Hacia la generación de reconocedores LR(1) de reducidas dimensiones

Fortes Gálvez, José
Departamento de Informática
Universidad de Las Palmas de Gran Canaria
Islas Canarias (España)

Abstract

Los métodos clásicos LR(1) presentan el problema de producir tablas de reconocimiento demasiado grandes para las gramáticas de lenguajes de programación. Una alternativa es construir un autómata que combine el símbolo de prelectura con la lectura de la pila del análisis sintáctico desde su cima, para determinar la siguiente acción de reconocimiento a realizar.

Aquí se exponen, junto a los desarrollos teóricos que han conducido a un método para construir este tipo de reconocedores, una breve descripción del mismo y de los resultados experimentales, siendo el aspecto más destacable las importantes reducciones que se han observado en tamaños de autómatas frente a los métodos estándar.

Palabras clave: análisis sintáctico, gramáticas LR(1), generadores, autómata inverso, procesadores de lenguajes

1 Introducción

Los métodos clásicos [11, 1, 17] de construcción de analizadores sintácticos para la clase de gramáticas LR(1) presentan el problema de producir autómatas o tablas de reconocimiento de enormes dimensiones en comparación con los tamaños de las gramáticas. Ello es especialmente grave para el caso de gramáticas de lenguajes de programación. Ante este problema, la solución que se ha adoptado ha sido la de utilizar reconocedores LALR(1), aún a costa de una pérdida en la capacidad de reconocimiento.

En los métodos de precedencia [15, 18, 9, 4] se utiliza en principio el par de símbolos (cima de la pila, ventana de prelectura) para decidir si se debe avanzar—almacenar el símbolo de la ventana en la cima—o reducir. El problema de lo limitado de la aplicabilidad de estos métodos radica en que es muy frecuente encontrar gramáticas en las que, para algunos casos, es necesario conocer algo más del contenido de la pila, lo que ha dado lugar a soluciones algo más potentes [5, 7, 12],

pero aún muy lejos de LR(1). A esto se suele añadir una limitación adicional en cuanto a las partes derechas de la gramática, y en particulár las reglas nulas están siempre prohibidas.

El enfoque que aquí se toma parte de la misma noción anterior, ampliamente experimentada como válida dado el gran número de métodos desarrollados a lo largo de muchos años, pero generalizándola hasta sus últimas consecuencias. Así, el método admite que, cuando sea necesario la pila se explore en la profundidad precisa, de forma que siempre se alcance una solución, por lo que el método garantiza la potencia LR(1). No se aplica ninguna restricción a las reglas de la gramática, y el analizador no sólo habrá de discriminar entre avance o reducción sino que, en este último caso, indicará qué regla aplicar.

En este artículo se introduce brevemente este novedoso método [6], que presenta las características antes citadas, junto con el desarrollo previo que nos ha llevado al mismo, para la generación de reconocedores LR(1) de pequeño tamaño.

2 Planteamiento del problema

Justo antes de la aparición de los métodos LR, el método que mejor compromiso [1] aportaba entre sencillez y eficiencia, por un lado, frente a potencia de reconocimiento, por otro, era el método de precedencia débil.

Su concepto base es muy claro: relación entre el símbolo superior de la pila y el de prelectura, con incidencia directa sobre el proceso de análisis ascendente, de forma que la relación $\lessdot$ implica avance, y la relación $\gtrdot$ implica reducción. El cálculo de tales relaciones es relativamente sencillo, y su implementación eficiente. Además, se conocen buenos métodos para incorporar el tratamiento automático de errores [8] al proceso de análisis. Incluso los posibles problemas de conflictos entre relaciones pueden ser solventados, aunque algo inelegantemente, mediante la introducción de nuevos símbolos no terminales.

Sin embargo, la precedencia débil presenta dos problemas que los métodos LR resolvieron de forma muy elegante. El primero de ellos la posibilidad de disponer de reglas nulas en la gramática, cosa que hasta entonces no se entendía muy bien, sobre todo desde el punto de vista de su utilidad práctica; concepto que hoy está claramente resuelto. Y el segundo, la necesidad práctica de incorporar algún mecanismo adicional para resolver los conflictos reducción-reducción, si no queremos estar realizando continuas transformaciones de reglas en la gramática. Es, fundamentalmente, con el objetivo de abordar este segundo problema, con el que surgen los métodos de exploración del contexto izquierdo [5, 7, 12].

Pero el impacto de los métodos LR es demasiado fuerte, y prácticamente hace desaparecer cualquier esperanza de que, trabajando en la mejora de los métodos de precedencia, pueda algún día alcanzarse algo comparable. Esto es especialmente cierto al aparecer el método LALR [3]. Posiblemente ha influido también su incorporación en forma de la utilidad `yacc` al sistema operativo Unix, y con ello su popularización y su conversión por tanto en método estándar de facto.

3 Mejorando los métodos de precedencia

Vamos, a pesar de todo, a plantearnos la posibilidad, aunque sólo sea circunscribiéndonos inicialmente a la exploración teórica, sin relevancia inmediata en el campo de su aplicación práctica, de la mejora de los métodos de precedencia, y en particular, de la precedencia débil. Así, de las dos desventajas que hemos mencionado para la precedencia débil, y que por tanto son los candidatos inmediatos a un estudio teórico para su mejora, podemos hacer unas primeras reflexiones.

En el caso de las reglas nulas, podemos decir que no son intrínsecas a la idea misma de precedencia, sino que más bien son consecuencia de la moda, por decirlo de alguna manera, imperante en el momento en que se desarrollan estos métodos. Así, cabe perfectamente replantearse la existencia de relaciones de precedencia débil, por ejemplo, que impliquen avance o reducción, considerando la posible existencia de reglas nulas.

Y en el caso de la exploración del contexto izquierdo para la resolución de conflictos reducción-reducción, podemos decir que se trata, simplemente, de que la precedencia débil pretende fundamentalmente permitirnos discernir entre cuándo se ha de avanzar, y cuándo reducir. Por tanto, este mecanismo está abierto a mejoras para la resolución de los citados conflictos.

Veamos, por tanto, cada uno de estos aspectos por separado.

3.1 Incorporación de reglas nulas

Corrobora nuestras afirmaciones anteriores constatar que solamente desde época muy reciente, ha recibido atención el estudio de la incorporación de reglas nulas a los métodos de precedencia, en particular [14, 13]. Aquí expondremos brevemente un método distinto, aunque con los mismos resultados finales, desarrollado independientemente por el autor.

A diferencia de [14], donde se desarrolla un nuevo mecanismo de cálculo de unas nuevas relaciones de precedencia, adaptadas a gramáticas con reglas nulas, nuestro interés es mantener una compatibilidad en los mecanismos de cálculo tradicionales, con las correcciones oportunas para contemplar las reglas nulas. Así, introducimos el concepto de *relación de regla nula*, de forma que todo símbolo no terminal que produce la secuencia nula ε introduce la correspondiente relación de precedencia. Más concretamente, decimos que dos símbolos $X \in V$, $c \in T$ están *N-relacionados* si $N \rightarrow \varepsilon$ y existe una forma sentencial canónica en la que X está en la cima de la pila, c es el símbolo de prelectura, y el handle ε ha de ser reducido a N. De esta forma, suponiendo que no haya conflicto, el reconocedor puede decidir si avanzar ($\lessdot$), reducir ε (N-relación) o reducir una regla no nula ($\gtrdot$).

Para el cálculo de las relaciones de regla nula introducimos la noción de gramática con terminales de regla nula, donde cada regla nula $N \rightarrow \varepsilon$ es sustituida por $N \rightarrow \mathcal{E}_N$, donde $\mathcal{E}_N$ es ahora considerado un símbolo terminal, de forma que la gramática queda convertida a gramática sin reglas nulas, para conservar el esquema tradicional de cálculo, como era nuestro objetivo inicial. En este mismo sentido,

es destacable que las nuevas relaciones son calculables a partir de las tradicionales relaciones de precedencia aplicadas a la gramática que incorpora los terminales de regla nula.

De la misma manera que con el caso tradicional, es posible realizar transformaciones automáticas mediante la introducción de nuevos símbolos no terminales, de forma que desaparezcan los conflictos de precedencia—en realidad quedan convertidos en problemas de contexto izquierdo, que es el tema que pasamos a estudiar a continuación. En cualquier caso, es destacable que la aplicabilidad práctica de la precedencia débil aumenta en sí misma notablemente con la incorporación de reglas nulas, siempre que estemos dispuestos a asumir la algo fastidiosa transformación de reglas.

3.2 Resolución de conflictos reducción-reducción

Es necesario abordar la resolución de conflictos reducción- reducción para una verdadera utilidad práctica del método, en comparación a los conocidos actualmente. Ya hemos indicado que en su momento se desarrollaron métodos que hacen uso de un número limitado- -y necesariamente pequeño por condicionantes propios del diseño—de los símbolos en la cima de la pila, para este propósito. Igualmente, sólo recientemente ha recibido de nuevo atención esta línea de trabajo, en particular en el tratamiento de errores [16, 2].

Sin embargo, está claro que por esta vía será difícil obtener algo similar a la potencia de, por ejemplo, LR(1) por la sencilla razón de que, por definición, el método LR(1) utiliza toda la información disponible en la pila más el símbolo de prelectura para, no sólo decidir entre avanzar o reducir, sino además saber qué regla es la que debe utilizarse en la eventual reducción.

Planteémosnos pues, aunque sea en principio como ejercicio puramente teórico, la posibilidad—necesaria por otro lado por lo que acabamos de ver—de utilizar *toda* la información de la pila para decidir en conflictos reducción-reducción. Es fácil ver que, en este caso, y combinando lo visto en la sección precedente, nos encontraríamos con potencias de análisis sintáctico próximas (aunque quizá algo inferiores) a LR(1). Evidentemente, el posible resultado merece el esfuerzo de intentarlo.

La solución desarrollada por el autor es la de construir un autómata finito determinista que, explorando la pila desde su cima hacia la base, en el momento de aplicar una reducción, sea utilizable para discernir qué reducción concreta ha de aplicarse.

Está claro que de alguna manera este autómata, en una versión simplificada, estaba ya presente como alternativa a la solución del problema de decidir qué regla reducir en el caso de precedencia débil en su forma original. Además, nuestro autómata ha de incorporarlo, pues en los métodos de precedencia siempre habremos de reconocer el handle completo para asegurar que las reducciones que vamos realizando son sintácticamente correctas. No vamos a entrar en el detalle interno de la construcción del autómata por las razones que ahora se verán.

Es importante considerar que, en la mayoría de los casos es posible conocer qué regla reducir tras leer muy pocos símbolos, y sólo en casos muy particulares y muy poco corrientes en gramáticas para lenguajes formales utilizables por personas, como las de lenguajes de programación—que por otro lado permitirían notables simplificaciones adicionales—será necesario hacer una exploración más profunda en la pila. En definitiva, no sólo el autómata se prevé que sea considerablemente pequeño, especialmente en comparación con su equivalente directo LR(1), sino que su utilización más frecuente incurrirá en leer (casi) sólo hasta el comienzo del handle que vamos a reducir.

En cualquier caso, podemos hacernos una idea de la disminución de tamaño en relación al autómata directo si consideramos que éste ha de *reconocer* el lenguaje completo de posibles contenidos de la pila más prelectura, mientras que nuestro autómata precisa sólo reconocer las colas—normalmente de muy pequeña longitud, como hemos dicho—de dicho lenguaje que nos permiten discernir entre distintas reducciones.

De esta forma, podríamos decir que disponemos de la potencia LR(1) *para reducir*. Hemos llegado a un resultado importante—potencia global próxima a LR(1)—mediante la combinación de las técnicas ya expuestas. Sin embargo, hay una cierta inelegancia, un cierto desequilibrio, en el resultado global que nos hizo reflexionar. Fundamentalmente se trata, por un lado, de la transformación, aunque fácilmente automatizable, pero siempre fastidiosa de reglas para solventar conflictos de precedencia, y, por otro, de la descompensación entre la sencillez del mecanismo de relaciones de precedencia para decidir si avanzar o reducir y la relativa complejidad conceptual del autómata para decidir qué reducción aplicar.

La idea que surgió fue: ¿por qué no utilizar el autómata para el problema completo, en lugar de sólo para una parte? ¿No será quizá más simple que la adición de los dos métodos? De alguna manera, el autómata podría englobar el cálculo de precedencia, si el análisis que realiza partiera, no de la cima de la pila, sino del símbolo de prelectura[1]. Aún más, puesto que haríamos uso de la información completa de la pila más prelectura, tendríamos en teoría completa potencia LR(1), eliminaríamos las inelegancias anteriores, y mantendríamos prácticamente en su totalidad la ventaja del pequeño tamaño.

4 Un autómata discriminante

La idea básica parte de que una gramática LR(1) permitirá siempre que un reconocedor ascendente pueda decidir, conociendo únicamente el contenido de la pila y el símbolo de la ventana actual, qué acción tomar a continuación: avanzar, reducir de acuerdo con alguna regla, o indicar un error. Por otro lado, es sabido que el contenido de la pila será siempre una sentencia de un lenguaje regular, por lo que será posible construir, para toda gramática LR(1) un autómata reconocedor.

[1]Como de hecho así ocurría; aunque aquí, por mayor sencillez en la exposición lo hemos presentado como comenzando desde la cima.

El método aquí propuesto, por tanto, consiste en *iniciar* la construcción de un autómata que reconozca el lenguaje de los contenidos de la pila, pero calculando para cada estado las diferentes acciones compatibles con el mismo, de forma que, si la acción fuera única, el algoritmo de construcción se para en esta parte del autómata. Es justamente este aspecto el que nos va a permitir obtener drásticas reducciones de tamaño en comparación con los métodos clásicos. Para obtener justamente la potencia LR(1) se ha de considerar, obviamente, el símbolo de la ventana: efectivamente, en cada estado se analiza la compatibilidad combinada de la cola de pila ya leída con el símbolo de la ventana, hasta que se halle una única acción por estado.

4.1 Utilización para el análisis

En cada paso de análisis, la exploración de la pila comienza desde el estado inicial partiendo desde la cima. En cada estado se utiliza en primer lugar el símbolo de la ventana para tomar una decisión; si ello no fuera posible, se utiliza el siguiente—en el sentido desde la cima hacia la base—símbolo de la pila para tomar una decisión o, en su defecto, una transición a un nuevo estado. De hecho, hemos podido comprobar que, aunque en teoría pudiera parecer que el método da lugar a repetidas y profundas exploraciones de la pila, en la práctica no es así.

La exploración finaliza con dos posibilidades: bien se decide una acción (el análisis termina si ésta es reducir al símbolo principal aumentado), bien ésta no puede decidirse debido a un error presente en la entrada. Esta última condición se puede detectar porque el símbolo siguiente en la pila no presenta transición legal en el estado actual, o porque se llega a la base de la pila sin haber alcanzado una decisión.

5 Resultados experimentales

De acuerdo con el método presentado, se ha construido un generador de analizadores sintácticos. Los resultados prácticos coinciden con la potencia de análisis y con los tamaños de autómata previstos. Como aún no disponemos de un generador clásico LR(1) con el que hacer comparaciones a igualdad de potencia, hemos elegido para realizar las comparaciones el bien conocido generador Yacc [10]. A pesar de que éste genera autómatas mucho menores que los correspondientes LR(1), las comparaciones experimentales han sido incluso favorables en todos los casos, y del orden de un 30 a un 40 por ciento menores en número de estados de los correspondientes autómatas. Podemos citar que, en particular, la gramática mayor que ha sido posible comparar, una gramática para el lenguaje C completo obtenida de la base de datos del grupo de noticias `comp.compilers`, adaptada para que fuera exactamente LALR(1), dio el siguiente resultado: con 235 reglas, Yacc generó 439 estados, mientras que nuestro autómata únicamente 284 estados.

6 Conclusiones

Se han presentado brevemente las bases teóricas que nos han conducido al desarrollo de un método para la construcción y utilización de un autómata que, recorriendo la pila desde la cima hacia la base, es capaz de decidir la siguiente acción sintáctica a tomar, en combinación con el símbolo de la ventana de prelectura.

Ello ha permitido obtener un nuevo analizador sintáctico para la clase de gramáticas LR(1). Dado que el método de construcción es discriminante, terminándose el proceso de construcción de estados desde el primer momento en que ello es posible, se han conseguido obtener muy importantes reducciones de tamaño en comparación con los métodos clásicos LR(1), manteniéndose los aspectos negativos dentro de márgenes razonables, poco relevantes desde un punto de vista práctico.

Desde un punto de vista teórico, se generaliza la idea de precedencia, aprovechando las ventajas que la intuición indicó a los primeros diseñadores de éstos métodos, abriéndose una nueva vía inexplorada de interesantes posibilidades teóricas y prácticas.

Agradecimientos

Este trabajo debe mucho a discusiones con Olivier Lecarme y Jacque Farré, del Laboratorio I3S del CNRS y la U. Niza-Sophia Antipolis (Francia), y al acogedor entorno del laboratorio. Desarrollado gracias a una ayuda del Gobierno y la Caja de Canarias. Quisiera reconocer especialmente a mi esposa, María del Mar.

References

[1] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling.* Prentice-Hall, 1973.

[2] G. V. Cormack. An LR substring parser for noncorrecting syntax error recovery. *ACM SIGPLAN Notices*, 24(7):161–169, July 1989.

[3] F. DeRemer. *Practical Translators for LR(k) Languages.* PhD thesis, M.I.T., Cambridge, Massachusetts, U.S.A., 1969.

[4] R. W. Floyd. Syntactic analysis and operator precedence. *J. ACM*, 10(3):316–333, 1963.

[5] R. W. Floyd. Bounded context syntactic analysis. *Comm. ACM*, 7(2):62–67, 1964.

[6] J. Fortes Gálvez. Generating LR(1) parsers of small size. In *International Workshop on Compiler Construction*, Paderborn, Germany, 1992. To be published in *Lecture Notes in Computer Science.*

[7] R. M. Graham. Bounded context translation. In *AFIPS Spring Joint Computer Conference*, pages 184–205, 1964.

[8] S. L. Graham and S. P. Rhodes. Practical syntactic error recovery. *Comm. ACM*, 18(11):639–650, 1975.

[9] J. D. Ichbiah and S. P. Morse. A technique for generating almost optimal Floyd-Evans productions for precedence grammars. *Comm. ACM*, 13(8):501–508, 1970.

[10] S. C. Johnson. Yacc—yet another compiler compiler. Computing Science Technical Report 32, AT&T Bell Laboratories, Murray Hill, N.J., 1975.

[11] D. E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, 1965.

[12] W. M. McKeeman, J. J. Horning, and D. B. Wortman. *A Compiler Generator.* Prentice-Hall, 1970.

[13] M. T. Milani. On the descriptive power of simple precedence grammars. *Intern. J. Computer Math.*, 39:29–49, 1991.

[14] M. T. Milani and D. A. Workman. Epsilon weak precedence grammars and languages. *Informatique théorique et Applications/Theoretical Informatics and Applications*, 24(3):241–266, 1990.

[15] C. Pair. Trees, pushdown stores and compilation. *RFTI—Chiffres*, 7(3):199–216, 1964.

[16] H. Richter. Noncorrecting syntax error recovery. *ACM Transactions on Programming Languages and Systems*, 7(3):478–489, July 1985.

[17] S. Sippu and E. Soisalon-Soininen. *Parsing Theory.* Springer-Verlag, 1990.

[18] N. Wirth and H. Weber. EULER: A generalization of ALGOL and its formal definition. Parts I and II. *Comm. ACM*, 9(1):13–23 and 89–99, 1966.