SIMPLIFICACIÓN DE EXPRESIONES REGULARES

Regular expression simplification

Lucio Torrico
luciotorrico@gmail.com

RESUMEN

Hay diferentes enfoques intentando simplificar expresiones regulares (e.r.)
Basados en el trabajo inicial de Alley Stoughton, "*Formal Language Theory. Integrating Experimentation and Proof*" (Stoughton, 2012), hemos efectuado observaciones y algunos aportes a su perspectiva de cómo hallar e.r. menos complejas utilizando reglas de reescritura.
El algoritmo denominado fuerte puede ser mejorado priorizando la aplicación de las reglas de simplificación (incluso cambiando algunas de ellas) y principalmente redefiniendo la noción de complejidad de una e.r. Se plantea además recurrir a autómatas para testear la subsunción.

Palabras clave:

Expresiones regulares; complejidad de expresiones regulares; simplificación de expresiones regulares

ABSTRACT

Different approaches trying to simplify regular expressions (r.e.) exist.
Based on the initial work of Alley Stoughton, "Formal Language Theory. Integrating Experimentation and Proof " (Stoughton, 2012), we have made comments and some contributions to their perspective of how to find r.e. less complex using rewrite rules.
Strong algorithm can be improved prioritizing the implementation of simplification rules (even changing some of them) and mainly redefining the notion of complexity of a r.e. It is further suggested use automata to test subsumption.

keywords:

Regular expression; regular expresión complexity; regular expression simplification

INTRODUCCIÓN

Existen diferentes maneras (teóricas y en software) para simplificar una e.r.
Se describen brevemente algunas y luego se opta por la perspectiva de Stoughton. De ella se indica el algoritmo más fuerte, así como una revisión somera de conceptos anexos imprescindibles para su funcionamiento.

Se plantea una discusión, los aportes efectuados, y el énfasis que debe dársele a la noción de complejidad y su cálculo. Se termina proponiendo un criterio adicional al que se sugiere dar mayor importancia.

MÉTODOS

Se trata de una teoría construida como un sistema lógico (definiciones, operaciones). Se han derivado y probado enunciados (teoremas, etc.), mismos que deben ser sometidos a crítica y nuevas revisiones. Se ha apelado a esquemas usuales de demostración (inducción, reducción al absurdo). Se ha hecho uso del análisis y la deducción, la ejemplificación de resultados (que clarifica los mismos, encuentra fisuras o nuevas ideas). Se ha entablado un fluido diálogo vía e-mail. Y el software subyacente y que es producto de la teoría adoptada se ha hecho con una metodología de programación funcional y un lenguaje de programación ML.

A MODO DE REVISIÓN Y CONCEPTUALIZACIÓN

Presumimos que el lector está familiarizado con los conceptos y nociones de la teoría de lenguajes formales, gramáticas, autómatas y e.r. en el sentido de (Martin, 2003) (Torrico, 2005), que siguen convenciones de notación muy similares. Especialmente (Stoughton, 2012).

Una variante muy común de obtención de expresiones regulares (e.r.) simplificadas, parte de un autómata finito y elimina estados obteniendo lo que se denomina grafos de transición generalizados, hasta obtener un grafo con sólo dos estados y la e.r. que denota al lenguaje aceptado por el autómata original (Linz, 2006).

Hay interesantes enfoques sobre cómo hacerlo y en particular cuál estado elegir para su eliminación (Morais y Delgado, 2004) (Gruber y Holzer, 2008) (Martin, 2003) (Torrico, 2005).

Otra alternativa igualmente popular es la utilización del teorema de Kleene o de un sistema de ecuaciones junto al lema de Arden (Martin, 2003) (Torrico, 2005).

Dada una e.r podemos reutilizar estas aproximaciones obteniendo primero el autómata finito (por ej. a través del algoritmo de Thompson (Torrico, 2005)), luego un autómata equivalente de estados mínimos y a partir de él una nueva e.r. (Martin, 2003) (Torrico, 2005)

El software revisado, principalmente CALCHALERO (Tamagnini, 2005) y JFLAP (Rodger, 2006), aunque no enfatiza la parte de simplificación, tiende a obtener las e.r. más simples que puede a partir de la teoría indicada.

Existen, sin embargo, alternativas que toman como entrada una e.r. y no utilizan autómatas para obtener otra menos compleja, haciendo uso más bien de avances en el cómputo simbólico y de reglas de reescritura (Trejo y Fernandez, 1996) (Mitchell, 2005).

Hay reglas de reescritura conocidas bajo el nombre de equivalencias y son muy populares, aunque normalmente se las presenta sin un procedimiento de uso (Martin, 2003) (Torrico, 2005).

El caso de Stoughton es paradigmático no sólo porque "ofrece parcialmente nóveles algoritmos y provee algunas reglas de

simplificación y conceptos no conocidos como el de complejidad en clausura" (Stoughton, 2012), sino porque acompaña la teoría con el software Forlan de rendimiento nada despreciable.

El algoritmo más fuerte

Dada una e.r. α se obtiene un conjunto X de e.r. β, fruto de la aplicación un número arbitrario de veces (a toda la e.r. α o a una subparte de ella): De la simplificación débil, las reglas estructurales, las reglas de reducción o las reglas de distribución. Si X *es vacío se devuelve* α.

Si X *no es vacío el algoritmo se llama a sí mismo recursivamente con el elemento menos complejo de* X. (Stoughton, 2012)

La implementación en Forlan de este algoritmo puede demorar en exceso, de manera que se acepta un parámetro extra *n*, limitando la cardinalidad de X en cada llamada. Ello lo hace más eficiente en términos de tiempo, pero puede no dar con la e.r. más simple posible.

Reglas de distribución, reglas estructurales, reglas de reducción, simplificación débil y subsunción

Las reglas de distribución son obvias:
$\alpha(\beta_1 + \beta_2) \rightarrow \alpha\beta_1 + \alpha\beta_2$
$(\alpha_1 + \alpha_2)\beta \rightarrow \alpha_1\beta + \alpha_2\beta$

Las 9 reglas estructurales son del tipo:
$(\alpha + \beta) + \gamma \rightarrow \alpha + (\beta + \gamma)$
$\alpha^{*}\alpha \rightarrow \alpha\alpha^{*}$

De las 26 reglas de reducción, algunas interesantes son:
$(\alpha^{*} + \beta)^{*} \rightarrow (\alpha + \beta)^{*}$
$\alpha^{*}(\% + \beta(\alpha + \beta)^{*}) \rightarrow (\alpha + \beta)^{*}$
If hasEmp α and hasEmp β, then $(\alpha\beta)^{*} \rightarrow (\alpha + \beta)^{*}$
If sub($\alpha\alpha^{*}$, β), then $\alpha^{*} + \beta \rightarrow \lambda + \beta$

La simplificación débil permite obtener e.r. sin subexpresiones tales como $\theta + \beta$, $(\beta^{*})^{*}$, ó $\lambda\beta$, donde los excesos son evidentes.

El algoritmo para obtener e.r. débilmente simplificadas es recurrente y básicamente desplaza la estrella de Kleene lo más a la derecha posible en cada subexpresión y elimina los excesos mencionados arriba.

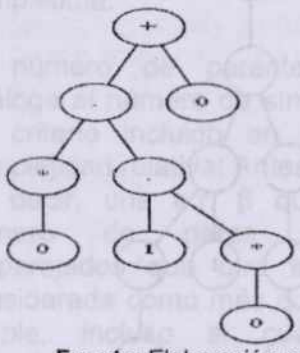
Nótese que la aplicación de varias reglas de reducción es condicional y requiere: Determinar si el lenguaje asociado a una e.r. incluye la cadena vacía λ, o bien que calculemos cuándo el lenguaje asociado a una (sub)e.r. dada está subsumido en el lenguaje asociado a otra.

En ambos casos (junto a la determinación de si el lenguaje asociado a una e.r. incluye al símbolo *a*) las soluciones son funciones/algoritmos recursivos bastante intuitivos, denominados respectivamente hasEmp, hasSym y obviousSubset.

Complejidad de una e.r.

Se han sugerido muchas medidas de complejidad respecto de e.r. (el peso-estrella y el número de símbolos alfabéticos son dos de ellas) (Ehrenfeucht y Zeiger, 1976) (Ellul, Krwetz, Shallit y Wang, 2005).

Aquí, las e.r. son árboles.
Por ejemplo, en la Figura 1, para la e.r. $\alpha = \mathbf{0^{*}10^{*}+0}$:



Fuente: Elaboración propia

Figura 1. La e.r. $\alpha = \mathbf{0^{*}10^{*}+0}$

El tamaño (size, número de nodos) de α es 9.
La altura de α es: 5 (si consideramos el número de nodos del camino más largo), ó 4 (si consideramos el números de arcos).

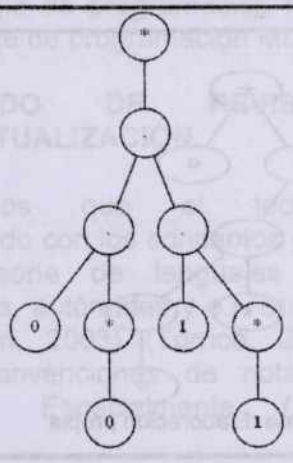
A partir de: $\lambda < \theta < a < * < \cdot < +$
Se define un orden total para las e.r. el cual se se calcula así:
Primero la raiz, y luego recursivamente sus hijos de derecha a izquierda.
· y + son asociativos por derecha.

La precedencia y asociatividad de los operadores es de mayor a menor: * , · , +.
El alfabeto de una e.r. es el conjunto de símbolos que aparecen en ella.

El número de símbolos de una e.r. es el número de ocurrencias de símbolos (un símbolo puede ocurrir más de una vez). El número de símbolos de a+aa es 3.

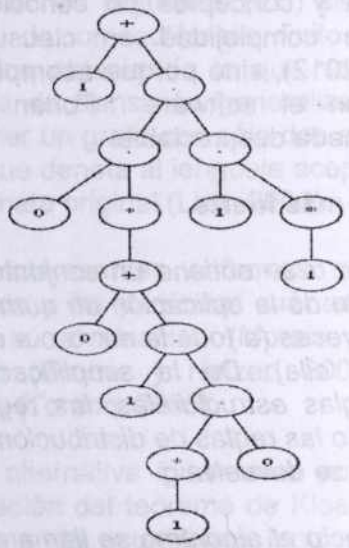
Complejidad en clausura: A partir del árbol de la e.r. se obtiene un lista ordenada (llamada ns) donde cada nodo(celda) de la lista representa una rama del árbol (desde la raiz a una hoja) y contiene el número de estrellas (de Kleene) en dicha rama.

Por ejemplo (véanse las Figuras 2 y 3):
α = **((00*)(11*))*** ns α=[2,2,1,1]
β = **1+(0(0+11*0)*)(11*)** ns β=[2,1,1,1,1,0,0,0]



Fuente: Elaboración propia

Figura 2. La e.r. α = **((00*)(11*))***



Fuente: Elaboración propia

Figura 3. La e.r. β = **1+(0(0+11*0)*)(11*)**

Una e.r. es menos compleja (complejidad en clausura) que otra si su lista es menor (celda a celda) (ó si son idénticas celda a celda pero tiene menos nodos). En el ejemplo: cc β $<_{cc}$ cc α.

Una e.r. se dice que está estandarizada: Si no hay paréntesis innecesarios [$(\beta_1+\beta_2)+\beta$, $(\beta_1\cdot\beta_2)\cdot\beta$]; si no ocurre que las estrellas de Kleene están lo más a la izquierda posible [$\beta^*\beta$, $\beta^*(\beta\gamma)$, $(\beta_1\beta_2)^*\beta_1$, $(\beta_1\beta_2)^*\beta_1\lambda$]; si no ocurre que las e.r. aparecen en desorden [$\beta_1+\beta_2$ con $\beta_1>\beta_2$, $\beta_1+(\beta_2+\beta_3)$ con $\beta_1>\beta_2$].

Para juzgar la complejidad relativa de las e.r. α y β (si una es más simple -menos compleja que otra-), se calcula en orden:
- La complejidad en clausura. En caso de que sean iguales:
- El tamaño. Si son iguales:
- El número de concatenaciones. Si son iguales:
- El número de símbolos. Si son iguales:
- Si están estandarizadas o no.

Se denota dicha relación así:

$$\alpha \leq_{simp} \beta.$$

RESULTADOS

Sin contar el apoyo para el proof-reading, el autor de este artículo ha sido parte de un excitante intercambio de puntos de vista por e-mail con Stoughton y ha sugerido cambios, como en la regla 15, hasta llegar a su versión actual:

If not(hasEmp β) and $(cc\,\alpha \cup cc\,\beta) <_{cc} cc\,\overline{\alpha}$, then

$$(\alpha^{\cdot}\beta)^{\cdot} \rightarrow \lambda + (\alpha + \beta)^{\cdot}\beta$$

donde cc denota la complejidad en clausura de α y el sobrerrayado denota el sucesor

Que si bien aumenta el tamaño, disminuye el peso-estrella.

Otro grupo de resultados se entienden mejor luego de la discusión, en lo que llamamos variante.

DISCUSIÓN

Stoughton ha proporcionado una base sólida para una simplificación de e.r. procedimental que no apele a autómatas.
La perspectiva merece algunos comentarios:

Algunas de las reglas originales de reducción privilegiaban la reducción del tamaño de la e.r. (*size*) incluso haciendo e.r. con un peso-estrella (*star-height*) elevado.

La función/algoritmo que determina subsunción adolece de incompletitud como la misma autora juzga.

Si bien la definición y la determinación de complejidad son sólidas, no supera algunas pruebas.

En particular, un sondeo realizado muestra que la e.r. α = a(a(a(λ+a))) es considerada por muchos entrevistados como más compleja que la e.r. β = aaa+aaaa. Es decir, el número de paréntesis aporta complejidad.

VARIANTE

Aunque requiera que apelemos a autómatas, el testeo de subsunción puede hacerse completo y, en la práctica, en tiempo aceptable.

En efecto, se sabe que $L_1 \subseteq L_2$ si y sólo si $L_1 \cap L^c_2 = \{\}$ (la ^{c} indica complemento)
Para calcular si el lenguaje asociado a una e.r. está subsumido en el lenguaje asociado a otra e.r. basta construir los autómatas determinísticos mínimos (AFDM) A_1 y A_2 para ambas expresiones, el AFDM para el complemento de A_2, llamémosle A_c, y el AFD para la intersección de A_1 y A_c. Y testear si dicho AFD para la intersección acepta sólo el conjunto vacío o no.

Existe abundante teoría para todas estas construcciones y de hecho Forlan permite hacerlo. Por ejemplo, el caso de testear si un AFD acepta sólo el conjunto vacío o no, es un conocido problema decidible: Podemos construir el conjunto de estados alcanzables desde el inicial, y si alguno de ellos es final, entonces A no acepta el conjunto vacío, en otro caso sí. O bien seguir la alternativa de Davis: $L(A) \neq \{\}$ si y sólo si $\exists w\ d^*(q1,w) \in F$ con $|w|<n$ (donde n es el número de estados de A).
Disminuye la belleza de hacerlo todo con e.r. (sin autómatas), pero aporta completitud.

El número de paréntesis (de forma análoga al número de símbolos) debe ser un criterio incluido en la jerarquía de complejidad relativa: Antes del tamaño.
Es decir, una e.r. β que tenga mayor número de pares de paréntesis emparejados que otra e.r. α debe ser considerada como más compleja ó menos simple, incluso si cuenta con más concatenaciones o es de mayor tamaño.
Es más, el caso β = aa+aaa, α = (λ+a)aa exige un replanteamiento aún mayor de la jerarquía, pues en este caso:
tamaño α = 7 < 9 = tamaño β

cc α = [0,0,0,0] $<_{cc}$ [0,0,0,0,0] = cc β
numConcats α = 2 < 3 = numConcats β
Pero β esmássimpleque α
es compatible con la idea que tienen los entrevistados y no parece un exceso ni un caso excepcional.

Es claro que pensamos en e.r. sin paréntesis innecesarios (como en la estandarización).
Queda en discusión si este número de paréntesis emparejados debe estar incluso antes de la complejidad en clausura, como parece indicar el ejemplo citado, o luego.

También parece razonable (mientras no podamos agotar todas las selecciones posibles en el conjunto X y en su obtención), plantear la posibilidad de priorizar el uso de las reglas de reescritura.

Las reglas de distribución no debieran ser una opción sino una primera elección, debido a que reducen el número de paréntesis aún a pesar de incrementar el número de concatenaciones (obviamente en cálculos intermedios esto puede relativizarse).

Ello obliga a pensar en la posibilidad de privilegiar el uso de algunas reglas (incluso diferentes a las de distribución) antes que otras.

Por supuesto estos cambios no se han introducido en Forlan, dado que la autora de dicho software es quien debe decidirlo.

CONCLUSIONES

Se han revisado los procedimientos de simplificación existentes, tanto a nivel teórico como a nivel de software gratuito accesible.

Aunque una alternativa posible es hacer uno relativamente nóvel en base a ellos, hemos decidido adoptar el punto de vista de Stoughton a través de reglas de reescritura y una enriquecida conceptualización, trabajar con ella y aportar (críticamente) a su visión del proceso de simplificación.

Participando en la redefinición de complejidad relativa, considerando más simples, e.r. con menor peso-estrella (incluso con mayor tamaño).

Modificando algunas reglas de reducción, que respeten la complejidad redefinida.
Promoviendo un testeo de subsunción completo, aún al costo de introducir autómatas en los cálculos intermedios.
Planteando una nueva redefinición de la complejidad relativa, haciendo más deseables (simples) e.r. con menor número de paréntesis.

Y sugiriendo reflexionar sobre la ventaja de privilegiar el uso de algunas reglas antes que otras.

Finalmente, mencionar que la inclusión de nuevas reglas de reducción (o afinamiento de las existentes) y el mejoramiento del cálculo de la subsunción sin autómatas son sólo dos de los tópicos que aún están abiertos, y que esperan aportes creativos.

BIBLIOGRAFÍA

Ehrenfeucht, A. y Zeiger, P. (1976). *Complexity measures for regular expressions.* Journal of Computer and System Sciences, 12, pp. 134-146

Ellul, K., Krawetz B., Shallit J., y Wang M. (2005) *Regular expressions: New results and open problems.* Journal of Automata, Languages and Combinatorics, 10(4), pp. 407-437

Gruber H., y Holzer M. (2008). *Provably shorter regular expressions from deterministic finite autómata.* In DLT 2008, LNCS, vol. 5257, pp. 383-395. Springer

Linz, P. (2006). *An introduction to Formal Languages and Automata.* (4ª ed.) Londres: Jones & Bartlett

Martin, J. (2003). *Introduction to Languages and the Theory of Computation.* (3ª ed.) Nueva York: McGraw-Hill

Mitchel, N. (2005). *Regular expression simplification.* [Homepage]. Consultado: [26, Enero, 2011] Disponible en: http://community.haskell.org/~ndm/resimplify

Morais, J., y Delgado, M. (2004). *Approximation to the Smallest Regular Expression for a Given Regular Language.* In CIAA 2004, LNCS, vol. 3317, pp. 312–314. Springer.

Rodger, S. (2006). *JFLAP.* [Software en línea]. Consultado: [15, Enero, 2012] Disponible en: http://jflap.org

Stoughton, A. (2012). *Formal Language Theory. Integrating Experimentation and Proof.* [Libro en línea]. Consultado: [15, Agosto, 2012] Disponible en: http://alleystoughton.com/forlan/book

Tamagnini, J. *Chalchalero.* (2005). [Software en línea]. Consultado: [26, Enero, 2011] Disponible en: http://www.ucse.edu.ar/fma/sepa

Torrico, L. (2005). *Lenguajes Formales.* La Paz: Universidad Mayor de San Andrés. [Libro en línea]. Consultado: [15, Agosto, 2012] Disponible en: https://sites.google.com/site/luciotorrico/libros-escritos/LenguajesFormales.pdf

Trejo, A., y Fernandez, G. (1996). *Regular expressions simplification.* The 2nd IMACS Conference on Applications of Computer Algebra, Hagenberg, Austria