SOTA, una herramienta educativa para la enseñanza de la tabla de símbolos

Gallego Carrillo, Micael, Gortázar Bellas, Francisco, Urquiza Fuentes, Jaime y Velázquez Iturbide, J. Ángel
Universidad Rey Juan Carlos, Grupo de investigación VIDO, C/ Tulipán s/n, 28933 MADRID
mgallego@escet.urjc.es, {francisco.gortazar,jaime.urquiza,angel.velazquez}@urjc.es

Resumen

Se presenta una herramienta educativa para el aprendizaje de la tabla de símbolos, dentro del marco de una asignatura de enseñanza de compiladores. La herramienta muestra de forma visual el funcionamiento de una tabla de símbolos durante el proceso de análisis de un código fuente. Planteamos también una propuesta de evaluación de la herramienta que se llevará a cabo en el segundo semestre del presente curso académico dentro de la asignatura de Procesadores de Lenguajes de la titulación de Ingeniería Informática en la Universidad Rey Juan Carlos.

1. Introducción

La visualización es un recurso ampliamente utilizado en la enseñanza de la informática. Una de las primeras experiencias en este ámbito fue Sorting Out Sorting [2], una animación de 30 minutos que visualiza el funcionamiento de diferentes algoritmos de ordenación. Existen múltiples estudios sobre la utilización de la visualización en la enseñanza de la informática [1, 5, 6, 11, 12, 16]. Actualmente existen herramientas que visualizan tanto la propia ejecución de programas [8, 9, 10, 14, 15] como estructuras de datos y algoritmos [4].

En el ámbito de la construcción de compiladores, y en general en cualquiera de los tipos de procesadores de lenguajes, también se han desarrollado numerosas herramientas que visualizan la ejecución de procesos relevantes en este área. Entre ellas se pueden mencionar algunas aproximaciones iniciales, como [7] y [13], y herramientas que aún cuentan con un activo desarrollo, como JFlap[3] y Jaccie[17].

La enseñanza de procesadores de lenguajes se suele dividir en dos partes claramente diferenciadas. Una parte de análisis léxico y sintáctico del código fuente que se realiza mediante técnicas basadas en la teoría de lenguajes formales. Y otra parte que engloba procesos adicionales como por ejemplo el uso de tablas de símbolos, técnicas de unificación de expresiones de tipos, técnicas de generación de código o técnicas de optimización de código. Las herramientas mencionadas anteriormente se centran en la parte de análisis léxico y sintáctico que está basada en la teoría de lenguajes formales, como por ejemplo algoritmos de transformación de autómatas no deterministas en deterministas, reconocimiento de cadenas de entrada con diferentes mecanismos dependiendo del tipo del lenguaje o simuladores de máquinas de Turing.

En la literatura relacionada no se mencionan herramientas con fines educativos que visualicen otras fases del proceso de compilación, como la utilización de la tabla de símbolos.

En este contexto se presenta SOTA, una herramienta educativa sencilla que se centra en la visualización de alto nivel del funcionamiento de una tabla de símbolos en el proceso de análisis del código fuente.

En la siguiente sección se describen los conceptos teóricos de la tabla de símbolos. La sección 3 muestra cómo SOTA visualiza, tanto estática como dinámicamente, dichos conceptos. La sección 4 expone una discusión sobre la utilidad pedagógica y se describe cómo se evaluará la herramienta. Finalmente, en la sección 5, se describen nuestras conclusiones y líneas de trabajo futuro.

2. Enseñanza de la Tabla de Símbolos

En esta sección se muestra un resumen de los conceptos teóricos de la tabla de símbolos. SOTA se ha diseñado para apoyar la explicación y el aprendizaje de estos.

En el proceso de compilación de un programa, la comprobación de tipos se encarga de asegurar que los elementos del lenguaje se utilizan correctamente (su correcta escritura y ubicación ya ha sido comprobada por los analizadores léxico y sintáctico respectivamente). Los elementos de los lenguajes de programación para los que se necesita hacer este tipo de comprobación son los identificadores.

Las principales características de los identificadores son: su nombre, y su ámbito de visibilidad y tipo, que indican respectivamente en qué lugar del programa se pueden usar y qué operadores se puede aplicar sobre ellos. El nombre de un identificador se encuentra en el propio token. Sin embargo, el ámbito de visibilidad y el tipo son datos que se encuentran asociados al identificador en una sentencia de declaración, que está compuesta por varios tokens agrupados en una o varias producciones de la gramática. Por lo que se necesita una estructura de datos donde almacenar toda esa información relativa a los identificadores.

La tabla de símbolos es esa estructura de datos, y la información a guardar es muy variada. Un identificador puede desempeñar diferentes papeles en un código fuente, éstos dependen del lenguaje de programación utilizado. Por ejemplo: variables, funciones, procedimientos, clases o tipos definidos por el usuario.

La información de tipo a guardar dependerá del papel desempeñado por el identificador. Si se trata de una variable, bastará con guardar su tipo asociado. Si es un matriz, ya habrá que guardar el tipo de los elementos y las dimensiones de la matriz. Para los procedimientos será necesario guardar los parámetros formales, y para las funciones, además habrá que guardar el tipo del valor devuelto. De los tipos definidos por el usuario se necesita guardar su definición. Por último, de las clases habría que guardar la información de herencia, así como sus atributos y métodos con su correspondiente información de acceso.

El ámbito de visibilidad de un identificador depende del lenguaje de programación utilizado, pero en general se define mediante entornos. Un entorno es una parte del programa delimitada de alguna forma. Todo el programa está estructurado de forma jerárquica en entornos: el programa en sí es un entorno, formado por tantos entornos como funciones y procedimientos tenga declarados dentro del programa principal. A su vez cada uno de estos entornos estará compuesto por otros resultantes de funciones o procedimientos definidos dentro de ellos (si es que el lenguaje lo permite) o cuerpos de bucles o sentencias de bifurcación, también llamados entornos anónimos, por no tener identificador asociado, como es el caso de funciones y procedimientos. La información del ámbito de visibilidad de un identificador se puede guardar de muchas formas. Una forma podría ser asignando códigos a entornos, guardando el entorno de declaración de un identificador, las relaciones entre los distintos entornos y el entorno activo actual. Otra forma es creando la estructura de la tabla de símbolos a modo de árbol, de forma que la raíz es el entorno principal y sus nodos hijos son los entornos creados a partir de él. Esta última forma es la utilizada en nuestro caso.

Otro punto importante es la manipulación de la tabla de símbolos. Ésta consiste en la creación de entornos, inserción de entradas en dichos entornos, modificación y eliminación de dichas entradas, y localización de entradas en la estructura de la tabla de símbolos. La primera acción en la tabla de símbolos es la creación del entorno raíz. A partir de aquí se insertarán entradas, comprobando los posibles errores de duplicidad; se crearán los entornos hijos correspondientes si procede; se buscarán identificadores en la estructura de la tabla de símbolos, donde entra en juego la información del ámbito de visibilidad; y se accederá a la información de las distintas entradas encontradas.

3. Visualización de la tabla de símbolos con SOTA

Esta sección describe la visualización que hace nuestra herramienta de los conceptos de la tabla de símbolos presentados en la sección anterior.

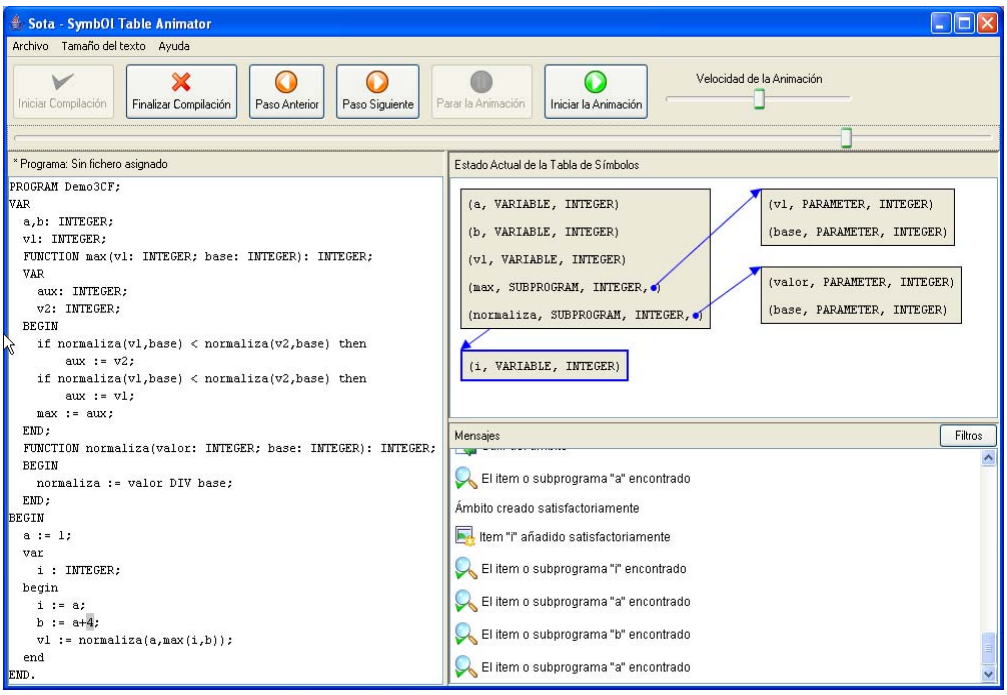

Figura 1. Interfaz de SOTA

En la Figura 1 se puede ver la interfaz de SOTA. Ésta se divide en tres zonas: zona de programa, zona de estado actual de la tabla de símbolos y zona de mensajes. La zona de programa, parte izquierda de la pantalla, es donde se muestra el programa que se está analizando. Dicho programa se deberá escribir utilizando una variante de Pascal que hemos denominado SimplePascal, cuya descripción se encuentra en la ayuda de la herramienta. La zona de estado actual de la tabla de símbolos, parte derecha superior, es donde se muestra la representación gráfica del estado actual de la tabla de símbolos, así como las últimas operaciones realizadas (inserción, creación de entorno y búsqueda con o sin éxito). La zona de mensajes, parte derecha inferior, muestra una breve descripción textual de las operaciones realizadas hasta el momento sobre la tabla de símbolos.

El/la alumno/a podrá tanto editar directamente en la herramienta sus propios programas, como utilizar un conjunto de programas de demostración que son descargados por la herramienta desde un sitio web, permitiendo al docente incorporar nuevos programas de demostración cuando lo considere necesario. En la Figura 2 se muestra el cuadro de diálogo que permite seleccionar una de estas demostraciones. Cada una de ellas viene acompañada de un nombre y una descripción del contenido.

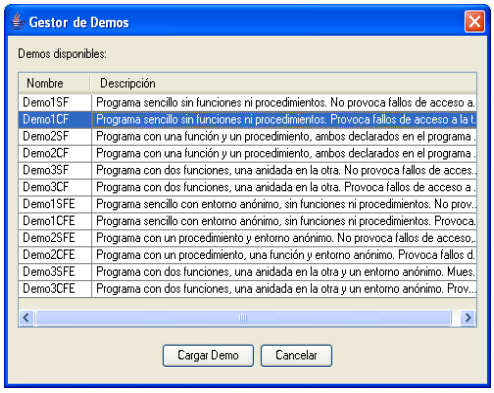

Figura 2. Demos utilizables.

A continuación se describirá la visualización de los conceptos de la tabla de símbolos, tanto de forma estática como dinámica.

3.1. Visualización estática

Para representar la estructura de la tabla de símbolos cuando está compuesta por distintos entornos, se utiliza un árbol cuyos nodos son los entornos y cuyos arcos son las relaciones entre entornos padres e hijos. El árbol crece hacia la derecha para los nuevos procedimientos y funciones y hacia abajo para los entornos anónimos. Dentro de cada entorno, las entradas se añaden en la parte inferior. Se puede ver un ejemplo de esta estructura arbórea en la Figura 3. El estado en el que se encuentra la tabla de símbolos en esta figura se corresponde con el punto en el que se procesa el token resaltado en el código fuente de la Figura 4.

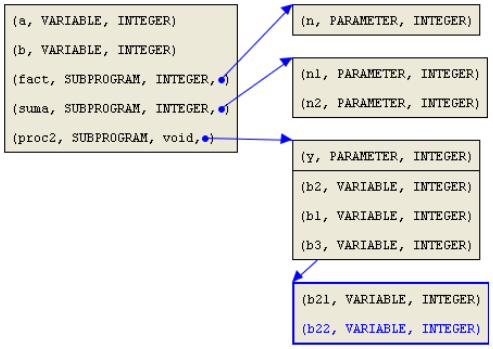

Figura 3. Estructura de árbol

Para facilitar la legibilidad de la visualización, es posible aumentar o disminuir el tamaño de la fuente. Esto permite adecuar el tamaño a una presentación para una clase teórica, o disminuirlo cuando al árbol crece más allá de los límites de la zona donde se visualiza.

Los entornos se representan gráficamente como rectángulos. La entradas se representan textualmente dentro del rectángulo asociado al entorno al que pertenecen. En la Figura 3 se pueden observar cinco entornos. El entorno raíz con cinco entradas, tres de ellas son subprogramas. Cada uno de ellos tiene su entorno asociado. En concreto, dentro del procedimiento "proc2" se distinguen, en primer lugar el parámetro "y", y a continuación las tres variables "b2", "b1" y "b3". Finalmente existe un entorno anónimo definido dentro del procedimiento "proc2" con dos variables ("b21" y "b22").

```
PROGRAM Ejemplo;
VAR
 a,b: INTEGER;
 FUNCTION fact(n : INTEGER) : INTEGER;
 BEGIN
   if (n > 1) then fact := n*fact(n-1)
   else fact := 1;
 END;
 FUNCTION suma(n1 : INTEGER; n2 : INTEGER) : INTEGER;
 BEGIN
   suma := n1 + n2;
 END;
 PROCEDURE proc2(y: INTEGER);
 VAR
   b2, b1, b3: INTEGER;
 BEGIN
   VAR
     b21, b22: INTEGER;
   BEGIN
     aux := a;
   END;
 END;
BEGIN
 b := fact(4);
END.
```

Figura 4. Código analizado

Para facilitar el seguimiento de las acciones realizadas sobre la tabla de símbolos, SOTA destaca tanto el último entorno creado como la última entrada insertada con color azul, de forma que se diferencien del resto. En la Figura 5 se destacan el último entorno creado, que es el correspondiente al cuerpo del procedimiento "proc2" y la última entrada insertada que es la variable "b3".

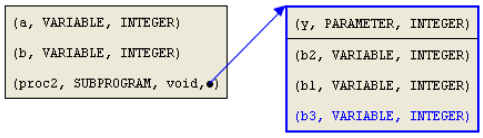

Figura 5. SOTA destaca el último entorno creado y la última entrada insertada.

También se destacan las acciones de búsqueda, así como su resultado. El entorno sobre el que se está buscando en cada momento se resalta en color rojo. Si la búsqueda no tiene éxito en ese entorno se marca con una línea diagonal también en rojo, pasándose a buscar en el entorno padre. Si la búsqueda encontró la entrada, ésta se resalta en color verde. En las siguientes tres figuras se presentan las tres posibilidades. En la Figura 6 se ve cómo se está buscando en el

entorno anónimo. En la Figura 7 se muestra que la búsqueda falló en el entorno anónimo, pero se continúa en el entorno padre. Finalmente, en la Figura 8 se visualiza que en el entorno del cuerpo de procedimiento también falló la búsqueda, pero en el entorno raíz se encontró la entrada que se buscaba.

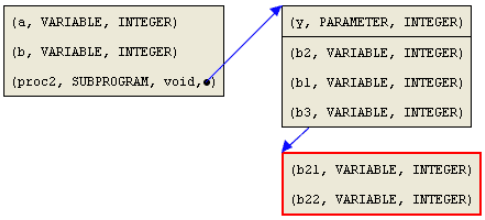

Figura 6. Búsqueda de una entrada en el entorno anónimo.

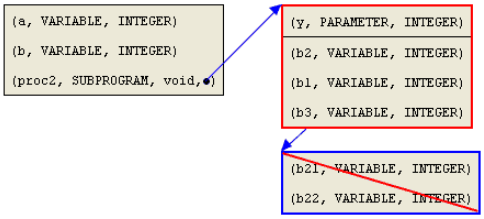

Figura 7. Búsqueda sin éxito en el entorno anónimo y traslado de la búsqueda al entorno padre.

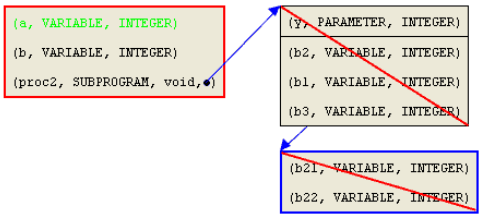

Figura 8. Búsqueda sin éxito en el entorno del cuerpo del procedimiento y éxito en la búsqueda en el entorno raíz.

3.2. Animación

La animación de la estructura de la tabla de símbolos consiste en la representación de las operaciones realizadas sobre ella a medida que se avanza en el proceso de análisis léxico y sintáctico.

Cada paso de la animación corresponde a una acción sobre la tabla de símbolos durante la compilación del programa. En el caso de las acciones de búsqueda, cada paso corresponde a la búsqueda en cada uno de los entornos. El proceso se inicia en el entorno actual y finaliza cuando la entrada es encontrada o en su defecto cuando se llega al entorno raíz.

Para controlar la animación se ofrecen los controles típicos: inicio, finalización, pausa, reproducción y selección de velocidad. Además se permite trasladarse al estado inmediatamente anterior o posterior con los controles "paso anterior" y "paso siguiente", o a un estado en concreto utilizando una barra de tiempo.

Para permitir que el usuario tenga siempre accesible información sobre las acciones que han ocurrido hasta el momento, se dispone de la zona de mensajes. En esta zona se muestra un mensaje por cada acción realizada sobre la tabla de símbolos, así como su resultado: creación de nuevos entornos, inserción de entradas y búsquedas. Además el usuario podrá elegir qué tipo de mensajes se mostrarán mediante filtros. Hay un filtro definido para cada tipo de acción que se puede realizar sobre la tabla de símbolos, si está seleccionado, se muestran los mensajes correspondientes a ese tipo de acciones, en caso contrario se omiten.

Cuando se selecciona un mensaje se resalta el punto del programa en el que se realizó la acción representada en el mensaje. Por ejemplo en la Figura 9 se puede ver que al seleccionar un mensaje de búsqueda con éxito, se resalta en el programa el token que provocó dicha búsqueda.

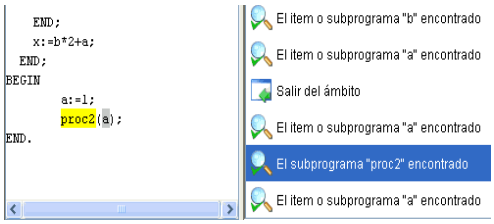

Figura 9. Resaltado de partes del programa relacionadas con los mensajes.

4. Discusión

Como ya se ha mencionado, existen múltiples estudios sobre la influencia de la visualización en la enseñanza de la informática. El contenido de la tabla de símbolos cambia durante el proceso de compilación, no es algo estático. Por tanto, disponer de una herramienta que permite mostrar el estado de la misma en cualquier punto del proceso puede ayudar en la comprensión de su funcionamiento.

Durante las explicaciones teóricas, el profesor puede hacer uso de la herramienta. Frente a técnicas tradicionales, como transparencias o pizarra, el uso de una herramienta de estas características permite ampliar las explicaciones en clase con múltiples ejemplos, avanzar o retroceder en el proceso de análisis con un mínimo esfuerzo y centrarse en las explicaciones teóricas de lo que está sucediendo contrastadas con un caso práctico real.

También creemos que es positivo que los alumnos dispongan de una herramienta que les permita afianzar y ampliar sus conocimientos mediante sucesivas pruebas con diferentes programas. Creemos importante que el/la alumno/a disponga, tanto de programas preparados por el profesor y que son accesibles fácilmente a través de la herramienta, como otros que el alumnado desee probar por su cuenta.

Para medir el grado de utilidad y de aceptación entre el alumnado se va a efectuar una evaluación en el actual curso académico enmarcada en la asignatura de Procesadores de Lenguajes de la titulación de Ingeniería Informática de la Universidad Rey Juan Carlos. La evaluación se va a realizar en dos fases: la primera estará compuesta de una sesión experimental de explicación de la tabla de símbolos; la segunda corresponde al análisis de la herramienta durante el resto del curso por parte de los/as alumnos/as.

Para la sesión experimental se dividirá a los/as alumnos/as en dos grupos: el de control y el experimental. La sesión constará de explicaciones teóricas y ejercicios prácticos. En el grupo experimental se utilizará la herramienta tanto para las explicaciones como para los ejercicios, y en el grupo de control se utilizarán los medios típicos como transparencias y pizarra. A ambos grupos se les realizará un test previo, para comprobar sus conocimientos iniciales. Tras la sesión se les realizará otro test con el objetivo de evaluar la utilidad de la herramienta.

Después de la sesión experimental, también se explicará la herramienta al grupo de control. Finalmente ambos grupos responderán a un cuestionario sobre la usabilidad de la herramienta e impresiones obtenidas del uso de la misma.

5. Conclusiones y trabajo futuro

Hemos desarrollado una herramienta de visualización del funcionamiento de la tabla de símbolos que sirve tanto al profesor para las explicaciones teóricas como al alumno/a para afianzar sus conocimientos y experimentar con ella. El docente puede proporcionar al alumnado múltiples y variados ejemplos que son accesibles a través de la herramienta. Creemos que el hecho de que se presente visualmente el funcionamiento de la tabla de símbolos relacionado con el proceso de análisis del código fuente facilitará la comprensión de los conceptos teóricos.

La herramienta está disponible vía web en la dirección http://vido.escet.urjc.es/sota. Además, se distribuye bajo licencia de software libre GPL.

No obstante, puesto que es necesario comprobar empíricamente la efectividad de la herramienta en los aspectos mencionados, se están ultimando los detalles para la realización de la sesión experimental durante el actual curso académico. Además, en la página web de la herramienta se encuentran disponibles dos cuestionarios que nos permitirán conocer la opinión de los profesores y alumnos que la usen. De esta forma, podemos mantener una evaluación continua de la misma.

En la herramienta se podrían introducir mejoras útiles en el proceso de aprendizaje como ofrecer al alumnado la posibilidad de experimentar con diferentes lenguajes de programación para que observen las diferencias entre ellos. También sería interesante que se pudieran establecer diferentes políticas de la tabla de símbolos, por ejemplo no permitiendo declarar variables con el mismo nombre en entornos anidados. Cabe también generalizar SOTA para que el propio usuario (profesor o alumno) pueda incorporar diferentes lenguajes suministrando los analizadores léxicos y sintácticos y usando una librería que la herramienta proporcionase para usar la tabla de símbolos. Finalmente, se está

estudiando la posibilidad de incorporar una nueva visualización de la tabla de símbolos desde el punto de vista de su implementación (por ejemplo mostrando las tablas hash).

Para permitir una mejor evaluación del uso que los/as alumnos/as hacen de la herramienta, sería deseable que pudiera generar un registro del uso de la misma. Esto es especialmente útil durante la sesión experimental de la herramienta con el alumnado, pues permitiría determinar pautas de aprendizaje.

Desde el punto de vista de la implementación, existen muchos aspectos del control de la animación que pueden reutilizarse en otras herramientas similares. Podrían generalizarse estos aspectos creando un conjunto de librerías que permitiera la construcción de las mismas con menos esfuerzo.

Referencias

[1] Anderson, J.M., Naps, T.L. *A context for the assessment of algorithm animation systems as pedagogical tools*. In Proceedins of the First Program Visualization Workshop (Porvoo, Finland, 2001), pp. 121-130

[2] Baecker, R. *Sorting Out Sorting: A Case Study of Software Visualization for Teaching Computer Sceince*. In Stasko, J.T. et al. (eds.), Software Visualization, MIT Press, Cambridge, Ma , USA, 1998, pp. 369-381

[3] Cavalcante, R., Finley, T., Rodger, S.H. *A Visual and Interactive Automata Theory Course with JFLAP 4.0*. Thirty-fifth SIGCSE Technical Symposium on Computer Science Education, 2004, pp.140-144

[4] Chen, T., Sobh, T.M. *A tool for data structure visualization and user-defined algorithm animation*. In Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, Reno, US, 2001

[5] Grissom, S. et al. *Algorithm Visualization in CS Education: Comparing Levels of Student Engagement*. ACM SOFTVIS 2003, pp. 87-94

[6] Hundhausen, C.D., Douglas, S.A., Stasko, J.T. *A Meta-Study of Algorithm Visualization Effectivenes.* Journal of Visual Languages and Computing, 2002, 13, pp. 259-290

[7] Lovato, M.E., Kleyn, M.F. *Parser visualizations for developing grammars with yacc*. In Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education, Nashville, USA, 1995, pp. 345-349.

[8] Moreno, A. et al. Visualizing programs with Jeliot 3. In Proceedings of the working conference on Advanced Visual Interfaces, Gallipoli, Italy, 2004, ACM Press, pp. 373-376

[9] Naharro-Berrocal, F. et al., Redesigning the animation capabilities of a functional programming environment under an educational framework. In M. Ben-Ari (ed), Proceedings of the second Program Visualization Workshop, Aarhus, 2002, pp. 60-69

[10] Naps, T. et al. *JHAVÉ -- An Environment to Actively Engage Students in Web-based Algorithm Visualizations*. In Proceedings of the SIGCSE Session, ACM Meetings, Austin, Texas, March 2000, pp. 109-113

[11] Naps, T. et al. *Exploring the role of visualization and engagement in computer science education*. ACM SIGCSE Bulletin, Junio 2003, 35(2), pp. 131-152

[12] Naps, T. et al. *Evaluating the Educational Impact of Visualization*. ACM SIGCSE Bulletin, Diciembre 2003, 35(4), pp. 124-136

[13] Resler, R.D. Deaver, D.M. *VCOCO: a visualisation tool for teaching compilers*. In Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education, Dublin City Univ., Ireland, 1998, pp. 199-202.

[14] Rodger, S. H., *Using Hands-on Visualizations to Teach Computer Science from Beginning Courses to Advanced Courses*. Second Program Visualization Workshop, Hornstrup Centert, Denmark, June 2002, pp. 103-112

[15] Rößling, G. et al. *Enhanced Expressiveness in Scripting Using AnimalScript V2*. In Proceedings of the Third International Program Visualization Workshop, Warwick, England, 2004, Ari Korhonen (Ed.), pp. 10-17

[16] Stasko, J.T., Lawrence, A. *Empirically assessing algorithm animations as learning aids*. In Stasko, J.T. et al. (eds.), Software Visualization, MIT Press, Cambridge, Massachusetts, USA, 1998, pp. 419-438.

[17] http://www2-data.informatik.unibw-muenchen.de/Research/Tools/JACCIE/