F. Drewes 2005-04-05

Datavetenskapens grunder, VT06
– Gruppövning 2 –

1. Beskriv på vanlig svenska (men ändå hyfsat noggrant) vilka strängar som ingår i $L(\alpha)$ där $\alpha = \diamond(ab \cup aba)^*\diamond$. Rita sedan en (deterministisk eller icke deterministisk) ändlig automat för språket. Lyckas ni förenkla första utkastet? Om det var icke deterministisk (som jag gissar), hittar ni en ekvivalent deterministisk automat?

2. En grammatik som beskriver enkla aritmetiska uttryck är $G = (V, \Sigma, R, S)$ där $V = \{S, P, X\}$, $\Sigma = \{+, -, *, /, (,), x, y, z\}$ och

$$R = \{ \begin{array}{lcl} S & \rightarrow & S + P \mid S - P \mid P, \\ P & \rightarrow & P * X \mid P / X \mid X, \\ X & \rightarrow & (S) \mid x \mid y \mid z \ \}. \end{array}$$

   Rita deriveringsträd för $y * x + y - z * z$ och $x + x * (y + z - x)$.

   Är grammatiken entydig? Diskutera hur träden återspeglar de vanliga reglerna för att tolka aritmetiska uttryck:

   - $*$ och / binder starkare än + och −,
   - $*$ och / resp. + och − associerar till vänster

3. Kom ihåg pumpinglemmat för reguljära språk:

   > För varje reguljärt språk A finns det något $p \in \mathbb{N}$ sådant att varje sträng $u \in A$ med $|u| \geq p$ kan delas upp i tre delsträngar $u = xyz$ som uppfyller följande krav:
   > (a) $xy^iz \in A$ för alla $i \geq 0$,
   > (b) $|y| > 0$,
   > (c) $|xy| \leq p$.

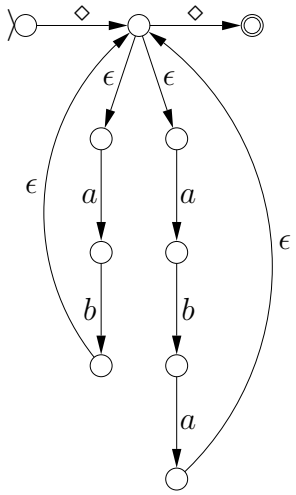
   Diskutera hur lemmat kan användas för att visa att ett språk inte är reguljärt, med språket $A = \{ab^nab^n \mid n \in \mathbb{N}\}$ som exempel. En argumentation som innehåller de flesta vanliga felen lyder ungefär så här:

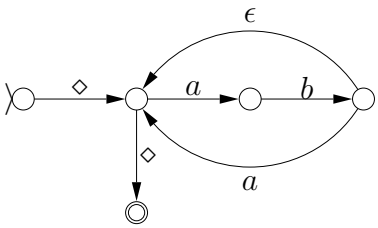
   > Välj $p = 4$ och betrakta $u = abbabb$. Enligt definitionen av A är $u \in A$. Dessutom gäller att $|u| \geq p$ så att pumpinglemmat kan användas. Låt $x = a$, $y = bb$ och $z = abb$. Då är $u = xyz$, $|y| > 0$, och $|xy| \leq p$. Däremot gäller att $xyyz = abbbbabb \notin A$. Första villkoret i lemmat gäller alltså inte för A som således inte kan vara reguljärt.

   Vilka misstag gjorde jag och hur kan de rättas till?

Kommentarer...

1. Språket består av alla strängar på formen $\diamond u \diamond$ där u kan delas upp i ett godtyckligt antal strängar $u_1, \ldots, u_n \in \{ab, aba\}$. Två automater som accepterar språket (den vänstra ganska "mekaniskt" konstruerad, den andra med ett litet tillskott på mänsklig intelligens) är



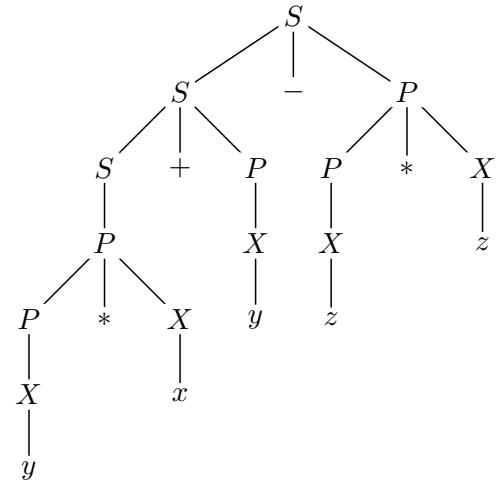


Andra automaten motsvarar ungefär det reguljära uttrycket $\diamond(ab(\epsilon \cup a))^*\diamond$ (som är ekvivalent med uttrycket i uppgiften).

2. Till vänster syns första trädet. (Jag utelämnar det andra.)

Grammatiken är entydig. Så länge vi inte använder parenteser får vi alltid ett derive-ringsträd som motsvarar en summa av produkter. Om vi skriver $x - x * y$ så tolkas det alltså "automatiskt" som $x-[x*y]$. Grammatiken tvinger oss att explicit skriva $(x-x)*y$ om det är det vi menar. Deriveringen är då

$$\begin{aligned} S &\Rightarrow P \\ &\Rightarrow P * X \\ &\Rightarrow^2 X * y \\ &\Rightarrow (S) * y \\ &\Rightarrow^* (x - x) * y. \end{aligned}$$



Dessutom är en summa som betår av flera termer en summa plus/minus en produkt (regeln är $S \to S + P$ istället för $S \to S + S$) som medför att t ex $x + x + x - x$ tolkas som $[[x+x]+x]-x$. Tolkningen $[x+x]+[x-x]$ är inte möjligt eftersom $x-x$ inte kan genereras utgående från icketerminalen P. Samma princip används när det gäller produkter.

3. En sammanfattning av felen:

   - Talet p får inte väljas – vi vet bara att *det finns ett p som uppfyller kraven* men vem säger att just $p = 4$ gör det? Med andra ord, p förblir nödvändigtvis en variabel i hela argumentationen.
   - Som en följd kan vi inte heller välja en specifik sträng. Dess storlek måste ju minst vara p. Pga att vi inte vet något om p har vi ingen annan möjlighet än att parametrisera u med p. I det här fallet är t ex $u = ab^p ab^p$ ett bra val. Observera dock att vi, så länge vi genom att använda p som en parameter ser till att strängens längd minst är p, får välja en typ av sträng som passar oss eftersom lemmat säger att (i)–(iii) gäller *för alla* dessa strängar. Ett motexempel räcker alltså.
   - Sist men inte minst är det inte tillräckligt att bara kolla en enda uppdelning av u. Lemmat säger ju endast att u *kan* uppdelas på ett sätt som uppfyller kraven. För att visa att det inte stämmer måste alla tänkbara uppdelningar prövas. I föreliggande fall ger det två möjligheter (om man tänker på kraven $y \neq \epsilon$ och $|xy| \leq p$). Den första är att $x = \epsilon$ och därmed $y = ab^k$ för något $k < p$. Så kan inte vara fallet eftersom $xy^0z = xz = z \notin A$ (det innehåller bara ett a). Det andra fallet är att $x = ab^k$ och $y = b^l$ där $l \geq 1$. Men det är inte heller möjligt eftersom $xz = ab^{n-l}ab^n$ igen inte är ett element i språket. (Argumentet funkar lika bra med $xyyz = ab^{n+l}ab^n$.)