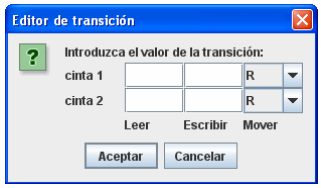
(b) Binaria



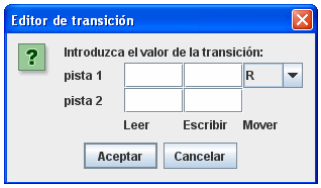
(c) Escribir/mover



(d) Multicabeza



(e) Multicinta



(f) Multipista