Depósito Legal ppi 201502ZU4668

CIENCIA

Vol. 26, No 1, 2
Enero - Junio 2018

An International Refereed Scientific Journal of the Facultad Experimental de Ciencias at the Universidad del Zulia

Esta publicación científica en formato digital es continuidad de la revista impresa
Depósito Legal: pp 199302ZU47
ISSN: 1315-2076

CIENCIA 26 (1,2), 15 - 22, 2018
Maracaibo, Venezuela

SITUALIZ: Software para la simulación de planes situacionales

Noelia Cegarra Ospino, Gerardo Pirela Morillo*

cegarranoelia@gmail.com, gepirela@fec.luz.edu.ve
Laboratorio de Lenguajes y Modelos Computacionales. Facultad Experimental de Ciencias.
La Universidad del Zulia

Recibido: 22-12-2017 Aceptado: 27-01-2018

Resumen

La planificación automática estudia la construcción de planes como secuencias de acciones, dado un conjunto de precondiciones y efectos, para llegar a satisfacer una meta. La lógica situacional ofrece un lenguaje formal para modelar y razonar sobre situaciones: objetos, acciones con precondiciones y efectos, y las condiciones para que las metas sean satisfechas. Asimismo, el cálculo situacional define mecanismos computacionales para la construcción de planes, aplicando razonamiento automático sobre los axiomas de la lógica situacional. Se desarrolló una herramienta de software que permite, a través de una interfaz gráfica, definir los elementos de lógica situacional para casos generales; se implementó un planificador situacional basado en búsqueda en anchura para garantizar hallar planes con la menor cantidad de acciones posibles; se integró un visualizador para el grafo y el plan situacional explorado hasta el momento de satisfacer la meta. Siguiendo una metodología de ciclo de vida de software, se utilizó Prolog para la representación e implementación de lógica y cálculo situacional, Java para la interfaz gráfica, TuProlog para la conexión entre la interfaz y el planificador, y GraphViz para la visualización de los grafos y planes situacionales.

Palabras clave: Planificación automática, lógica situacional, grafo de situaciones, Prolog, GraphViz.

SITUALIZ: Situational planning simulation software

Abstract

Automatic planning studies building plans as action sequences to satisfy goals given a set of constraints such as preconditions and other conditions depending on specific modeling strategies. Situational logic provides a formal framework to define situations: their elements, actions, and constraints (preconditions and effects), as well as the conditions to call all goals satisfied. Situational calculus provides computational approaches to build situational plans reasoning over the axioms of situational logic. The resulting software provides a graphic interface to define situations' elements, both the initial and goal situations, and the permitted actions with their preconditions and effects; the core situational planner was implemented based on the breadth-first search algorithm to guarantee finding fewest-action (shortest-length) plans. The software also displays graphical rendering of the situational plans which the algorithm explores up to the point where the goal situation is found, as well as the fewest-action plan found. Following a software life-cycle methodology, the authors used Prolog for representing and implementing situational logic and calculus, Java for the graphical interface, TuProlog for the connection between the interface and the situational planner, and GraphViz for rendering the situational graph and the resulting plan.

Keywords: Automatic Planning, Situational Logic, Situational Graph, Prolog, GraphViz

Introducción

De los casi 70 años de historia de la inteligencia artificial, en las últimas dos décadas ha alcanzado un auge muy importante que la han vuelto ubicua en casi todos los aspectos de la vida cotidiana y a la cabeza en los principales avances científicos y tecnológicos. Dentro de ésta, la planificación automática se encarga de la construcción de principios computacionales para la representación de objetos y acciones permitidas (con sus precondiciones y efectos) en el entorno y la subsecuente construcción de planes en la forma de secuencias de acciones para llegar de una condición inicial a la satisfacción de metas (1).

El estudio de técnicas de programación lógica, razonamiento automático y especialmente planificación automática es fundamental dentro del estudio más amplio de la inteligencia artificial.

Planificación automática sigue siendo un tema importante en el currículo de inteligencia artificial. Herramientas de software que faciliten la edición, simulación y visualización de elementos de planificación automática serían de gran ayuda, tanto en el contexto académico como en el contexto exploratorio general de tales conceptos. Sin duda serían positivamente conducentes al proceso de enseñanza-aprendizaje.

Existen varias herramientas de apoyo a la enseñanza de temas importantes de inteligencia artificial y modelos computacionales. Por ejemplo, la suite Fenix-Logios-Asix consiste en tres herramientas desarrolladas para apoyar la enseñanza de conceptos fundamentales de teoría de la computación: Fénix permite la construcción, manipulación y simulación de autómatas finitos determinísticos y no determinísticos (2). Asix integra la manipulación y simulación de máquinas de Turing estándares y de acceso aleatorio (3). Logios incorpora la edición, simulación y visualización de modelos de *parsing* universal basados en gramáticas libres de contexto: proceso de cómputo del autómata de pila equivalente a la gramática y algoritmo YCK de la forma normal de Chomsky equivalente de la gramática dada (4).

Con el mismo espíritu de crear una herramienta para facilitar el proceso de enseñanza-aprendizaje de algunos conceptos básicos de planificación automática, se creó el software acá presentado. Se tomó de Logios la arquitectura general de conectar una interfaz gráfica de usuario con un motor de razonamiento automático sobre el que se implementó el planificador, así como la visualización del espacio de planes explorados a través de la conexión con un generador de grafos.

Planificación automática

Es una rama de la inteligencia artificial para la producción de planes como secuencias de acciones para alcanzar una meta (1). Para efectuar un plan es necesaria la creación de un leguaje que defina los estados iniciales, las acciones a tomar, el resultado de la toma de dichas acciones y la meta a alcanzar, y esto es realizado por parte de un planificador que trabaja con algoritmos que generan dichos planes.

Un planificador descompone el mundo en condiciones lógicas y representa los estados iniciales como una secuencia de literales positivos conectados y estos deben ser simples y sin dependencias funcionales; las acciones son especificadas en términos de las precondiciones que deben cumplirse antes de ser ejecutadas y de las consecuencias que se siguen cuando se ejecutan; la meta es representada como una secuencia de literales positivos y simples, un estado proposicional s satisface un objetivo g si s contiene todos sus elementos en g. El esquema general de acción consiste en tres partes:

- El nombre de la acción y la lista de parámetros.
- La precondición: unión de literales positivos sin dependencia funcional, estableciendo lo que debe ser verdad en un estado antes que una acción sea ejecutada.
- El efecto: unión de literales positivos sin dependencia funcional que describe cómo cambia el estado cuando la acción es ejecutada.

Russell y Norvig (1) indican que la planificación automática en el campo de la inteligencia artificial surgió de la investigación en búsqueda en espacio de estados, demostración automática de teoremas, entre otras áreas. Uno de los primeros planificadores basados en lógica de primer orden fue el STRIPS (de la Universidad de Stanfod). De éste empezaron a evolucionar versiones cada vez más sofisticadas, con aplicaciones principalmente en el área de robótica industrial. En la década de los 2000 se logró estandarizar un lenguaje de descripción de acciones y un lenguaje descripción de dominio. Se crearon competencias y se tipificaron los problemas de planificación automática y basada en grafos de espacios de búsqueda como NP-Duros. Las herramientas de planificación se volvieron cada vez más específicas a los dominios de aplicación.

Planificación situacional

La idea detrás de la planificación situacional es que los estados pueden ser definidos en términos de las acciones necesarias para llegar a ellos (5). Cada relación que pueda cambiar con el tiempo se

maneja dando una situación adicional al predicado correspondiente.

Según describen Russell y Norvig (1), en la planificación situacional solo se consideran ambientes completamente observables y acciones determinísticas. De tal manera que una situación se describe completamente en el lenguaje formal de lógica de primer orden, como la conjunción de hechos observables sobre los elementos del entorno y sus interrelaciones. Asimismo, se definen las acciones legales en cualquier situación, especificando sus precondiciones y efectos. Las precondiciones se definen como una lista de hechos que deben cumplirse en cualquier situación para poder tomar la acción descrita. Los efectos se describen como una lista de hechos que serán ciertos en la siguiente situación si se ejecutara la acción descrita.

Siguiendo este esquema, a partir de una situación dada S_i se pueden calcular las situaciones siguientes posibles con los efectos de las acciones válidas desde S_i, según los hechos que son ciertos en Si; cada una de dichas acciones potencialmente generaría exactamente una situación nueva S_j, dependiendo de los efectos de la acción tomada. Este tipo de cálculo se denomina cálculo situacional. De esta forma, se genera dinámicamente un grafo de situaciones, en el que los nodos son situaciones y cada arco es una acción legal para el nodo situación hacia el nodo situación resultante.

Definiendo una situación inicial y una situación meta, un planificador situacional es cualquier algoritmo que utilice cálculo situacional para generar dinámicamente el espacio de soluciones parciales (grafo situacional), hasta llegar a la situación meta o cumplir un criterio de divergencia que permita detectar si no existen planes posibles. De existir al menos un plan posible, la salida del planificador es un camino en el grafo de situaciones, desde la situación inicial hasta la situación meta; el plan se representaría como la secuencia de acciones indicadas por los arcos del camino.

Metodología

Para el desarrollo de esta investigación se utilizó la metodología del ciclo de vida para el desarrollo de sistemas (6). Esta metodología especifica que un sistema sigue una estructura bien organizada y claramente planteada a lo largo de cuatro fases fundamentales: investigación preliminar, diseño, desarrollo de la herramienta y pruebas del producto final.

Durante la fase de investigación preliminar se revisaron herramientas para el apoyo al proceso de enseñanza de temas específicos del área de modelos computacionales e inteligencia artificial. La herramienta descrita por (4) resultó particularmente pertinente para el presente trabajo por cuanto combina interfaz gráfica para definición de elementos, generación de código en lógica de primer orden y construcción de resultados en forma de grafos de espacio de búsqueda y caminos sobre dicho grafo. Se requirió, entonces, conectar un motor de razonamiento automático para el planificador, una herramienta gráfica para la visualización de los grafos y los planes situacionales, y una interfaz gráfica para la edición e integración con el planificador y el visualizador. La herramienta resultante se bautizó “Situaliz”.

Arquitectura de Situaliz

El planificador situacional se implementó en SWI-Prolog, utilizando un algoritmo de recorrido en anchura (7) para garantizar que, de existir un plan, el planificador halle uno de menor longitud (menor cantidad de acciones posibles entre la situación inicial y la meta). Para la integración de este módulo se migró el código del planificador a TuProlog: una versión de Prolog ligero aplicaciones e infraestructura distribuidas (8); también se creó un lenguaje de alto nivel para la especificación de los elementos de lógica situacional. Se usó el entorno de desarrollo IntelliJ IDEA, en conjunto con JavaFX Scene Builder, creando una interfaz amigable que ofrece al usuario la oportunidad de una interacción abierta con la herramienta. Se creó un módulo de visualización de grafos de situaciones con la ayuda de la herramienta Graphviz en su versión 2.38. Esta arquitectura a alto nivel se aprecia en la figura 1.

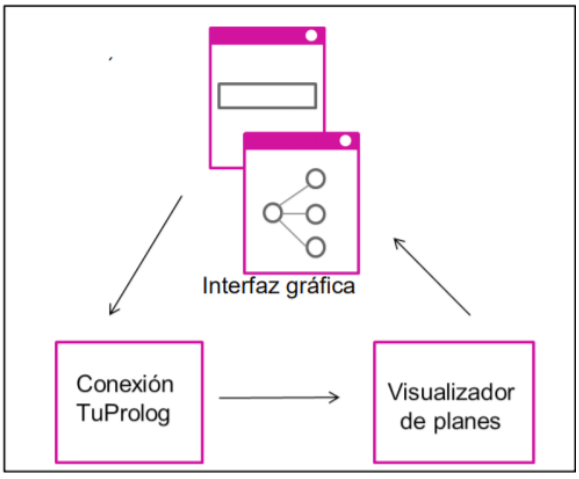

Figura 1. Arquitectura de Situaliz

La interfaz gráfica de la herramienta se desarrolló en el lenguaje de programación Java. Este módulo permite que el usuario edite los elementos de un plan situacional, especificando los elementos de

entorno, características del entorno, estado inicial, meta, acciones, precondiciones y efectos. Luego estos datos son convertidos al código Prolog que es ejecutado usando la herramienta TuProlog, produciendo el plan situacional en este módulo que posteriormente es enviado al visualizador de planes. En el módulo de visualización de planes se crea un archivo .dot a partir del plan situacional generado para así establecer comunicación con GraphViz y poder crear la imagen del grafo de situaciones que es mostrado al usuario por medio de la interfaz gráfica.

Situaliz fue diseñado con una interfaz minimalista que proporciona al usuario una interacción clara, haciendo sencillo el ingreso de los objetos necesarios para el planificador situacional: elementos y características del entorno, situaciones inicial y meta, acciones: precondiciones y efectos. Esta interfaz fue creada en Java, mediante el EDI IntelliJ IDEA y JavaFX Scene Builder; para el diseño de este módulo también se utilizó el leguaje FXML que es un lenguaje basado en XML, que provee la estructura para construir una interfaz de usuario separada de la lógica del código de la aplicación (9). La figura 2 muestra el menú principal de la herramienta.

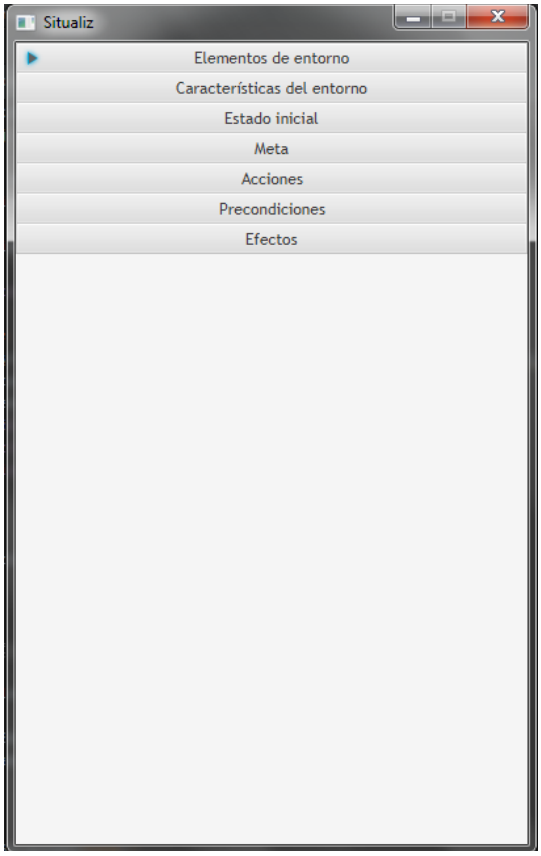

Figura 2. Ventana principal de Situaliz

Una vez que el usuario por medio de la interfaz haya definido los elementos básicos de un planificador situacional, se puede simular el planificador a través de la conexión con la librería TuProlog: las definiciones ingresadas por la interfaz gráfica se convierten en código Prolog con la sintaxis y el formato que espera el planificador situacional, el resultado de cuya ejecución contiene la información del grafo situacional y el camino que representa el plan hallado. Esta información es convertida a lenguaje .dot para que GraphViz lo interprete como grafo y se pueda visualizar en la ventana respectiva.

La visualización del grafo de situaciones que se genera con los datos que el planificador devuelve se muestra como una imagen generada con GraphViz, a partir del archivo .dot generado en la sección de simulación. Este grafo tiene coloreado de azul los nodos que representan las situaciones resultantes de la secuencia de acciones que llevan a la meta; las aristas están etiquetadas con la acción que conecta las dos situaciones. Esta ventana permite realizar zoom al grafo mostrado.

Resultados y discusión

Para mostrar un caso de uso desarrollado con Situaliz, se introduce el "Mundo de Bloques": sobre una mesa se disponen un número finitos de bloques, en una configuración inicial (algunos bloques sobre la mesa, otros bloques sobre algún otro bloque) y se desea realizar una secuencia de movimientos para llevar los bloques a una configuración meta. La figura 3 muestra un ejemplo de tal escenario.

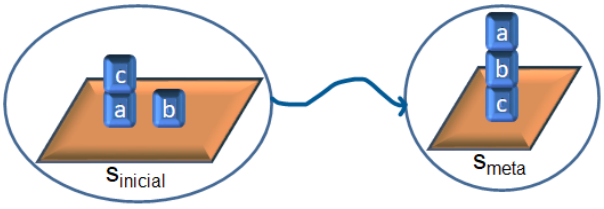

Figura 3. Mundo de Bloques – bloques etiquetados con las letras a,b,c sobre una mesa. La configuración de la izquierda es la situación inicial; la configuración de la derecha es la situación meta. Se desea llegar de una a la otra con movimientos sucesivos de bloques.

En este "Mondo de Bloques", los elementos del mundo serían la mesa y los bloques, identificados con alguna etiqueta distintiva; la descripción de las situaciones sería la conjunción de relaciones `sobre(X,Y)` que representaría que el objeto X (algún bloque) está encima del objeto Y (algún otro bloque o la mesa) y `despejado(X)` para indicar que

el objeto X (bloque o mesa) no está despejado (no tiene ningún otro bloque encima); la única acción permitida sería `mover(X,Y)` que representaría el movimiento del objeto X (algún bloque) hacia otro objeto Y (otro bloque o la mesa). Los planes serían, entonces, una secuencia de movimientos que permitan transformar la configuración descrita por la situación inicial en la configuración definida en la situación meta.

La figura 4 muestra la interfaz gráfica donde se definiría los bloques y la mesa, así como la definición de las relaciones para describir las situaciones. En el caso de uso, se definen tres bloques (a, b, c), la mesa (m) y las relaciones sobre y despejado. La notación sobre/2 indica que es una relación binaria: admite dos argumentos (el objeto indicado por el primer argumento está encima del objeto representado por el segundo argumento); mientras que despejado/1 indica una relación unaria: admite solo un argumento (el objeto indicado por el argumento no tiene ningún otro objeto encima).

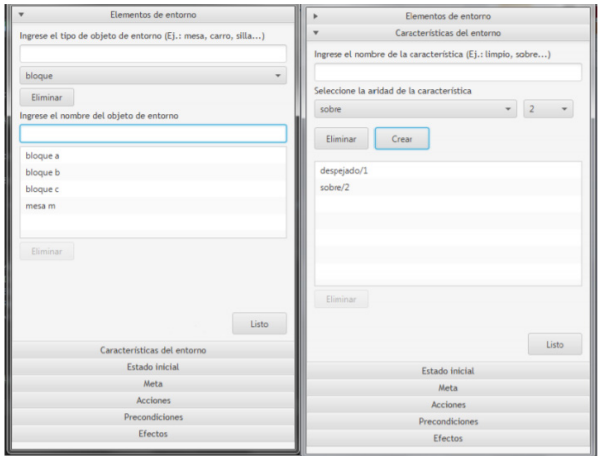

Figura 4. Definición de los elementos del mundo y las relaciones para definir las situaciones.

En la figura 5 se puede observar el uso de la interfaz gráfica para la definición de las situaciones inicial y final. En la situación inicial, los tres bloques están despejados sobre la mesa; la situación meta, los tres bloques están apilados sobre la mesa: a sobre b, b sobre c, c sobre la mesa (el hecho de que el bloque a está despejado resulta redundante en este caso).

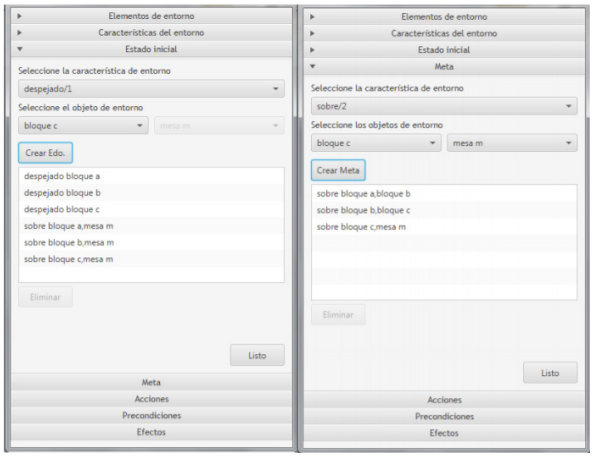

Figura 5. Definición de las situaciones inicial y meta.

Para definir la acción `mover(X,Y)` se deben definir primero las precondiciones que deben cumplirse en una situación para que se pueda ejecutar el movimiento del objeto X hacia el objeto Y: el objeto a mover, X, debe ser un bloque (no puede ser la mesa) y estar despejado; además, el objeto que recibirá el movimiento, Y, debe estar despejado. Luego se definen los efectos: en la situación resultante, X quedará sobre Y; Y ya no estará despejado (a menos que Y sea la mesa, la cual siempre tendrá espacio para recibir bloques), el objeto sobre el cual estaba X quedará despejad, X seguirá despejado y todos los demás bloques que no estuvieron involucrados en la acción permanecerán sin cambio. Estas definiciones, a través de la interfaz gráfica de Situaliz, se observan en las figuras 6 y 7.

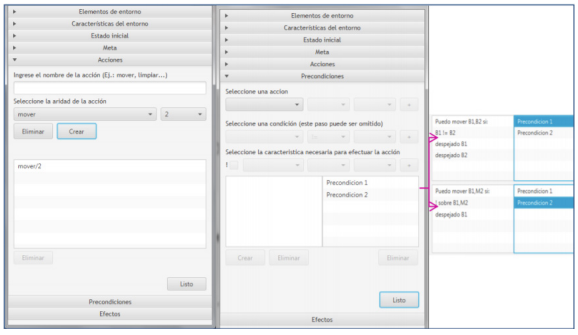

Figura 6. Definición de las acciones: precondiciones.

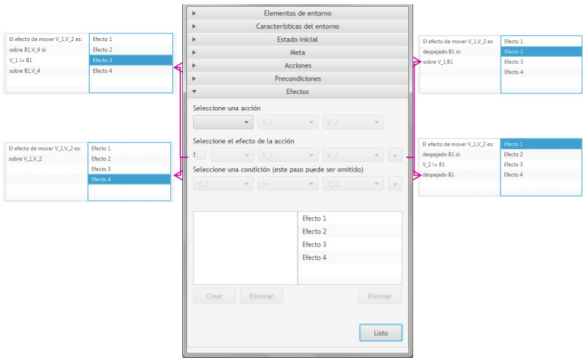

Figura 7. Definición de las acciones: efectos.

Una vez definidos todos los elementos de lógica situacional, el botón de "Listo" activa la generación del código Prolog, tal como se muestra en la tabla 1. Este código es integrado con el resto del planificador situacional basado en búsqueda de anchura y se ejecuta el planificador. En el proceso de salida, el grafo situacional es generado dinámicamente y, paso a paso, la salida es transformada en formato .dot, tal como se muestra en la tabla 2. El grafo resultante, en formato .dot, es pasado por GraphViz para generar una imagen que es visualizada en la ventana final del simulador, tal como se muestra en la figura 8.

Tabla 1. Código Prolog generado

% Elementos del mundo bloque(a). bloque(b). bloque(c). mesa(m).	% Situación inicial cierto(sobre(a,m),sito). cierto(sobre(b,m),sito). cierto(sobre(c,m),sito). % Situación meta: Meta=y(y(sobre(a,b), sobre(b,c)), sobre(c,m)).
% Acciones: Precondiciones puedo(mover(B1,B2),Sit):- bloque(B1), bloque(B2), B1\==B2, despejado(B1), despejado(B2) \+cierto(sobre(B1,B2),Sit). puedo(mover(B1,B2),Sit):- bloque(B1), mesa(B2), \+cierto(sobre(B1,B2),Sit).	% Acciones: Efectos cierto(sobre(B1,V_4), result(mover(V_1,V_2),Sit)):- bloque(B1), V_1\==B1, cierto(sobre(B1,V_4),Sit). % Se generan otros efectos

Tabla 2. Segmento de archivo .dot generado

Salida del planificador	Formato .dot equivalente
sito result(mover(a,b),sito) sito result(mover(b,a),sito) sito result(mover(b,m),sito)	"Sito" -> "Sit1" [label="mover(a,b)"]; "Sito" -> "Sit2" [label="mover(b,a)"]; "Sito" -> "Sit3" [label="mover(b,m)"];

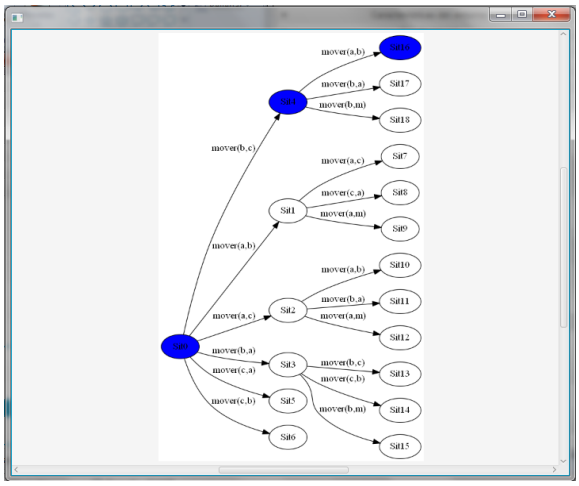

Figura 8. Grafo situacional con el plan resultante resaltado en azul sus nodos.

En caso de que no exista plan posible para llegar a la situación meta a partir de la situación inicial definidas, se muestra un error como el descrito en la figura 9. El software se diseñó con buenas prácticas de desarrollo de software, así que se captura y se reporta cualquier otro error que ocurra durante la edición o la simulación del planificador situacional.

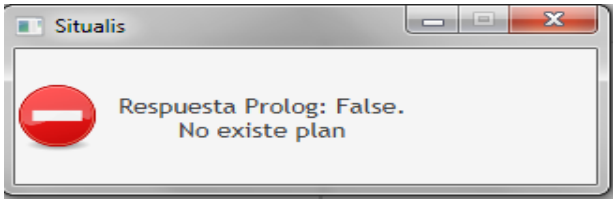

Figura 9. Respuesta cuando el planificador diverge y no halla plan válido.

Conclusiones

Situaliz se presenta como una herramienta de software para apoyo al proceso de enseñanza de cálculo y planificación situacional, en el contexto del área de planificación automática en el campo de la inteligencia artificial. Esta herramienta para el modelaje y simulación de resolución de problemas en estas técnicas de planificación situacional, a través de una interfaz gráfica intuitiva que permite la edición de los elementos de lógica situacional: objetos del mundo, acciones, precondiciones y efectos; luego simula la construcción de planes situacionales y genera la visualización del grafo de situaciones.

Durante el desarrollo de la herramienta se creó un lenguaje de alto nivel que facilita al usuario especificar de manera detallada los elementos de lógica situacional, por medio de una interfaz gráfica amigable. Luego de esta interacción se da paso a los procesos necesarios para realizar las consultas al planificador, para lo cual fue indispensable la comunicación entre los lenguajes Java y Prolog, lo que permitió que la herramienta tomara ventaja de dicha unión para su buen funcionamiento.

El planificador situacional se implementó en Prolog, basado en el algoritmo de búsqueda en anchura sobre el grafo de situaciones que se va generando dinámicamente. La búsqueda en anchura garantiza que el planificador se ejecute con una complejidad computacional temporal lineal sobre el tamaño del grafo (cantidad de nodos situaciones más cantidad de arcos acciones), además de garantizar que el planificador no explore ciclos de forma redundante y que el plan resultante siempre contenga la menor cantidad de acciones posibles.

Se construyó un módulo para la visualización del grafo de situaciones, que permite al usuario observar las acciones tomadas por el planificador y la secuencia que conduce a la solución. El resultado del planificador en Prolog es traducido a formato .dot que GraphViz convierte en imagen para ser visualizada en la ventana final del simulador.

Las pruebas funcionales a la herramienta permitieron detectar y corregir errores a lo largo del desarrollo de la misma. Se probaron varios casos de uso. En este manuscrito se describió el caso del mundo de bloque; también se probó otros casos de uso, que permitieron explorar las limitaciones tanto de la herramienta como de la planificación situacional. Por ejemplo, una limitante conocida del cálculo situacional es la necesidad de modelar explícitamente como efectos las características que no cambian de una situación a la siguiente, para poder calcularse nuevamente como parte de la construcción de la situación resultante.

Finalmente, se plantean las siguientes vías de trabajo futuro:

- Optimizar la interactividad de la herramienta, de manera que el usuario observe paso a paso las acciones tomadas por parte del planificador.
- Explorar los límites de lógica situacional con la que está programada el planificador, pudiéndose detectar los mismos a través de la interfaz.
- Limitar las iteraciones realizadas por parte del planificador mediante la detección de alguna entrada que conlleve a un siclo infinito.

- Darle continuidad a la investigación en esta materia para fortalecer las líneas de investigación de modelos computacionales y la simulación de agentes inteligentes.

Referencias bibliográficas

1. RUSSELL S., NORVING P. ***Artificial Intelligence: A Modern Approach.*** Prentice Hall. Kent (USA). 1152pp. 2009.
2. FERRER D. Herramienta Integrada para la construcción, manipulación y simulación de autómatas finitos (Para obtener el título de Licenciado en Computación). Facultad Experimental de Ciencias. Universidad del Zulia. Maracaibo (Venezuela). 99 pp. 2011.
3. CASPERSEN S. Herramienta integrada para la manipulación y simulación de m**áquinas de Turing** estándares y de acceso aleatorio (Para obtener el título de Licenciada en Computación). Facultad Experimental de Ciencias. Universidad del Zulia. Maracaibo (Venezuela). 127 pp. 2011.
4. NÚÑEZ H., PIRELA-MORILLO G. ***REDIELUZ***. 5(1 y 2): 49-58. 2015.
5. POOLE D., MACKWORTH, A. ***Artificial Intelligence: Foundations of Computational Agents.*** Cambridge University Press. Cambridge (USA). 820 pp. 2017.
6. http://analisideinformacion.blogspot.com/2012/09/metodologia-senn.html. Fecha de consulta: 05/12/2016.
7. CORMEN T., LEISERSON C., RIVEST R., STEIN C. ***Introduction to Algorithms.*** The MIT Press. Cambridge (USA). 1251 pp. 2010.
8. DENTI E. ***TuProlog Manual version: 2.9.0.*** Alma Mater Studiorum – Università di Bologna. Bologna (Italy). 255 pp. 2014.
9. https://www.java.com/es/about/. Fecha de consulta: 01/04/2017.

UNIVERSIDAD
DEL ZULIA

CIENCIA

Vol.26 Nº1, 2

Esta revista fue editada en formato digital y publicada en junio de 2018, por el ***Fondo Editorial Serbiluz, Universidad del Zulia. Maracaibo-Venezuela***

www.luz.edu.ve
www.serbi.luz.edu.ve
produccioncientifica.luz.edu.ve