Aportes para la Enseñanza de Teoría de Autómatas y Lenguajes Formales

Juan Carlos Vázquez, Leticia Constable

Departamento de Ingeniería en Sistemas de Información
Facultad Regional Córdoba, Universidad Tecnológica Nacional
Maestro López y Cruz Roja Argentina – 5016 Ciudad Universitaria – Córdoba
jcvazquez@frc.utn.edu.ar, lconstable@frc.utn.edu.ar

Resumen

La teoría de autómatas y lenguajes formales conforma la base conceptual de numerosas áreas de la Informática. Su contenido resulta abstracto por su fuerte sabor matemático, lo que dificulta la comprensión de los estudiantes de ingeniería en los primeros años. Como productos del proyecto *"Detección de errores sintácticos bajo el algoritmo de Earley"* se han desarrollado explicaciones detalladas, módulos de software y ejemplos de aplicación, que pueden ser de utilidad como elementos didácticos para la enseñanza de estos temas y para la práctica efectiva de la teoría en laboratorio. Además, se han desarrollado seminarios de actualización para docentes y una propuesta de cambio en la teoría de autómatas con el objetivo de acercar la teoría a la práctica real.

Palabras clave: *Autómatas, Lenguajes, Análisis Sintáctico.*

1. Identificación

EIUTNCO02168: Detección de errores sintácticos bajo el algoritmo de Earley.

Formalmente el proyecto se encuentra enmarcado en el anterior Programa de Electrónica, Informática y Telecomunicaciones de la Secretaría de Ciencia, Tecnología y Posgrado de nuestra Universidad, debido a su primer objetivo general (técnico-científico), aunque por su segundo objetivo (académico) se podría identificar con el tema prioritario del Programa de Tecnología Educativa y Enseñanza de la Ingeniería denominado "la didáctica en la Universidad y la práctica docente universitaria".

Fechas de inicio y finalización: 01/01/2014 y 31/12/2016, con prórroga al 31/12/2017.

El proyecto se formula y desarrolla en la Facultad Regional Córdoba de UTN.

2. Introducción

Toda carrera de Informática cuenta con algún curso introductorio de teoría de lenguajes y autómatas; en las ingenierías este curso aporta al currículo fundamentos de lcomputación indispensables para el perfil profesional y además exigidos por los actuales estándares. Dependiendo del plan de la carrera, los conocimientos logrados se limitarán sólo a fundamentos teóricos, o eventualmente se profundizarán en otras materias de diseño de herramientas de software de base, construcción de compiladores, inteligencia artificial, métodos formales de ingeniería de software u otras, que aplican la teoría a temas específicos. Muchas veces, estas otras asignaturas son optativas y por ello no todos los alumnos las cursarán.

Pero el modo de pensar acerca de los problemas, de enfrentarlos y de solucionarlos "*al estilo ingenieril"* (esto es, usando lógica, matemática, procedimientos efectivos, haciendo estudios de factibilidad y de eficiencia de las soluciones propuestas) que enseña la teoría de autómatas y lenguajes formales, es aplicable en general a todas las áreas de la disciplina. Por lo que, no solo es importante enseñar los fundamentos (Ciencias de la Computación), sino lograr en los estudiantes

la destreza y las habilidades de aplicarlos en situaciones reales (tecnología asociada), ya sea que cursen o no otras asignaturas electivas que los utilicen.

Hace tiempo que los profesores de la asignatura *Sintaxis y Semántica de los Lenguajes* (SSL), correspondiente al tercer cuatrimestre de la carrera de Ingeniería en Sistemas de Información en nuestra Facultad, vienen desarrollado distintos proyectos de investigación y desarrollo con fines mixtos: a) técnica y científicamente, investigar problemas no resueltos de la teoría de autómatas y lenguajes para aportar nuevos conocimientos en ese campo (como todo proyecto de investigación); b) académicamente, lograr conocimiento práctico real en el uso de los conceptos teóricos, generando nuevas herramientas didácticas y obteniendo experiencia en aquellas ya existentes, para aprovecharlas en la enseñanza y que sirvan como elementos de práctica que, en los alumnos noveles de segundo año, mejoren la comprensión de temas difíciles y abstractos, y ayuden a conseguir ese "modo ingenieril de pensar y de solucionar problemas".

Este artículo se refiere a experiencias y propuestas logradas durante el desarrollo de uno de estos proyectos: *Detección de errores sintácticos bajo el algoritmo de Earley*.

En 1970, el psicólogo Jay Earley presentó en su tesis doctoral (Earley,1970) el algoritmo de análisis sintáctico que lleva su nombre y que permite analizar cadenas de símbolos de cualquier lenguaje independiente del contexto, sin imponer restricciones a sus reglas gramaticales. Sin embargo, el algoritmo es no determinista y se consideró poco eficiente en su momento como para ser puesto a funcionar en computadoras (Aho,1990), a pesar de que Earley analizó su complejidad máxima como $O(n^3)$ al operar sobre gramáticas ambiguas. Por otro lado, en su artículo a la comunidad, el autor especificó el funcionamiento del algoritmo, pero no cómo utilizarlo para indicar específicamente los errores en las cadenas analizadas.

Trabajos posteriores (McLean,1996), (Aycock,2002), (Trevor,2010), han propuesto formas más eficientes del algoritmo de Earley, pero no se encuentran fácilmente referencias respecto de su uso efectivo en un lenguaje de programación, y en especial, del tema de cómo informar errores detectados. Por ello se llega al planteo de las siguientes preguntas:

- ¿Cómo se puede indicar la presencia de un error en una cadena de entrada de un lenguaje independiente del contexto, usando el algoritmo de análisis sintáctico de Earley?
- ¿Qué tan específica (en cuanto a su localización y causa) puede ser esta indicación para que se pueda entender y corregir el error?
- ¿Cuán eficiente puede hacerse esta operación?

Como investigadores, nos preocupan estas preguntas *per sé*, pero además nos interesan en comparación con la respuesta que dan a las mismas los clásicos algoritmos de análisis sintáctico predictivo (descenso recursivo, LL(k) y LR(k)). Por ello también se cree aconsejable tener estos algoritmos claramente implementados para lograr contrastar funcionamientos y desempeño entre ellos y el de Earley.

Se piensa en la programación de estos algoritmos, ya que habiendo utilizado los generadores de compiladores (lex, flex, jflex, yacc, byacc, bison, jcup y otros), puede decirse que incorporan tantas alternativas debido a su gran versatilidad y al largo tiempo de vida, que los analizadores lexicográficos y sintácticos que producen no son manejables por alumnos con poca experiencia en programación, y son ciertamente difíciles de analizar aún para profesionales con experiencia.

Como docentes de la cátedra SSL y coautores del libro (Giró,2015) en uso actualmente en la misma, durante largo tiempo hemos estudiado y enseñado en forma introductoria temas de lingüística matemática, teoría de autómatas, compiladores y complejidad. La materia es del segundo año de la carrera, los

alumnos son recién iniciados en programación y los temas abordados bastante abstractos. Una estrategia para fijar conocimientos y bajar las abstracciones a la realidad es el uso de simuladores (tanto de desarrollo local –proyecto GHD de UTN-FRC, simulador de Máquina de Turing de la cátedra SSL– como externos), para que los alumnos comprueben la ejercitación propuesta y realicen trabajos prácticos; otra estrategia es hacer que los alumnos programen sus propias soluciones, pero aún no están preparados para ello en nuestro caso. Sin embargo, se estima que sí están preparados para entender un código al verlo, por lo cual una tercera posibilidad sería mostrar las soluciones ya desarrolladas, tal vez con errores a corregir, comentarios a completar, o sencillamente para discutir su funcionamiento paso a paso. En esto pueden inclusive ayudar herramientas de visualización del proceso de análisis sintáctico (Almeida,2011), las cuales han sido identificadas por nuestro equipo de trabajo, aunque aún no evaluadas.

Es de interés por lo anterior, aprovechar este proyecto para efectivizar la construcción de código sencillo y claro de analizadores léxicos y sintácticos de distinto tipo, en lenguajes que el alumno conozca, con el fin de generar la tercera alternativa indicada, que se cree redundará en una mejor comprensión de los temas impartidos en la asignatura SSL.

Se intenta que estos programas (práctica) sigan al pie de la letra la teoría, para mapear una en otra muy claramente. Sin embargo, hemos visto que no puede transitarse directamente de la teoría que enseñamos a la práctica en máquina; deben agregarse conceptos y técnicas que no están en la teoría como *el principio de la subcadena más larga* y *la incorporación de símbolo de fin de cadena* en las definiciones de autómatas, hechos detallados en nuestros anteriores artículos, indicados en el quinto punto del presente trabajo.

3. Objetivos, Avances y Resultados

Como se indicó el proyecto tiene un objetivo científico/técnico:

- Determinar con qué especificidad pueden informarse errores sintácticos, usando el método de Earley de análisis sintáctico.

y uno académico:

- Generar código y explicaciones claras para acrecentar el material didáctico para la enseñanza de Sintaxis y Semántica de los Lenguajes.

El primero de ellos intenta responder las preguntas formuladas anteriormente; el segundo da origen al presente artículo.

Además de haber participado los integrantes del proyecto de la confección del libro en uso actualmente por la cátedra SSL, el proyecto ha logrado para el objetivo académico señalado los siguientes avances:

- Diseño formal del lenguaje de programación para la máquina RAM, con descripción detallada de su sintaxis y semántica.
- Ejecutor de programas escritos en lenguaje RAM, para PC de escritorio y en web.
- Analizador léxico para programas RAM.
- Analizador sintáctico por descenso recursivo de programas RAM.
- Analizador sintáctico LL(1) y LR(0) del lenguaje RAM.
- Documentos de explicación de funcionamiento detallado y fundamentos de los algoritmos de análisis sintáctico por descenso recursivo, LL(1), LR(0), SLR(1), LR(1) y de Earley, con ejemplos aplicados a varias gramáticas simples y a la gramática del lenguaje RAM.
- Distintos seminarios de capacitación sobre estos temas para docentes de SSL.
- Una propuesta de modificación de la teoría de autómatas, para su mejor enseñanza y didáctica en carreras de Ingeniería.
- Propuesta de creación de un Grupo I+D en la Facultad (Grupo de Investigación, Desarrollo y Transferencia en Aprendizaje Automático, Lenguajes y Autómatas), que genere un ámbito apropiado para reunir a proyectos de estudio de lenguajes.

- Diseño de un lenguaje de programación procedural simple en español, con estudio sobre su gramática y semántica.

Están en desarrollo los módulos de análisis sintáctico LR(1) y de Earley para el lenguaje RAM, y un sencillo análisis semántico, ya que el lenguaje intermedio que se generará en los distintos algoritmos es también RAM, por lo que puede correrse en el simulador ya desarrollado.

Tenemos como deuda aún, la aplicación de los algoritmos desarrollados al lenguaje de programación en español lo cual se espera pueda ser realizado en una segunda fase. Una derivación posible es también el estudio de la posible aplicación del algoritmo de Earley a la paralelización de la compilación, ya que se ha visto preliminarmente que estaría actuando como una red de Petri al procesar una cadena de entrada (un programa).

4. Formación de Recursos Humanos

El equipo de investigación está compuesto por docentes-investigadores que se desempeñan como profesores de teórico, práctico y ayudantes en la cátedra SSL de nuestra Facultad. Se ha contado además con el apoyo de tres becarios alumnos y de un graduado de la carrera de Ingeniería en Sistemas de Información.

Uno de los integrantes tiene en curso la Maestría en Sistemas de Información, siendo ya Magister en Docencia Universitaria.

Otro integrante es doctorando en ingeniería, mención Sistemas de Información, aunque su tesis no tiene relación directa con el proyecto.

5. Publicaciones relacionadas con el PID

Vázquez J., Constable L., Meloni B., Jornet W., Arcidiácono M., Parisi G. (2015); Detección de errores sintácticos bajo el algoritmo de Earley; Workshop de Investigadores en Ciencias de la Computación 2015; Salta, Argentina.

Vázquez J., Constable L., Jornet W., Meloni B. (2015); Enseñanzas de la Implementación de un Analizador Léxico; Congreso Nacional de Ingeniería Informática/Sistemas de Información 2015; Buenos Aires, Argentina.

Vázquez J., Constable L., Jornet W., Meloni B., Carballo N. (2016); Enseñanzas de la Implementación de un Analizador Sintáctico por Descenso Recursivo; Congreso Nacional de Ingeniería Informática/Sistemas de Información 2016; Salta, Argentina.

Vázquez J., Constable L., Meloni B., Jornet W., Arcidiácono M. (2018); Propuesta de Cambio en la Teoría de Autómatas para Mejorar su Enseñanza en Ingeniería; Cuarta Conferencia Iberoamericana en Complejidad, Informática y Cibernética; Orlando, Florida, EE.UU.

Vázquez J., Castillo J., Constable L., Cardenas M. (2018); GAALA: Grupo de Aprendizaje Automático, Lenguajes y Autómatas; Workshop de Investigadores en Ciencias de la Computación 2018; Corrientes, Argentina.

Está enviado además para su evaluación este año 2018:

Vázquez J., Constable L. (2018); Detección de errores sintácticos bajo el algoritmo de Earley. Informe Final; Congreso Nacional de Ingeniería Informática/Sistemas de Información 2018; Mar del Plata, Buenos Aires, Argentina.

Se presentó también el proyecto en Jornadas Internas de I+D del Departamento de Ingeniería en Sistemas de Información de UTN, frente a docentes, investigadores y alumnos.

Además, en los ciclos de capacitación docente que la cátedra SSL realiza año a año, se han presentado algunos de los resultados del proyecto y de las propuestas generadas en el mismo para la enseñanza. También se ha convenido con otros proyectos de nuestra Facultad que utilizan lenguajes de aplicación específica, poner a disposición nuestras herramientas para que los mismos utilicen en sus desarrollos.

Todos los productos de software una vez pulidos y probados, serán registrados en el registro nacional de la propiedad intelectual

y serán puestos a disposición de la cátedra SSL para su evaluación y uso.

Referencias

Aho A., Sethi R., Ullman J. (1990); *Compiladores: principios, técnicas y herramientas*; Addison Wesley Iberoamericana S.A.; D.F., México.

Almeida Martínez F. (2011); *Generación de visualizaciones educativas del análisis sintáctico*; Tesis Doctoral, Dpto. de Lenguajes y Sistemas Informáticos I – Escuela Superior de Ingeniería Informática - Universidad Rey Juan Carlos; España.

Aycock J., Horspool N. (2002); *Practical Earley Parsing*; The Computer Journal, British Computer Society, Vol 45 – Nr. 6 – pp 620-630; Apr 2002; UK.

Earley J. (1970); *An Efficient Context-Free Parsing Algorithm*; Communications of ACM, Vol. 13 – Nr. 2 – pp 94-102; Feb 1970; NY, USA.

Giró J., Vázquez J., Meloni B., Constable L. (2015); *Lenguajes Formales y Teoría de Autómatas*; Alfaomega; Buenos Aires, Argentina.

McLean P., Horspool N. (1996); *A Faster Earley Parser*; Proceedings of International Conference on Compiler Construction, Springer, pp 281-293; 1996; Canada.

Trevor J., Mandelbaum Y. (2010); *Efficient Early Parsing with Regular Right-Hand Sides*; Electronic Notes in Theoretical Computer Science, Elsevier; Vol. 253 – Nr. 7 – pp 137-148; Amsterdam, The Netherlands.