Verificación de Autómatas y Gramáticas

Jairo Rocha
Dept. de Matemàtiques i Informàtica
Universitat de les Illes Balears
jairo@uib.es

Resumen

La verificación de autómatas de estados finitos y de pila, y de gramáticas libres de contexto puede ser materia de enseñanza en una asignatura de Lenguajes Formales usando las metodologías descritas en este artículo. Se intenta mostrar que para los autómatas y gramáticas la verificación es posible en la práctica diaria. Las metodologías se basan en la asociación de *significados* (lenguajes) a los símbolos que aparecen en la solución de un problema (estado, contenido de la pila o variable de una gramática). Los significados corresponden a ciertas propiedades que son invariantes a través de las configuraciones. Los estudiantes de segundo curso de Ingeniería Informática han sido capaces de aplicar las metodologías y aumentar sus elementos de juicio a fin de asociar los significados de los símbolos que aparecen en la solución de un problema.

1. Introducción

Los costes de producción de software son tan altos, en parte, por la falta de corrección de los sistemas que se diseñan e implementan. Los errores son un resultado natural de la alta complejidad de estos sistemas desarrollados y, para intentar reducirlos, se hace necesario especificar rigurosamente los subsistemas y sus relaciones.

Este artículo trata de metodologías que permiten asegurar ls corrección de soluciones a problemas de especificación usando autómatas y lenguajes formales que son fácilmente asimilables por estudiantes de segundo año.

La mayor parte de las soluciones a problemas propuestos en una asignatura de Lenguajes Formales son el resultado de modificaciones que se han producido durante el intento de demostración de soluciones erróneas. Es decir, la solución correcta no se encontró hasta que la demostración exigió completar todos los casos y cubrir aquella combinación de situaciones que no se pensó durante la descripción de la solución.

Por el contrario, en la práctica diaria de un programador no se verifican los programas, por lo que los casos en los que las soluciones propuestas no funcionan aparecen durante las pruebas y experimentos posteriores. Esta situación es comprensible dada la imposibilidad teórica de verificar los programas de manera automática, y la imposibilidad práctica de escribir demostraciones para programas de longitud normal.

El objetivo primordial de este artículo es argumentar que se deben aumentar los elementos de juicio y afinar la intuición de los programadores con el fin de que se produzcan programas más correctos. El programador deberá descubrir en las metodologías de verificación que cada símbolo que aparece en la solución de un problema tiene un significado. Y que los significados se relacionan lógicamente de una manera muy precisa para satisfacer las especificaciones o definiciones de las soluciones.

El razonamiento que se propone para la verificación de autómatas se basa en la identificación de ciertas propiedades que en una configuración (estado actual, contenido de la pila, lenguaje generado por una variable de una gramática, etc.) del autómata son invariantes; es decir, que se puede mostrar que no cambian (siempre son ciertas) cuando el autómata llega a la misma configuración, aunque haya pasado por diferentes pasos de cómputo, tales como cambio de estados, contenido de pilas, de cintas o de variables. Por ejemplo, el estudiante debe saber

que todas las palabras que en un autómata de estados finitos determinista llegan a un estado fijo satisfacen una propiedad que no satisfacen palabras que llegan a otro estado; y que hay propiedades que relacionan una configuración de la pila con la palabra leída, o una variable de una gramática con sus palabras generadas.

Nuestro objetivo quedará cumplido si el uso de las técnicas descritas trae como resultado el hábito duradero (y casi inconsciente) de escribir soluciones, y programas en particular, respetando estrictamente las relaciones entre los significados intuitivos de los símbolos usados, sean estos estados, nombres de variables en una gramática, vectores, nombres de variables y posiciones dentro de un programa, nombres de procedimientos, de clases, etc.

Nuestra experiencia docente, y en particular con la asignatura de Autómatas y Lenguajes Formales, nos ha convencido de que la gran mayoría de los detalles técnicos de los métodos, definiciones o teoremas se olvida con facilidad y prontitud. Por lo tanto, si los estudiantes se concentran en estos detalles, ellos y los docentes verán sus esfuerzos difuminarse rápidamente.

El énfasis de la asignatura debería estar en la justificación. Se deben dar herramientas para responder a la pregunta de por qué cierto autómata reconoce un lenguaje previamente definido formalmente, con la intención de que el estudiante cree el hábito de preguntarse sobre la fiabilidad absoluta de las transiciones, reglas o instrucciones que decide incluir en una solución. En resumen, el contenido de la asignatura de Lenguajes Formales debería ayudar a mejorar al alumno sus habilidades de justificación en el área de la computación.

Esta actitud se contrapone al aparente énfasis que algunos estudiantes dan a las recetas relacionadas con estos temas; es fácil que un estudiante simplifique el temario de un curso de teoría de lenguajes formales y autómatas a recetas para, por ejemplo, encontrar un autómata de estados finitos determinista equivalente a uno no determinista, minimizar autómatas de estados finitos, encontrar la forma normal de Chomsky de una gramática, o escribir máquinas de Turing que solucionen problemas concretos simples o no. La importancia de evitar que una asignatura cualquiera se resuma al dominio de recetas no sobra que sea enfatizada. El estudiante fácilmente olvida las recetas y, por lo tanto, el resultado final de producir un cambio de actitud en el estudiante y enseñar conceptos importantes y duraderos no se alcanza.

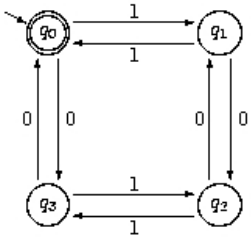
Así, el estudiante debería conocer los métodos completamente automáticos que permiten cambiar una solución de un simbolismo a otro, pero el objetivo no es que el estudiante aprenda esos métodos a la perfección, sino que sepa de su existencia y utilidad para generar soluciones fiables. En particular, el estudiante no se debería evaluar sobre el desarrollo de una máquina de Turing para un lenguaje dado, cuando en la práctica nadie usa máquinas de Turing y usando otros formalismos obtendría soluciones más fáciles; las máquinas de Turing se enseñan con el objetivo de definir formalmente un programa de manera simple y estudiar los límites últimos de las máquinas universales, más comúnmente llamadas ordenadores. Igualmente, el estudiante se debería evaluar sobre la verificación de que un autómata de pila es correcto respecto a un lenguaje dado, y no pidiéndole que escriba el autómata asociado a una gramática, problema que puede ser solucionado mejor por un ordenador.

En este artículo se describen temas que no se tratan, o sólo de manera tangencial, en los textos usuales (por ejemplo, [1, 2]): verificación de autómatas finitos, de gramáticas y de pila. La verificación de máquinas de Turing no se cubre dado que existen textos que incluyen metodologías de demostración de lenguajes de programación generales (por ejemplo, las *wp* en [4]).

Sugerimos dar prioridad a la verificación sobre temas más tradicionales como minimización de autómatas, los lemas de bombeo y las formas normales de gramáticas porque la verificación ayuda más a crear el hábito de justificar que lo que ayuda el conocimiento de aquellos temas más tradicionales.

2. Autómatas de Estados Finitos

Explicaremos la metodología para cada tipo de verificación con un ejemplo. En este caso,



supongamos que el problema es verificar que el siguiente autómata A reconoce el lenguaje

$L=\{w \in (0+1)^* \mid w$ tiene par de 0's y de 1's $\}$.

2.1. Ecuaciones

El primer paso es definir las ecuaciones asociadas al autómata en términos de los lenguajes asociados a cada estado:

$$l_i = \{w \mid (q_0,w) \Rightarrow^* (q_i,w)\}$$

Es decir, l_i es el conjunto de palabras que en cero o más pasos del autómata llegan a q_i partiendo del estado inicial. Las ecuaciones son:

$$l_o = l_1 1 + l_3 0 + \lambda$$
$$l_1 = l_0 1 + l_2 0$$
$$l_2 = l_1 0 + l_3 1$$
$$l_3 = l_0 0 + l_2 1$$
$$L(A) = l_0$$

en las que el lenguaje del autómata es el asociado a q_0 porque es el único estado final. Notese que las ecuaciones por sí solas describen totalmente el autómata.

2.2. Significados

El siguiente paso es el más delicado de la metodología. Consiste en definir un lenguaje asociado a cada estado en términos similares a los del lenguaje L. En este caso, se hace necesario adivinar el conjunto de palabras asociadas a cada estado en términos de la paridad de los símbolos, independientemente del autómata. Este paso es el menos mecánico de todos y se realiza normalmente por prueba y error. El lector debe descubrir los siguientes lenguajes:

$L_0 = \{w \mid w$ tiene par de 0´s y 1´s$\}$
$L_1 = \{w \mid w$ tiene par de 0´s e impar de 1´s$\}$
$L_2 = \{w \mid w$ tiene impar de 0´s y 1´s$\}$
$L_3 = \{w \mid w$ tiene impar de 0´s y par de 1´s$\}$.

2.3. Partición

Si el autómata es determinista se hace necesario mostrar que los lenguajes del paso anterior forman una partición de Σ^*. En la mayoría de los autómatas, este paso se hace al mismo tiempo que se ensayan varias definiciones de significados pues es una manera de saber que estas definiciones van por buen camino. En este ejemplo, es obvio que estos lenguajes forman una partición.

2.4. Una Solución de las Ecuaciones

Si en las ecuaciones de arriba se reemplaza l_i por L_i, se debe comprobar que las ecuaciones siguen siendo ciertas. Si este es el caso, entonces por unicidad de la solución del sistema, debe ser cierto que $l_i = L_i$. La unicidad es una consecuencia de la aplicación iterativa del Lema de Arden [1].

Si el autómata es determinista es suficiente mostrar que las ecuaciones valen en la dirección de derecha a izquierda, pues las otras direcciones valen automáticamente si los significados forman una partición [3]. Si el autómata no es determinista se hace necesario mostrar cada ecuación en ambas direcciones.

En este ejemplo, hay que demostrar cada una de las siguientes afirmaciones:

$L_o \supseteq L_1 1 + L_3 0 + \lambda$
$L_1 \supseteq L_0 1 + L_2 0$
$L_2 \supseteq L_1 0 + L_3 1$
$L_3 \supseteq L_0 0 + L_2 1$

Todas estas comprobaciones hacen la verificación larga, pero en la mayoría de las veces, cada comprobación es simple, como en este caso.

2.5. El Paso Final

Debido a que $l_i = L_i$, para cada estado q_i, y, suponiendo que los estados finales son $q_{f1},..., q_{fk}$,

$$L(A) = l_{f1} \cup ... \cup l_{fk} = L_{f1} \cup ... \cup L_{fk},$$

por lo que la verificación termina mostrando que

$$L_{f1} \cup ... \cup L_{fk} = L.$$

En este ejemplo basta mostrar que $L_0 = L$, lo que es obvio.

3. Gramáticas Libres de Contexto

Nos interesa demostrar que el lenguaje generado por la gramática G

$I \rightarrow \lambda \mid aB \mid bA$
$A \rightarrow aI \mid bAA$
$B \rightarrow bI \mid aBB$

es
$L = \{ w \in (a+b)^* \mid w$ tiene el mismo número de a'es que de b'es$\}$.

La metodología es muy similar a la de la sección anterior, y como antes, lo que la hace simple es un teorema de unicidad de las soluciones de un sistema de ecuaciones.

3.1. Ecuaciones

El primer paso es definir las ecuaciones asociadas a la gramática:

$X_I = \lambda + aX_B + bX_A$
$X_A = aX_I + bX_AX_A$
$X_B = bX_I + aX_BX_B$

en las que X_I, X_A y X_B son variables de lenguajes. Los lenguages generados por las variables,

$Gen(I) = \{w \mid I \Rightarrow^* w\}$
$Gen(A) = \{w \mid A \Rightarrow^* w\}$
$Gen(B) = \{w \mid B \Rightarrow^* w\}$

son solución de las ecuaciones. Las condiciones de unicidad de la solución son ciertas para estas producciones porque las partes derechas de las producciones son constantes o tienen un símbolo terminal[3].

3.2. Significados

El siguiente paso es nuevamente el más difícil de la metodología. Consiste en definir un lenguaje asociado a cada variable de la gramática en términos similares a los del lenguaje L.

Después de varias derivaciones, debe estar claro que lo más es seguro es que los lenguajes generados por las variables sean

$L_I = \{ w \mid |w|_a = |w|_b \}$
$L_A = \{ w \mid |w|_a = |w|_b + 1 \}$
$L_B = \{ w \mid |w|_b = |w|_a + 1 \}$.

donde $|w|_\sigma$ representa el número de veces que σ aparece en w.

Nos interesa ver que L_I, L_A y L_B son también solución de las ecuaciones, por lo que, por unicidad $L_I = Gen(I)$, $L_A = Gen(A)$ y $L_B = Gen(B)$.

3.3. Una Solución de las Ecuaciones

Basta demostrar que (L_I, L_A, L_B) satisfacen cada ecuación. La demostración de que cada ecuación se satisface necesita dos direcciones (no hay atajos como en la sección anterior). La demostración de cada uno de los casos en cada dirección normalmente es simple, aunque todo junto resulta ser largo.

En este ejemplo concreto, la demostración de

$L_A \subset aL_I + bL_AL_A$

requiere demostrar que si una palabra de L_A comienza por b, el resto de la palabra se puede dividir en dos subpalabras, cada una con una a más; para esto se hace necesario mostrar cómo dividir efectivamente la palabra, contando de izquierda a derecha la diferencia entre el número de a'es y el de b'es.

3.4. El Paso Final

Debido a que $L_X = Gen(X)$ para cada variable X de la gramática, $L(G) = Gen(S)$, donde S es la variable inicial. Entonces, la verificación termina mostrando que

$$L_S = L.$$

En este ejemplo, basta mostrar que $L_I = L$, lo que es inmediato.

4. Autómatas con Pila

La verificación de autómatas de pila deterministas se realiza por inducción en el tamaño de la palabra procesada. Esta estrategia implica una complicación simbólica que se había evitado en las dos metodologías anteriores usando teoremas de unicidad sobre un sistema de ecuaciones. Pero para este caso no hay un teorema de unicidad que conozcamos, aunque sí hay lenguajes asociados a configuraciones (estado,pila) del autómata.

4.1. Lenguaje Asociado a una Configuración de la Pila

Como en las metodologías anteriores, la definición de ciertos lenguajes intermedios asociados a significados de símbolos (estados, variables, configuraciones, etc.) es decisiva para conseguir el objetivo.

Consideremos el siguiente ejemplo. Sea

$$L = \{ w \mid w \in (a+b)^* \text{ y } |w|_a = |w|_b \}.$$

El autómata de pila M que reconoce $L\#$ por pila vacía es:

$\delta(q,a,Z_0) = (q, aZ_0)$ $\quad$ $\delta(q,b,Z_0) = (q, bZ_0)$
$\delta(q,a,a) = (q, aa)$ $\quad$ $\delta(q,b,b) = (q, bb)$
$\delta(q,a,b) = (q,\lambda)$ $\quad$ $\delta(q,b,a) = (q, \lambda)$
$\delta(q,\#,Z_0) = (q, \lambda)$

Los lenguajes intermedios (o significados) corresponden a conjuntos de palabras de entrada con las cuales se llega a un estado y a un contenido de la pila dados. Dado un autómata de pila $(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$, se define el siguiente lenguaje para cada $q \in Q$ y $\gamma \in \Gamma$:

$$L(q,\gamma) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \Rightarrow^* (q_0, \lambda, \gamma) \}.$$

En nuestro ejemplo, $L(M) = L(q, \lambda)$.

Demostraremos las siguientes igualdades:

$$L(q,a^kZ_0) = \{w \mid |w|_a = |w|_b + k \},$$
$$L(q,b^kZ_0) = \{w \mid |w|_b = |w|_a + k \}.$$

La pregunta en este instante debería ser: ¿de dónde se han obtenido estas ecuaciones? La respuesta es: corresponden a la formalización de las ideas intuitivas de los significados de los estados y del contenido de la pila que se dieron antes de definir el autómata. Por ejemplo, la primera ecuación dice que para llegar al estado q con k $a$es encima de Z_0, se debe haber leído una palabra que tiene k $a$es más que $b$es.

4.2. Inducción

La demostración de los significados de los lenguajes se hace por inducción simultáneamente sobre las dos igualdades. Se deben hacer simultáneamente porque se puede pasar de la configuración de una ecuación a la otra leyendo un símbolo, es decir, las ecuaciones se relacionan entre sí, como es lo normal en autómatas y gramáticas.

Las ecuaciones corresponden a demostrar por inducción en $|w|$ que para todo k:

$$(q,w,Z_0) \Rightarrow^* (q, \lambda, a^k Z_0) \Leftrightarrow |w|_a = |w|_b + k \quad \text{y}$$
$$(q,w,Z_0) \Rightarrow^* (q, \lambda, b^k Z_0) \Leftrightarrow |w|_b = |w|_a + k.$$

Cuando se considera $w\sigma$ en el paso de inducción, se calculan todas las posibles configuraciones (q, σ, γ) antes de que la entrada quede vacía. Los diferentes casos para q, σ y dependen de las transiciones del autómata y pueden ser varios, lo que hace la demostración larga. Cada caso requiere usar la hipótesis de inducción sobre la variable k apropiada, lo que no es difícil. Sin embargo, los estudiantes deben hacer varias demostraciones antes de aprender a hacer un manejo simbólico correcto.

4.3. El Paso Final

Para terminar veamos que $L(M) = L$. Sabiendo que $L(M) = L(q, \lambda)$, y observando que la única forma de vaciar la pila es eliminando el símbolo Z_0 al leer #, se tiene que

$(q, w\#, Z_0) \Rightarrow^* (q, \#, Z_0) \Rightarrow (q, \lambda, \lambda)$
ssi $(q, w, Z_0) \Rightarrow^* (q, \lambda, Z_0)$
ssi (según lo demostrado para $a^0 Z_0$)
$|w|_a = |w|_b + 0$
ssi $w \in L$,

lo que muestra que el autómata es correcto.

4.4. Experiencia Docente

Los estudiantes de segundo año aprenden que un ejercicio que requiera dar un autómata o gramática no estará completo hasta que no se dé la verificación correspondiente. Es obvio que la mayoría de los estudiantes preferiría dar soluciones a los ejercicios sin dar las verificaciones como se hace en los libros de lenguajes formales tradicionales y en la mayoría de las asignaturas de programación. Esto hace aún más importante que en al menos esta asignatura se mantenga como objetivo la necesidad de justificar cada parte de una solución presentada. Sin embargo, no disponemos de ningún elemento objetivo que permita medir si los estudiantes saben dar, a largo plazo, mejores justificaciones.

Las metodologías son perfectamente accequibles a los estudiantes. Es decir, más de un 50% de los estudiantes son capaces de dar verificaciones correctas en los exámenes. Solamente la verificación de automatas de pila requiere varias semanas hasta que los estudiantes reemplazan los símbolos correctamente en los pasos de las demostraciones por inducción de cada uno de los casos. Esta estrategia es sólamente un poco más compleja que los casos que se consideran en el uso de los lemas de bombeo, sólo que los resultados son más prácticos: la garantía de la corrección de las soluciones.

Referencias

[1] Rafel Casas, Lluís Màrquez. *Llenguatges, Gramàtiques i Autòmats: Curs Bàsic.* Universitat Politècnica de Catalunya, 1997.

[2] Dean Kelley. Teoría de Autómatas y Lenguajes Formales. Prentice Hall, 1995.

[3] Francesc Roselló, Jairo Rocha. *Autòmats i Llenguatges: Verificació, Implementació i Concurrència.* Materials Didàctics. Universitat de les Illes Balears, 2003 (En imprenta).

[4] Robert Sebesta.. *Concepts of Programming Languages*, Addison Wesley, 2002.