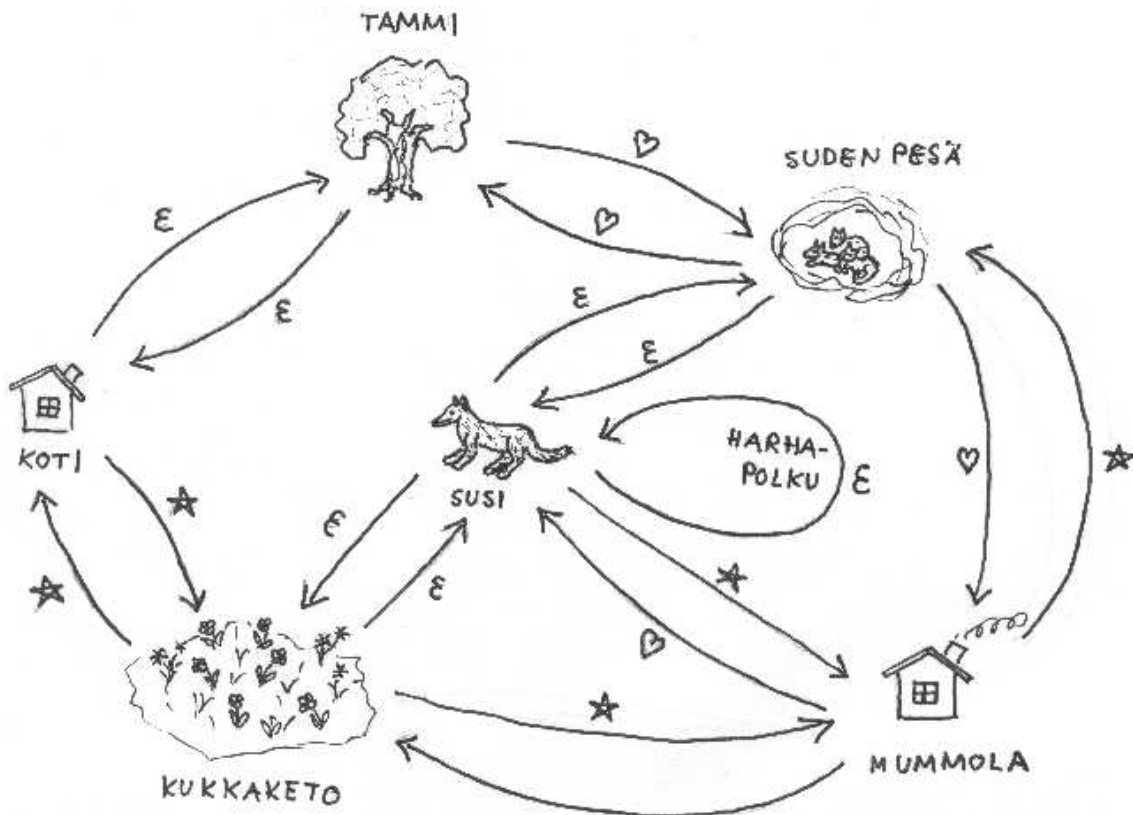


3.4.3  $\epsilon$ -automaatti

## Ongelma: Peikkojen pipariautomaatti

Punahilkka haluaa vierailla Mummonsa luona. Metsässä risteilee useita polkuja, joista osaa voi kulkea vapaasti, mutta osaa vartioi peikko. Peikot vaativat läpipääsystä tulla yhden piparkakun – joko täden- tai sydämenmuotoisen mieltymyksensä mukaan. Karttaan on merkitty peikkojen kullakin polulla vaatimat piparkakut.  $\epsilon$  tarkoittaa sitä, ettei polulta peritä mitään tulla. Äiti on kuitenkin antanut Punahilkalle vain yhden tähtipiparkakun. Miten pitkälle Punahilkka pääsee? Entä yhdellä sydänpiparkakulla?



Kiltti susi kutsuu Punahilkkan pesäänsä ja tarjoaa yhden sydänpiparkakun. Nyt Punahilkka pääsee Mummolaan, mutta miten hän pääsee takaisin kotiin?

$\epsilon$ -automaatti on epädeterministinen äärellinen automaatti, jossa sallitaan  $\epsilon$ -siirtymät ts. siirtymät tyhjällä merkkijonolla. Kyseessä on siis äärellisten automaattien mallin laajennus

*Määritelmä:*  $\epsilon$ -automaatti on viisikko  $M = (Q, \Sigma, \delta, q_0, F)$ , missä siirtymäfunktio  $\delta$  on kuvaus

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$$

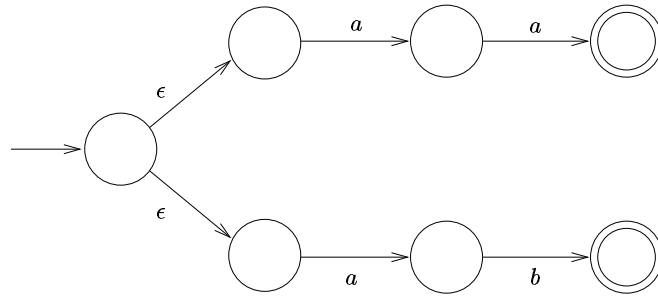
Tilanne  $(q, w)$  voi johtaa suoraan tilanteeseen  $(q', w')$

$$(q, w) \vdash_M (q', w'),$$

jos

- (i)  $w = aw'$  ( $a \in \Sigma$ ) ja  $q' \in \delta(q, a)$ ; tai
- (ii)  $w = w'$  ja  $q' \in \delta(q, \epsilon)$

Muut määritelmät kuten tavallisilla epädeterministisillä äärellisillä automaateilla



Kuva 3.4: Esimerkki: Kielen  $\{aa, ab\}$  tunnistava  $\epsilon$ -automaatti.

*Lemma:* Olkoon  $A = L(M)$  jollakin  $\epsilon$ -automaatilla  $M$ . Tällöin on olemassa myös  $\epsilon$ -siirtymätön epädeterministinen automaatti  $\widehat{M}$ , jolla  $A = L(\widehat{M})$ .

Todistus: Olkoon  $M = (Q, \Sigma, \delta, q_0, F)$  jokin  $\epsilon$ -automaatti. Automaatti  $\widehat{M}$  toimii muuten kuten  $M$ , mutta simuloi kunkin askelensa yhteydessä myös kaikki  $M$ :n mahdolliset  $\epsilon$ -siirtymät.

Formaalisti:

$$\widehat{M} = (Q, \Sigma, \hat{\delta}, q_0, \hat{F}),$$

missä

$$\hat{\delta}(q, a) = \{q' \in Q \mid (q, a) \vdash_M^* (q', \epsilon)\};$$

$$\hat{F} = \begin{cases} F \cup \{q_0\}, & \text{jos } (q_0, \epsilon) \vdash_M^* (q_f, \epsilon) \text{ jollain } q_f \in F; \\ F, & \text{muuten.} \end{cases} \quad \square$$

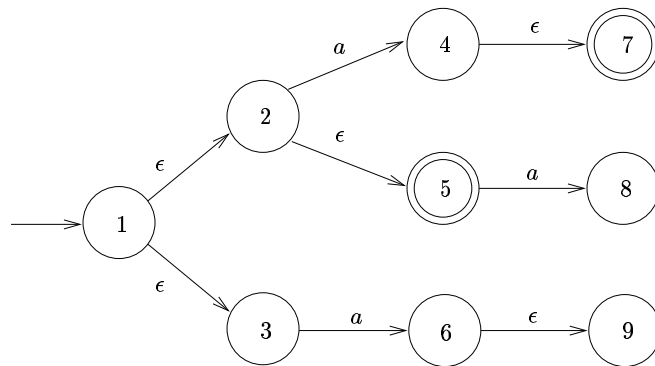
3.4.4  $\epsilon$ -siirtymien poisto

- Muodosta siirtymä  $q \xrightarrow{a} q'$ , jos  $(q, a) \vdash_M^* (q', \epsilon)$

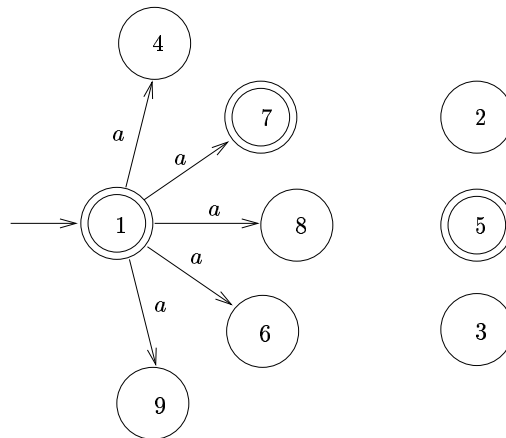
esim.  $q \xrightarrow{\epsilon} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon} q'$

- Lisää alkutila  $q_0$  lopputiloihin, jos  $(q_0, \epsilon) \vdash_M^* (q_f, \epsilon)$  ( $q_f \in F$ )

esim.  $q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} q_f$



Kuva 3.5:  $\epsilon$ -automaatti  $M$ .



- Huom! muodostettavissa  $\epsilon$ -automaateissa aina yksikäsitteiset alku- ja lopputilat.

### 3.4.5 Lausekeautomaatit

Määritellään vielä yksi äärellisten automaattien laajennus: lausekeautomaatti:

*Määritelmä:* Merk.  $\text{RE}_\Sigma$  = aakkoston  $\Sigma$  säännöllisten lausekkeiden joukko. *Lausekeautomaatti* on viisikko

$$M = (Q, \Sigma, \delta, q_0, F),$$

missä siirtymäfunktio  $\delta$  on äärellinen kuvaus

$$\delta : Q \times \text{RE}_\Sigma \rightarrow \mathcal{P}(Q)$$

(so.  $\delta(q, r) \neq \emptyset$  vain äärellisen monella parilla  $(q, r) \in Q \times \text{RE}_\Sigma$ ).

Yhden askelen tilannejohto määritellään:

$$(q, w) \vdash_M (q', w')$$

jos on  $q' \in \delta(q, r)$  jollakin sellaisella  $r \in \text{RE}_\Sigma$ , että  $w = zw'$ ,  $z \in L(r)$ . Muut määritelmät samat kuin  $\epsilon$ -epädeterministisellä automaatilla

## 3.5 Äärelliset automaatit ja säännölliset kielet

Osoitetaan seuraava tulos:

Kieli on säännöllinen  $\Leftrightarrow$  Kieli voidaan tunnistaa äärellisellä automaatilla.

Todistuksen idea:

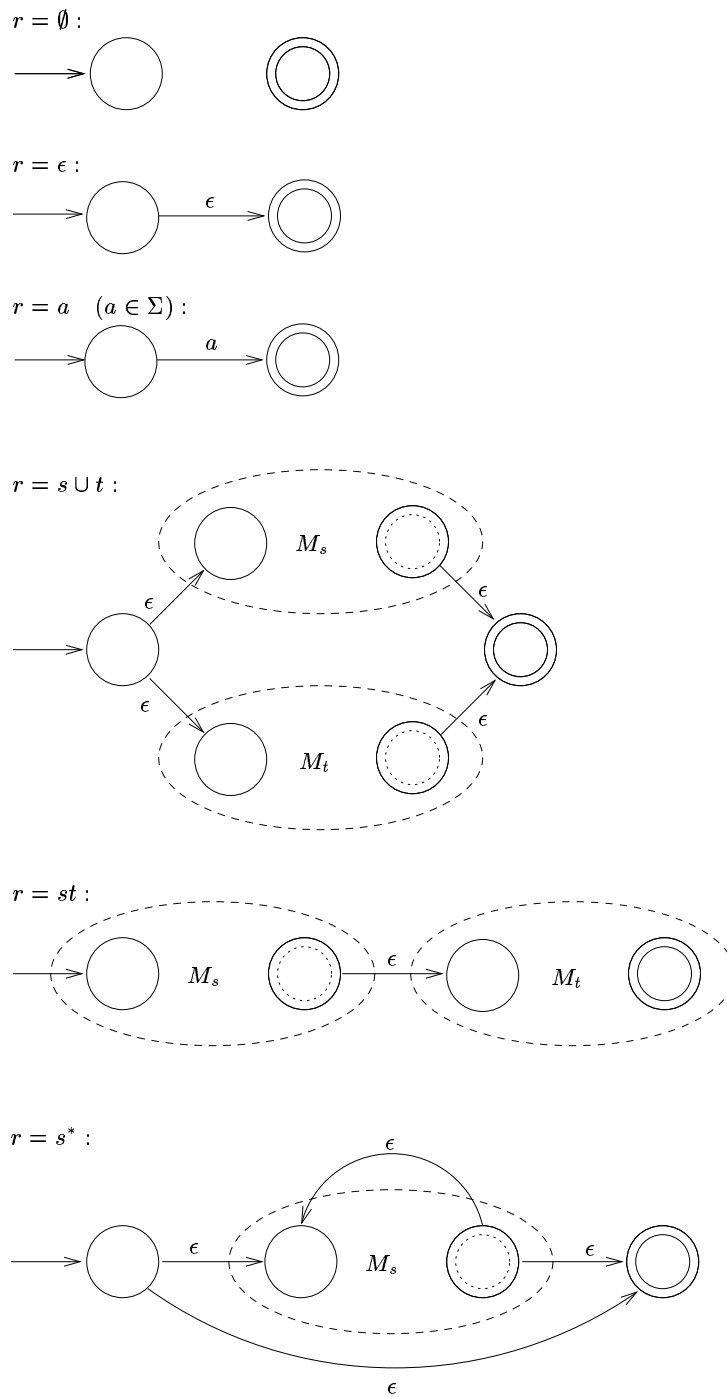
1. Kieli  $L(r)$  on säännöllinen  $\Rightarrow L(r)$  voidaan tunnistaa äärellisellä automaatilla  $M$ :
  - Muodostetaan säännöllistä lauseketta  $r$  vastaava  $\epsilon$ -automaatti
  - Tällöin on olemassa vastaava  $\epsilon$ -siirtymätön epädeterministinen automaatti
  - Haluattessa epädeterministinen automaatti voidaan vielä determinisoida (ja minimoida)
2. Kieli  $L(M)$  voidaan tunnistaa äärellisellä automaatilla  $M \Rightarrow L(M)$  on säännöllinen kieli:

- Tarkastellaan äärellisten automaattien laajennosta, *lausekeautomaatteja* - jos väite pätee lausekeautomaateille, se pätee myös tavallisille äärellisille automaateille (jotka ovat niiden erikoistapaus)
- Redusoidaan lausekeautomaatti korkeintaan 2-tilaiseksi automaatiksi, josta voidaan lukea suoraan vastaava säännöllinen lauseke

*Lause:* Jokainen säännöllinen kieli voidaan tunnistaa äärellisellä automaatilla.

Todistus:

- Muodostetaan mielivaltaista säännöllistä lauseketta  $r$  vastaava  $\epsilon$ -automaatti  $M_r$ , jolla  $L(M_r) = L(r)$  (ks. kuva 3.6)
- $M_r$ :stä voidaan poistaa  $\epsilon$ -siirtymät edellisen lemmän mukaisesti, ja tarvittaessa voidaan syntyvä epädeterministinen automaatti determinisoida aiemmin esitetyn konstruktion avulla.  $\square$



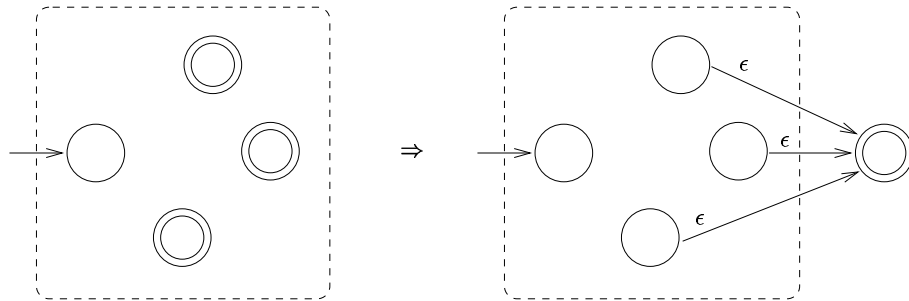
Kuva 3.6: Lauseketta  $r$  vastaavan  $\epsilon$ -automaatin  $M_r$  muodostaminen.

*Lause:* Jokainen äärellisellä automaatilla tunnistettava kieli on säännöllinen.

*Todistus:* Osoitetaan, että jokainen lausekeautomaatilla tunnistettava kieli on säännöllinen.

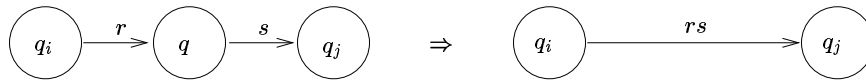
Idea: Redusoidaan lausekeautomaatti s.e. siitä voidaan lukea vastaava säännöllinen lauseke:

- Yhdistetään  $M$ :n lopputilat yhdeksi  $\epsilon$ -siirtymillä (ks. kuva 3.7)
- while (muita kuin alku- ja lopputiloja) {
  - valitse  $q$ ,  $q \neq q_0$ ,  $q \neq q_f$  ( $q_f \in F$ )
  - poista  $q$  reitiltä  $q_i \rightarrow q \rightarrow q_j$   $q_i = q$ :n edeltäjätila ja  $q_j = q$ :n seuraajatilaa) reduktiosäännöllä (kuva 3.8) (Huom! Muista käydä läpi kaikki  $q$ :ta sisältävät polut!)
  - yhdistä rinnakkaiset siirtymät (kuva 3.9)
- }

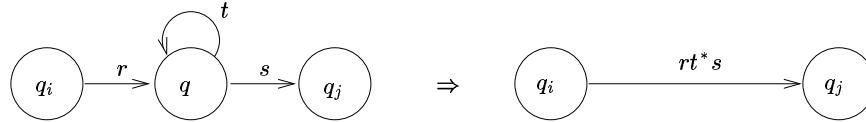


Kuva 3.7: Lausekeautomaatin lopputilojen yhdistäminen.

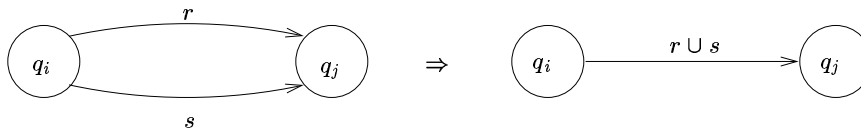
(i):



(ii):



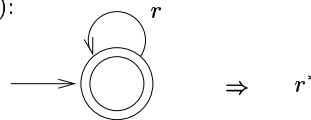
Kuva 3.8: Tilan poistaminen lausekeautomaatista.



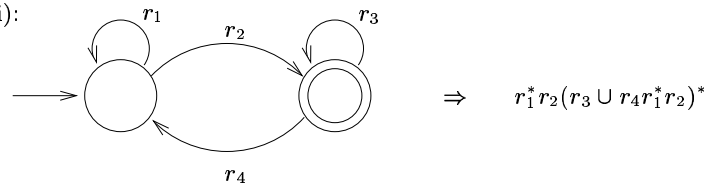
Kuva 3.9: Rinnakkaisten siirtymien yhdistäminen lausekeautomaatissa.

- Tiivistyksen päättyessä jäljellä olevaa enintään 2-tilaista automaattia vastaava säännöllinen lauseke muodostetaan kuten kuvassa 3.10.

(i):



(ii):

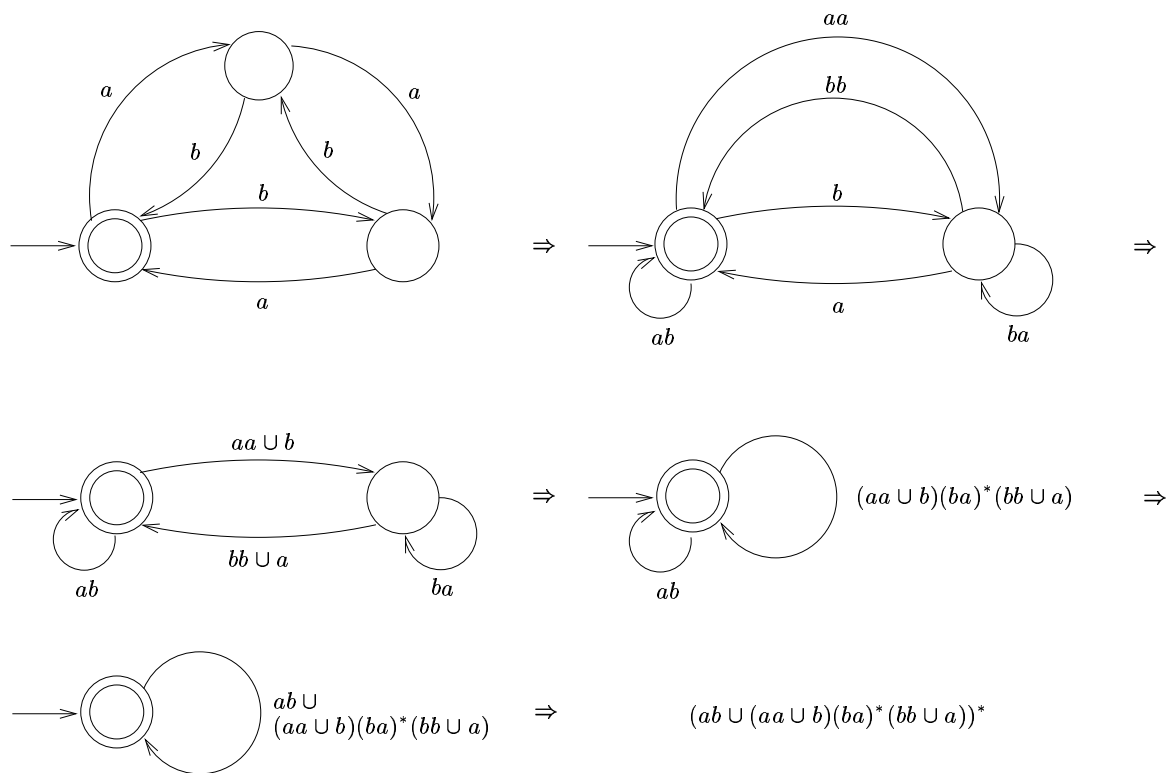


Kuva 3.10: Säännöllisen lausekkeen muodostaminen redusoidusta lausekeautomaatista.

□



**Esimerkki:**



### 3.6 Hauska lisätieto: Säännöllisen kielen sulkeumaominaisuudet

Automaattien avulla voidaan osoittaa seuraava tulos:

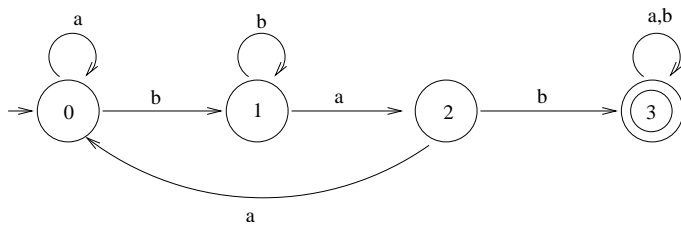
*Lause:* Olkoon  $L_1$  ja  $L_2$  säännöllisiä kieliä. Tällöin myös

1.  $L_1 \cup L_2$  (kielten yhdiste)
2.  $L_1 \cap L_2$  (kielten leikkaus)
3.  $L_1 L_2$  (kielten katenaatio)
4.  $\overline{L_1} = \Sigma^* \setminus L_1$  (kielen komplementti)
5.  $(L_1)^*$  (kielen sulkeuma)
6.  $(L_1)^R$  (käänteiskieli, jossa kaikki  $L_1$ :n sanat on kirjoitettu takaperin)

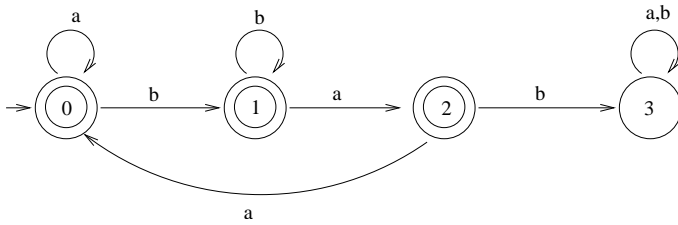
ovat säännöllisiä.

*Todistus:* Harjoitustehtävä!

Esim.



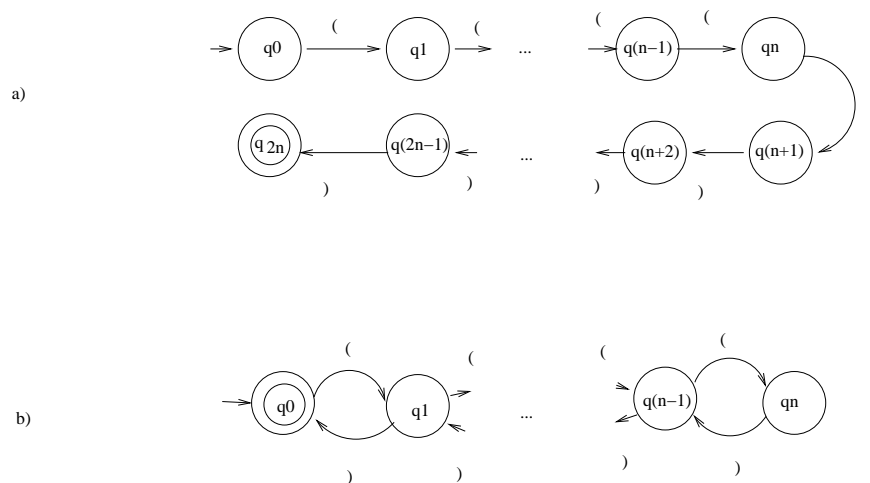
Kuva 3.11: Automaatti  $M$ , joka tunnistaa kielen  $L(M) = \{w \in \{a, b\}^* \mid w \text{ sisältää merkkijonon } bab\}$ .



Kuva 3.12: Komplementtiautomaatti  $\overline{M}$ , joka tunnistaa kielen  $L(\overline{M}) = \{w \in \{a, b\}^* \mid w \text{ ei sisällä merkkijonoa } bab\}$ .

### 3.7 Säännöllisten kielten rajoituksista

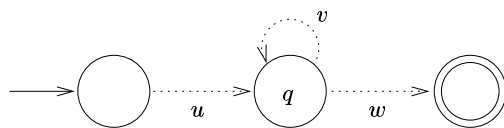
- Minkä tahansa aakkoston formaaleja kieliä (päättöongelmia) on ylinumeroituva määrä, mutta säännöllisiä lausekkeita vain numeroituva määrä  $\Rightarrow$  kaikki kielet eivät voi olla säännöllisiä
- Perusrajoitus: äärellisillä automaateilla on vain rajallinen “muisti”
- Äärelliset kielet ovat aina säännöllisiä
- Milloin on ääretön kieli säännöllinen?
  - oltava jokin toistuva rakenne (sulkeuma)
  - vastaavassa automaatissa silmukka
- Esim. tasapainoisten sulkujonojen muodostama kieli  $L_{\text{match}} = \{(k)^k \mid k \geq 0\}$  ei ole säännöllinen, eli sitä ei voi tunnistaa äärellisellä automaatilla.



- tunnistaa tasan  $n$  sulkuparia sisältävän merkkijonon
- entä jos enemmän tai vähemmän sisäkkäisiä sulkuja?
- esim.  $a^{n-1} \in L_{\text{match}}$ ,  $a^{n+1} \in L_{\text{match}}$ , mutta automaatti ei tunnista niitä
- Entä kieli  $\{a^k b^k \mid k \geq 0\}^*$ ?  
 $\Rightarrow$  ei voida tunnistaa äärellisellä automaatilla (eikä siten myöskään mv. aritmeettisia lausekkeita)
- “Pumppauslemma” formalisoi tämän rajallisen muistin idean
- Idea: mitä tahansa annetun säännöllisen kielen riittävän pitkää merkkijonoa voidaan “pumppata” keskeltä, ilman että kielen tunnistava äärellinen automaatti huomaa muutosta
- *Lemma*[Pumppauslemma]: Olkoon  $A$  säännöllinen kieli. Tällöin on olemassa  $n \geq 1$  s.e. mikä tahansa  $x \in A$ ,  $|x| \geq n$ , voidaan jakaa osiin  $x = uvw$ ,  $|uv| \leq n$ ,  $|v| \geq 1$ , ja  $uv^i w \in A$  kaikilla  $i = 0, 1, 2, \dots$

Todistus:

- Olk.  $M$  jokin  $A$ :n tunnistava deterministinen äärellinen automaatti ja  $n = M$ :n tilojen määrä.
- Tarkastellaan automaatin läpikäymiä tiloja sen tunnistessa merkkijonoa  $x \in A$ ,  $|x| \geq n$ :
  - \* Koska  $|x| \geq n \Rightarrow$  täytyy kulkea jonkin tilan kautta (ainakin) kaksi kertaa (itse asiassa jo  $x$ :n  $n$ :ää ensimmäistä merkkiä käsitellessään)
  - \* Olk.  $q$  ensimmäinen tila, jonka automaatti toistaa  $x$ :ää käsitellessään.
  - \* Olk.  $u = M$ :n käsittelemä  $x$ :n alkuosa sen tullessa ensimmäisen kerran tilaan  $q$
  - \* Olk.  $v =$  se osa  $x$ :stä  $u$ :n jälkeen jonka  $M$  käsittelee ennen ensimmäistä paluutaan  $q$ :hun, ja
  - \*  $w =$  loput  $x$ :stä
  - \* Tällöin on  $|uv| \leq n$ ,  $|v| \geq 1$ , ja  $uv^i w \in A$  kaikilla  $i = 0, 1, 2, \dots$   $\square$



Kuva 3.13: Merkkijonon  $x = uvw \in A$  pumppaus.

- Pumpauslemma on kiinnostava vain äärettömille kielille. (Äärellisille kielille, jotka kaikki ovat säännöllisiä, voidaan valita  $n = \max\{|x| \mid x \in L\} + 1$ , jolloin ei ole olemassa yhtään  $x \in L$  s.e.  $|x| \geq n$ )
- Esim. Väite: Sulkulausekekieli

$$L = L_{\text{match}} = \{(^k)^k \mid k \geq 0\}.$$

on epäsäännöllinen.

Tod.: Vastaoletus:  $L$  on säännöllinen.  $\Rightarrow \exists$  jokin  $n \geq 1$ , jonka pituisia  $L$ :n merkkijonoja voidaan pumpata.

Valitaan  $x = (^n)^n$ , jolloin  $|x| = 2n > n$ . Lemman mukaan  $x$  voidaan jakaa pumpattavaksi osiin  $x = uvw$ ,  $|uv| \leq n$ ,  $|v| \geq 1$ ; siis on oltava

$$u = (^i, v = (^j, w = (^{n-(i+j)})^n, \quad \text{missä } i \leq n-1, j \geq 1.$$

Mutta esimerkiksi "0-kertaisesti" pumpattu merkkijono  $uv^0w = (^{i(n-(i+j))}^n = (^{n-j})^n$  ei kuulu kieleen  $L$ . Ristiriita. Siten  $L$  ei voi olla säännöllinen.  $\square$

- Huom! Pumpauslemma ei sano, että voimme valita  $u$ :n ja  $v$ :n haluamallamme tavalla, vaan todistuksessa meidän on kokeiltava kaikkia jakoja  $x = uvw$  s.e.  $|uv| \leq n$  ja  $|v| \geq 1$
- Todistusstrategia: Valitaan tarkasteltavan jonon pituudeksi esim.  $2n$ , jossa ensimmäiset  $n$  merkkiä muodostavat  $uv$ :n, muttei kiinnitetä tarkkaan ottaen miten
- Esim. Väite:  $L = \{a^k b^l c^{k+l} \mid k, l \geq 0\}$  ei ole säännöllinen.  
Tod.: Vastaoletus:  $L$  on säännöllinen  $\Rightarrow \exists n \geq 1$ , jonka pituisia merkkijonoja voidaan pumpata. Valitaan  $x = a^n b^l c^{n+l}$ , jollakin  $l \geq 0$ . Nyt  $|x| = 2n + 2l > n$ . Koska  $|uv| \leq n$ ,  $uv$  koostuu vain  $a$ -merkeistä ja  $m = |v| \geq 1$ . Tällöin  $uv^0w = a^{n-m} b^l c^{n+l} \notin L$ . Ristiriita. Siten  $L$  ei voi olla säännöllinen.  $\square$

Esim. 2: Väite:  $L(a^m b^n c^{m+n})$  ei ole säännöllinen.

Tod: Tarkastellaan merkkijonoja, joiden pituus vähintään  $n = k + l$ . Valitaan  $x = a^k b^l c^{k+l}$ ,  $|x| = 2k + 2l > n$ . Nyt osa  $uv$  koostuu  $x$ :n korkeintaan  $k + l$ :stä ensimmäisestä merkistä (sisältää vain  $a$ :ta ja mahd.  $b$ :tä) ja  $v$ :n olt. vähintään 1-merkkinen. Kaksi jakovaihtoehtoa:

1.  $u = a^i, v = a^{k-i} b^j, w = b^{l-j} c^{k+l}$ , missä  $i \neq k$  tai  $j > 0$ . Tällöin 0-kertaisesti pumpattu jono on  $uv^0w = a^i b^{l-j} c^{k+l}$ , joka kuuluu kieleen vain jos,  $i = k$  ja  $j = 0$ , mikä ei mahd.
2.  $u = a^k b^i, v = b^j, w = b^{l-i-j} c^{k+l}$ , missä  $j \geq 1$ .  $uv^0w = a^k b^i b^{l-i-j} c^{k+l} = a^k b^{l-j} c^{k+l}$ , joka kuuluu kieleen vain, jos  $j = 0$ , mikä ei mahd.

⇒ Kieli ei siis säännöllinen.

- Esim. 3: Väite: Kieli  $L_{pr} = \{1^p \mid p \text{ on alkuluku}\}$  ei ole säännöllinen.  
Tod.: Vastaoletus:  $L_{pr}$  on säännöllinen ⇒ ∃ jokin  $n \geq 1$ , jota pitempiä  $L_{pr}$ :n merkkijonoja voidaan pumpata.  
Valitaan  $x = 1^p$ , jollakin alkuluvulla  $p \geq n + 2$ . Pumppauslemman mukaan  $x = uvw$ ,  $|uv| \leq n$ ,  $m = |v| \geq 1$ . Nyt  $|uv^{p-m}w| = |uw| + (p-m)|v| = p-m + (p-m)m = (m+1)(p-m)$ . Tämä on yhdistetty luku, koska  $m+1 \geq 2$  ja  $p-m \geq n+2-m \geq 2$  (sillä  $m = |v| \leq |uv| \leq n$ ). Siis  $uv^{p-m}w \notin L_{pr}$ .  
Ristiriita. □

## Näin näytät pitkää nenää Pumppauslemmalle

(heuristisia ohjeita PL-todistuksiin )

1. Mieti annettua kieltä: mikä ehto täytyy rikkoa, jotta merkkijono ei kuulu kieleen?
  - ehto voi koskea esim. tiettyjen merkkien lukumäärien keskinäistä suhdetta (esim.  $L_1 = \{a^k b^m c^m \mid k, m = 0, 1, 2, \dots\}$ ,  $L_2 = \{a^m b^{2m} \mid m = 0, 1, 2, \dots\}$ )
  - tai liittyä alkulukuihin, joiden jono on epäsäännöllinen  $L_3 = \{a^{p_i} \mid p_i \text{ on } i\text{:s alkuluku}\}$
  - tai sanan osia: esim. sanan alku- ja loppuosa riippuvat jotenkin toisistaan  $L_4 = \{ww^R \mid w \in \Sigma^*, w^R \text{ on } w \text{ takaperin kirjoitettuna}\}$ ,  $L_5 = \{ww \mid w \in \Sigma^*\}$
  
2. Mieti, mikä olisi yksinkertaisin merkkijono, jossa esiintyvät eheys ehdon osapuolet. Joskus kielessä on todistuksen kannalta täysin turhia (säännöllisiä) osia, esim.  $L_1$ :ssä  $a$ :n lukumäärällä ei ole mitään väliä – voidaan valita merkkijono  $b^m c^m$ .
  - jos ehdon osapuolten välissä on tuollainen säännöllinen osa, se saattaa olla tarpeen osapuolien erottamiseen toisistaan, esim.  $L_6 = \{a^m b^k a^m \mid m, k = 0, 1, 2, \dots\}$ :ssä tarvitaan ainakin yksi  $b$  erottamaan alku- ja loppu- $a$ :t. Valitaan esim.  $a^m b a^m$ .
  - jos kielen alku- ja loppuosa riippuvat jotenkin toisistaan, mutta muuten ne saavat olla mitä tahansa, riittää erottaa alku- ja loppuosa toisistaan, esim. kielen  $L_5$  kohdalla voidaan valita  $a^m b a^m b$  tai  $b a^m b a^m$ .
  
3. Valitse nyt se mystinen  $n$  siten, että eheys ehdon toinen osapuoli kuuluu ensimmäiseen  $n$ :ään merkkiin ja sitä päästään pumppaamaan. Toinen tavoite on, että merkkijonon mahdollisia jakoja osiin  $uvw$  olisi mahdollisimman vähän (tämä ei ole kuitenkaan yhtä kriittistä, säästää vain työtä).
  - esim.  $x = a^m b^m$ . Kannattaa valita  $n = m$ , jolloin osat  $uv$  kuuluvat  $a^m$ :ään. Tällöin pääsemme pumppaamaan  $a$ :ta ja eheys ehto rikkoontuu. Huom! Voimme valita myös  $n = 2m$ , jolloin tulee lisää töitä: Pumpattava osa voi koostua vain  $a$ :sta, vain  $b$ :stä tai molemmista (muotoa  $a^i b^j$ ). Viimeisessä tapauksessa pumppaus  $uv^2 w$  sotkee asiat ( $a^i b^j a^i b^j$  ei kuulu kieleen).
  
4. Testaa nyt **kaikki** Pumppauslemman mukaiset jaot  $x = uvw$ ,  $|uv| \leq n$  ja  $v \neq \epsilon$ . Jokaisella jaolla kokeile pumppausta  $i$ :n arvoilla  $0, 2, 3, \dots$  kunnes löytyy sellainen  $i$ , että  $uv^i w \notin A$ .

- tavallisesti arvo  $i = 0$  tai viimeistään  $i = 2$  tuottaa halutun tuloksen.

5. Jos onnistuit rikkomaan eheyshdon kaikilla jaoilla  $uvw$ , voit huudahtaa voittonriemuisena: ”Heureka!” Onneksi olkoon!

### Miksi pumppauslemma käytetään näin?

Esitetään PL loogisena lauseena:

$R(A) \Rightarrow \exists n \forall x P(x) \exists uvw Q(x, u, v, w) \forall i S(u, v, w, i)$ , missä

$P(x)$  on lyhenne ehdolle  $x \in A \wedge |x| \geq n$

$Q(x, u, v, w)$  ehdolle  $x = uvw \wedge |uv| \leq n \wedge |v| \geq 1$ , ja

$S(u, v, w, i)$  ehdolle  $uv^i w \in A$ .

Käytetään kontrapositiosääntöä:

$$A \Rightarrow B \equiv \neg B \Rightarrow \neg A$$

Toisaalta tiedämme:

$$\neg \exists x P(x) \equiv \forall x \neg P(x)$$

$$\neg \forall x P(x) \equiv \exists x \neg P(x).$$

Käännetään Pumppauslemma näiden avulla ja se on yhä tosi:

$$\begin{aligned} & \neg(\exists n \forall x P(x) \exists uvw Q(x, u, v, w) \forall i S(u, v, w, i)) \Rightarrow \neg R(A) \\ & \equiv \forall n \neg(\forall x P(x) \exists uvw Q(x, u, v, w) \forall i S(u, v, w, i)) \Rightarrow \neg R(A) \\ & \equiv \forall n \exists x P(x) \neg(\exists uvw Q(x, u, v, w) \forall i S(u, v, w, i)) \Rightarrow \neg R(A) \\ & \equiv \forall n \exists x P(x) \forall uvw Q(x, u, v, w) \neg(\forall i S(u, v, w, i)) \Rightarrow \neg R(A) \\ & \equiv \forall n \exists x P(x) \forall uvw Q(x, u, v, w) \exists i \neg(S(u, v, w, i)) \Rightarrow \neg R(A). \end{aligned}$$

Siis: Jos kaikilla  $n$  on joku  $x$   $|x| \geq n$ , kaikilla  $uvw$   $|uv| \leq n$ ,  $|v| \geq 1$ , on olemassa  $i$  siten että  $uv^i w \notin A$ , niin  $A$  ei ole säännöllinen.



## Punahilkkapeli

Ahneet Peikot ovat niin innostuneita piparibisneksestään, että pyörittävät pipari-automaatteja ympäri metsää. Kuitenkin myös vanha sairas Mummo kaipaisi kipeästi pipareita. Punahilikka kumppaneineen keksii seuraavan ratkaisun: jos äiti alkaa leipoa epäsäännöllisen kielen mukaisia piparijonoja, eivät Peikot pysty rakentamaan koko kieltä tunnistavaa automaattia - eivätkä siis pysty saamaan kaikkia pipareita.

### Pelin idea

Punahilkkapelissä on kaksi joukkuetta, joista toiseen kuuluvat Punahilikka, Mummo ja Susi ja toiseen Peikot. Punahilkan tehtävänä on viedä Mummolle niin paljon pipareita, kuin hän pystyy, ilman että Peikot syövät niitä matkalla. Peikot näet syövät vain säännöllisiä piparijonoja (esim. (sydän \* tähti sydän\*)).

Äiti leipoo jatkuvasti uusia pipareita, jotka voivat olla tähden tai sydämenmuotoisia. Punahilkan ja Mummon pitäisi keksiä sellainen piparikieli, jonka mukaiset piparijonot Punahilikka pystyy kuljettamaan Peikkojen ohitse. Kiltti Susi auttaa Punahilikkaa ja Mummoa. Esim. kielen (tähti<sup>n</sup>sydän<sup>n</sup>)  $n \geq 1$  piparijonot (sanat) eivät kelpaa Peikoille.

Saatte korttipakan, joista kukin kuvaa joko säännöllisen tai epäsäännöllisen piparikielen.

### Pelin kulku

**Punahilkan puoli:** Valitkaa satunnainen kortti pakasta.

**Peikot:** Yrittäkää osoittaa, että kyseinen piparikieli on säännöllinen. Jos onnistutte, niin saatte kaikki kielen kuvaamat piparit.

**Punahilkan puoli:** Jos Peikot eivät onnistuneet, niin voitte yrittää kuljetusta. Osoittakaa, että piparikieli ei ole säännöllinen. Jos onnistutte, saatte vietyä piparit onnellisesti Mummolle.

Peikot ja Punahilkan puoli kinastelevat piparikielestä niin kauan, kunnes joko Peikot saavat sen tai Punahilikka saa vietyä sen läpi. Tämän jälkeen Punahilikka kumppaneineen yrittää seuraavaa piparikieltä.

Se puoli, joka on lopuksi kerännyt eniten piparikieliä, voittaa pelin. Vaihtakaa välillä puolia!

## 3.8 Ekskursio: Säännöllisten kielten sovelluksia

### 3.8.1 Hahmontunnistus

Tavoite: halutaan löytää tekstistä  $y$  hahmo  $x$ . Edellä tutustuimme jo yhteen tapaan ratkaista tämä ongelma äärellisten automaattien avulla. Laadimme äärellisen automaatin, joka tunnistaa hahmon  $x$ . Jos haluamme lisäksi sanan  $x$  sijainnin merkkijonossa, täytyy lisätä laskuri, joka pitää kirjaa luetuista merkeistä (siirtymistä), kunnes hahmo tunnistettu.

Äärellisiä automaatteja voidaan käyttää toisellakin tapaa hahmon tunnistukseen. Muodostetaan äärellinen automaatti tekstin  $y$  suffikseista. Automaatin kuhunkin kaareen (siirtymään) liittyy sekä kirjain että numero. Kunkin siirtymän yhteydessä katenoidaan kaaren kirjain luettujen merkkien jonoon. Lisäksi kasvatetaan sijaintilaskuria siirtymän ilmoittamalla askelmäärällä. Tuloksena saadaan pari  $(p, i)$ , missä  $p$  on suffiksi  $i$  kertoo sen sijainnin (montako merkkiä luettava tekstin alusta, jotta kyseinen suffiksi löytyy).

Tällainen *suffiksiautomaatti* toimii itse asiassa tekstin hakemistorakenteena, jonka avulla voidaan etsiä mikä tahansa merkkijono tekstistä.

Muistathan myös, että säännöllisillä lausekkeilla voidaan helposti kuvata etsittäviä hahmoja, esim. tiedon haussa. Vastaava hakukone voidaan toteuttaa äärellisenä automaattina.

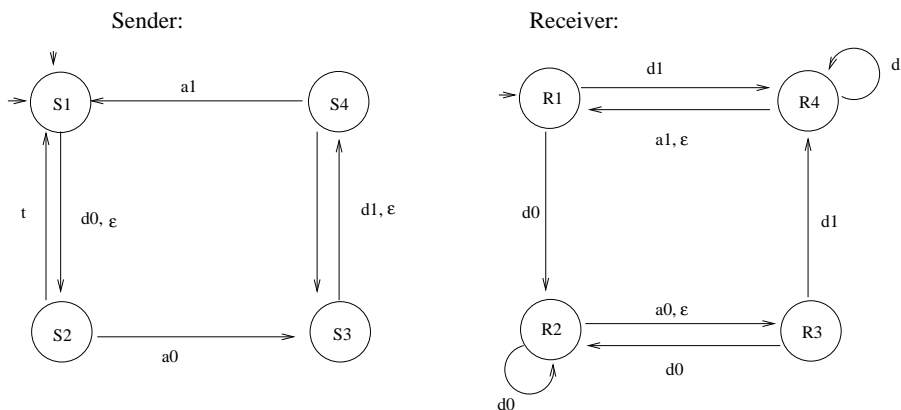
### 3.8.2 Viestinvälitysprotokollat äärellisinä automaatteina

Myös erilaisia protokollia voidaan kuvata äärellisinä automaatteina. Tällä tavalla voidaan esimerkiksi verifioida, että jokin viestinvälitysprotokolla toimii oikein.

Tarkastellaan yksikertaista AB (alternating bit)-protokollaa, jossa lähettäjäprosessi  $S$  ja vastaanottajaprosessi  $R$  kommunikoivat *vuorosuuntaisen kanavan* kautta. Vuorosuuntaisessa kanavassa viestejä saa lähettää vain yhteen suuntaan kerrallaan. Kanava voi hukata tai vääristää sanomia, mutta viestien järjestys säilyy eikä kanava monista sanomia.

Koska kanavassa on kerrallaan vain yksi viesti, riittää viestien esittämiseen kaksi bittiä, 1 ja 0. Lähettäjä yrittää ensin lähettää viestin  $d0$  ja jos se saa siitä kuittauksen  $a0$ , se siirtyy lähettämään viestiä  $d1$ . Jos se saa  $d1$ :stäkin kuittauksen  $a1$ , se voi lähettää taas viestin, jota merkitään  $d0$ :lla jne. Lähettäjä ei kuitenkaan lähetä uutta viestiä ennen kuin se on saanut edellisestä kuittauksen. Jos kuittausta ei kuulu tietyn ajan  $t$  kuluessa se lähettää viestin uudestaan. Vastaanottaja puolestaan lukee kaikki kanavasta tulevat viestit ja lähettää niistä kuittauksen, mahdollisesti useaankin kertaan (jos sama viesti tulee useita kertoja).

Lähettäjän ja vastaanottajan toiminta voidaan kuvata seuraavanlaisina epädeterministisinä ( $\epsilon$ -) automaateina.



### 3.8.3 Leksikaalinen analyysi

Leksikaalinen analyysi on kääntäjän osa, joka jakaa lähdekoodin loogisesti yhteenkuuluviin osiin (tokens). Tällaisia osia voivat olla esimerkiksi varatut sanat, tunnisteen, ym. Leksikaalinen analysointia voidaan toteuttaa esim. UNIX-komentojen `lex` ja `flex` (edellisen GNU-versio) avulla. `Lex`- ja `flex`-komento saavat syötteenään listan säännöllisistä lausekkeista sekä ohjeet, mitä tehdä vastaavan osan kohdalla, ja tuottavat tämän pohjalta leksikaalisen analysointin kyseisille säännöille.

Syöte voisi olla esimerkiksi seuraavanlainen:

```

else                {return(ELSE); }
[A-Za-z][A-Za-z0-9]* {code to enter the found identifier in the symbol table;
                    return(ID); }
>=                 {return(GE); }
=                  {return(EQ); }
...

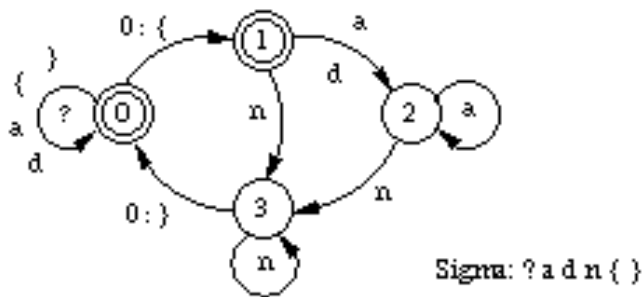
```

Huom! Tällaisessa määrittelyssä varattu sana "else" täsmää myös tunnisteiden kuvaukseen. Ongelma voidaan ratkaista antamalla säännöille prioriteetijärjestys: valitaan ensimmäinen lauseke, johon merkkijono täsmää.

### 3.8.4 Luonnollisen kielen esiprosessointi

Luonnollisen kielen ymmärtäminen tai tiedon eristäminen tekstistä (joka ei pyrikään koko tekstin ymmärtämiseen, vaan etsimään siitä ainoastaan haluttua tietoa) on hyvin haastava tehtävä, johon säännölliset kielet eivät riitä. Niitä voidaan kuitenkin käyttää apuna tekstin esiprosessoinnissa samaan tapaan kuin ohjelmointikielen leksikaalisissa analyysissä.

Eräs lähestymistapa on muodostaa *transduktori* (engl. *transducer*), joka on eräänlainen uudelleenkirjoitettava äärellinen automaatti. Laajennetaan esimerkiksi lausekeautomaattia siten, että jokaisen siirtymän yhteydessä automaatti sekä lukee merkkijonon että korvaa sen jollain toisella merkkijonolla (tai tyhjällä merkkijonolla). Tällaisella transduktorilla voidaan esimerkiksi lukea pois turhat täytesanat (kuten englannin artikkelit), tunnistaa prepositiot (esim. korvataan symbolilla *PRE*) tai keksiä heuristiikka erisnimien tunnistamiseen (korvataan sovitulla symbolilla). Alla esimerkki transduktorista, joka pyrkii erottamaan (englanninkielisestä) lauseesta substantiivilmaukset (noun phrases), jotka ovat muotoa  $[(d)a * n+]$ , missä  $d$  on artikkeli,  $a$  on adjektiivi ja  $n$  on substantiivi. (Huom! Sanaluokat on siis jo tunnistettu jollain tapaa.)



Kuva 3.14: Transduktori, joka kirjoittaa sulut kaikkien tunnistamiensa substantiivilausekkeiden ympärille.