Formalisoimme nyt edellä kuvatun laskennan.

Jos $M = (Q, \Sigma, \delta, q_0, F)$ on äärellinen automaatti ja $w = w_1 \dots w_n$ on $n$ merkkiä pitkä aakkoston $\Sigma$ merkkijono, niin automaatti $M$ *hyväksyy* merkkijonon $w$, jos on olemassa $n+1$ tilan jono $(r_0, r_1, \dots, r_n) \in Q^{n+1}$, jolla

- $r_0 = q_0$,
- $r_{i+1} = \delta(r_i, w_{i+1})$ kun $0 \leq i \leq n-1$ ja
- $r_n \in F$.

Siis laskenta alkaa alkutilasta, noudattaa syötemerkkien mukaisia siirtymiä ja päätyy hyväksyvään tilaan.

Jos $M$ ei hyväksy, se *hylkää* merkkijonon $w$.



Hyväksymisen ja hylkäämisen kannalta vain tilajonon ensimmäisellä ja viimeisellä tilalla on väliä. Siksi on hyödyllistä määritellä kuvaus $\delta^* \colon Q \times \Sigma^* \to Q$ seuraavasti:

> $\delta^*(q, w)$ on se tila, johon tilasta $q$ päädytään merkkijonolla $w$.

Erityisesti $\delta^*(q, \varepsilon) = q$ kaikilla $q$. Toinen perustapaus on $\delta^*(q, a) = \delta(q, a)$ kaikilla $a \in \Sigma$.

Nyt voidaan sanoa, että automaatti hyväksyy merkkijonon $w$, jos $\delta^*(q_0, w) \in F$.



Funktio $\delta^*$ voidaan määritellä siirtymäfunktion $\delta$ avulla rekursiivisesti:

$$\delta^*(q, w) = \begin{cases} q & \text{jos } w = \varepsilon \\ \delta^*(\delta(q, a), v) & \text{jos } w = av \text{ missä } a \in \Sigma. \end{cases}$$

Tämä tarkoittaa samaa kuin rekursiivinen proseduuri

DeltaStar$(q, w)$

1 olkoon $w = w_1 \dots w_n$, missä $w_i \in \Sigma$ kaikilla $i$
2 **if** $n = 0$
3    **then return** $q$ $\quad \triangleright\ w = \varepsilon$
4    **else return** DeltaStar$(\delta(q, w_1), w_2 \dots w_n)$
$\quad \triangleright\ a = w_1,\ v = w_2 \dots w_n$

Rekursio aina päättyy, sillä

1. rivin 4 rekursiivisessa kutsussa parametrina olevan merkkijonon pituus aina pienenee ja
2. merkkijonon pituudella on ehdoton alaraja 0.



## Reunahuomatus: miksi formalismia

Formalismin avulla

- määritellään terminologia ja merkintätavat, joilla asioista voidaan puhua matematiikan kielellä, sekä
- varmistetaan, että mikään yksityiskohta ei jää epäselväksi tai tulkinnanvaraiseksi.

Jatkossa haluamme esittää väitteitä muotoa

> Kaikki äärelliset automaatit toteuttavat ehdon $p$

ja konstruktioita

> Mistä tahansa ehdon $r$ toteuttavasta automaatista voidaan muodostaa automaatti, joka toteuttaa ehdon $s$.

Tätä voidaan tiettyyn rajaan saakka tehdä piirtämällä kuvia, mutta uskottavuuden ja ymmärrettävyyden rajat tulevat melko pian vastaan.

Matemaattinen formalismi on voimakas työkalu, jota on hyvä opetella käyttämään.

Automaatin $M$ *tunnistama kieli* $L(M)$ on kaikkien niiden merkkijonojen joukko, jotka $M$ hyväksyy. Esim. edellistä automaattia $M_1$ tarkastelemalla voidaan todeta, että se hyväksyy tasan ne merkkijonot, jotka loppuvat 01; siis

$$L(M_1) = \{ w_1 \ldots w_n 01 \mid n \geq 0, w_i \in \{0,1\} \text{ kaikilla } i. \}$$

Sanomme, että kieli $A$ on *säännöllinen*, jos jokin äärellinen automaatti tunnistaa sen, ts. $A = L(M)$ jollain $M$.

Huomaa, että säännöllisyys on *kielen* ominaisuus, ei yksittäisen merkkijonon. Ei ole mielekästä kysyä yksittäisestä merkkijonosta, onko se säännöllinen.

(Mistä tahansa merkkijonosta $w \in \Sigma^*$ voidaan toki muodostaa yksialkioinen kieli $\{ w \} \subseteq \Sigma^*$, joka on selvästi säännöllinen.)



Äärellinen automaatti on hyvin rajoittunut laskennan malli. Esim. niinkin yksinkertainen kieli kuin $\{ 0^n 1^n \mid n \in \mathbb{N} \}$ (eli merkkijonot joissa on ensin jono nollia ja sitten saman verran ykkösiä) **ei ole** säännöllinen. (Tähän palataan.)

Äärelliseen automaattiin voidaan helposti lisätä muutakin tulostusta kuin pelkkä hyväksyminen tai hylkääminen. Tällaisen modifikaatiot eivät tämän kurssin kannalta toisi mitään oleellista uutta asiaan. Muissa yhteyksissä ne voivat kuitenkin olla hyödyllisiä.



## Äärellisen automaatin laatiminen [Sipser s. 41–43]

Halutaan laatia äärellinen automaatti, joka tunnistaa kielen

$$L_1 = \{ 0^n 1^m \mid n, m \in \mathbb{N} \} \cup \{ 1^n 0^m \mid n, m \in \mathbb{N} \}.$$

Siis joko nollat ovat ennen ykkösiä tai toisin päin; lukumäärällä ei ole väliä. Toisin sanoen vaihdoksia nollan ja ykkösen välillä saa olla korkeintaan 1.

Lähtökohta: seuraava algoritmi ratkaisee, kuuluuko syötejono kieleen $L_1$:

```
vaihtunut ← False
if syöte ei tyhjä    then Get(edellinen)
while syöte ei loppu
    do Get(nykyinen)
       if nykyinen ≠ edellinen
         then if not vaihtunut
                 then vaihtunut ← True    ▷ ensimmäinen vaihdos
                 else return False        ▷ toinen vaihdos
       edellinen ← nykyinen
return True
```

Tässä $nykyinen$ ja $edellinen$ ovat merkkejä ja Get lukee seuraavan syötemerkin.



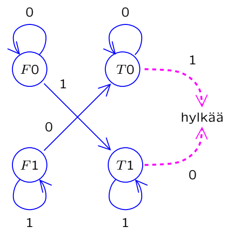
Äärellisen automaatin siirtymämekanismi huolehtii sellaisenaan uuden merkin lukemisesta.

Äärellisellä automaatilla ei ole muuta *muistia* kuin sen tilat. Kaikki algoritmin muuttuva tieto (jopa ohjelmalaskuri) pitää koodata tiloihin.

- Tässä ohjelmalaskuria ei tarvita kuin alussa ja lopussa (tästä enemmän alempana). Kukin silmukan kierros vastaa yhtä siirtymää.
- Algoritmissa on kaksi muuttujaa, joiden arvo pitää säilyttää seuraavalle kierrokselle. Kummallakin on kaksi arvoa, joten valitsemme neljä tilaa:
  $F0$: $vaihtunut$ = False ja $edellinen = 0$
  $F1$: $vaihtunut$ = False ja $edellinen = 1$
  $T0$: $vaihtunut$ = True ja $edellinen = 0$
  $T1$: $vaihtunut$ = True ja $edellinen = 1$

  Yleisemmin: tarvitaan oma tila jokaista mahdollista muuttujien arvojen kombinaatiota kohden.

Seuraava hahmotelma toteuttaa **while**-silmukan em. tilojen avulla:




Tätä pitää vielä täydentää, että

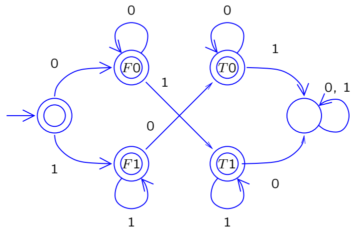
- alustukset ja
- hyväksyminen

saadaan hoidetuksi äärellinen automaatti -formalismin mukaisesti.



Alustukset hoituvat helposti lisäämällä alkutila.

Koska algoritmin "oletusarvo" on hyväksyminen, tehdään ensin *kaikista* tiloista hyväksyviä. Lisätään sitten "virhetila", johon mennään hylkäämistapauksessa:



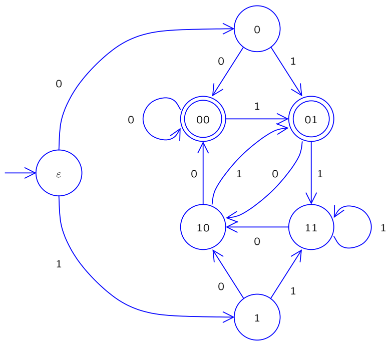



Toisena esimerkkinä olkoon

$$L_2 = \{\, b_1 \dots b_{n-2} 0 b_n \in \{\, 0, 1 \,\}^* \mid n \geq 2 \,\}$$

eli ainakin kahden pituiset bittijonot, joiden toiseksi viimeinen bitti on 0.

Nyt pitää muistaa kaksi viimeisintä merkkiä:




Tilojen nimet esittävät viimeksi luettuja kahta merkkiä.



Aiemmin mainittiin, että kieli

$$L_3 = \{\, 0^n 1^n \mid n \in \mathbb{N} \,\}$$

eli merkkijonot, joissa on ensin nollia ja sitten *sama määrä* ykkösiä ei ole säännöllinen.

Kielen tunnistavan automaatin tulisi muistaa nollien lukumäärä, jotta se voi varmistaa, että ykkösiä on yhtä monta.

- Ohjelmassa tähän tarvitaan vain yksi muuttuja, mutta muuttujan arvolla ei ole ylärajaa.
- Automaatissa tarvittaisiin oma tila jokaista $n$:n arvoa kohti, siis *ääretön* määrä tiloja.

Tämä on selitys $L_3$:n ei-säännöllisyydelle. Huomaa kuitenkin, että tämä ei ole formaali todistus.

## Tilasiirtymäjärjestelmien sovelluksia

Äärelliset automaatit sellaisenaan ovat hyödyllisiä *merkkijonojen* käsittelyssä, mistä kohta lisää.

Lisäämällä tilasiirtymiin satunnaisuutta saadaan *Markovin ketjut* eli satunnaisprosessit, joilla on äärellinen muisti. Perusversiossa Markovin ketjuihin ei tosin liity mitään syötettä. *Markovin piilomallit* (hidden Markov models) ovat lähempänä tässä esitettyjä äärellisiä automaatteja.

Tilasiirtymäjärjestelmiä käytetään (etenkin hajautettujen järjestelmien) *spesifioinnissa ja verifioinnissa*.



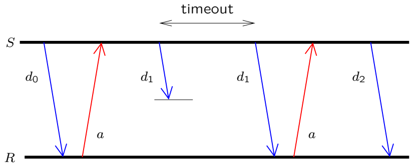
## Sovellusesimerkki: vuorottelevan bitin protokolla

Lähettäjä $S$ lähettää datapaketteja $d_0, d_1, d_2, \ldots$ vastaanottajalle $R$.

Paketti voi hukkua matkalla.

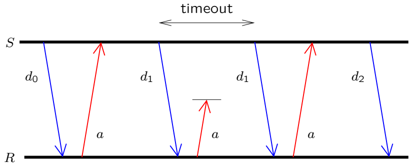
Seuraava datapaketti lähetetään vasta, kun on edelliseen saatu kuittaus $a$.

Jos kuittausta ei tule kohtuullisen ajan kuluessa, lähetys uusitaan:

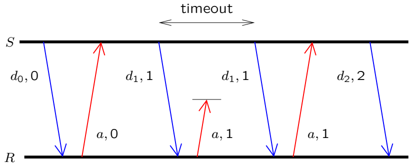





**Ongelma**: jos kuittaus hukkuu, $R$ voi saada duplikaattiviestin.

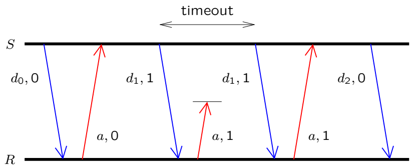



**Ratkaisu:** numeroidaan viestit ja kuittaukset

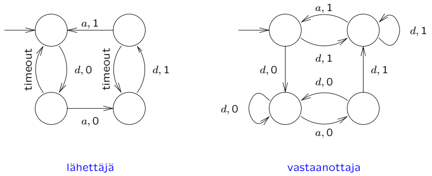





**Parannus:** itse asiassa riittää käyttää kahta numeroa (0 ja 1):




Lähettäjä ja vastaanottaja voidaan mallintaa tilasiirtymäjärjestelminä:




**Huom.** Esimerkin yksityiskohdat eivät ole tärkeitä tämän kurssin kannalta; ks. *Spesifioinnin ja verifioinnin perusteet*.

## **Hahmon etsiminen syötteestä** (johdatteleva esimerkki)

Unix-komennolla

`grep` *hahmo [ tiedosto ]*

voidaan etsiä hahmon esiintymiä tiedostosta (tai syötevirrasta):

```
$ grep Kisaveikot SM-tulokset.txt
$ ps aux | grep mmeikala
$ ps aux | grep '\(mmeikala\|tteikala\)'
```

Tässä siis haetaan

- SM-tuloksista rivit, joilla esiintyy seura Kisaveikot
- prosessilistasta ne rivit, joilla esiintyy käyttäjätunnus mmeikala, ja
- prosessilistasta ne rivit, joilla esiintyy käyttäjätunnus mmeikala tai tteikala.



Eräs idea `grep`-toiminnon toteuttamiseksi olisi seuraava:

1. Muodostetaan äärellinen automaatti, joka hyväksyy tasan sellaiset merkkijonot, joissa esiintyy *hahmo*.
2. Selataan tiedosto rivi kerrallaan käyttämällä tätä automaattia, ja tulostetaan hyväksytyt rivit.

Selausvaihe toimii nopeasti eikä edellytä tiedoston esiprosessointia, mikä on tässä tärkeää.

**Kysymys:** Kuinka monimutkaisia hahmoja tällä periaatteella voidaan käsitellä?

Esim. edellä muodostettiin hahmoista mmeikala ja tteikala uusi hahmo tai-operaattorilla " | ". Kuinka voimakkaat operaattorit voidaan siis sallia?



## **Operaatiot (säännöllisillä) kielillä** [Sipser s. 44–47]

Kuten muistetaan, kielet ovat merkkijonojoukkoja.

Niillä voidaan siis suorittaa normaaleja joukko-opillisia operaatioita, kuten *yhdiste*, *leikkaus* ja *komplementointi*: jos $A$ ja $B$ ovat kieliä, niin

- kieli $A \cup B$ koostuu niistä merkkijonoista, jotka kuuluvat *ainakin toiseen* kielistä $A$ ja $B$,
- kieli $A \cap B$ koostuu niistä merkkijonoista, jotka kuuluvat *sekä* kieleen $A$ *että* $B$, ja
- kieli $\overline{A}$ koostuu niistä merkkijonoista, jotka *eivät* kuulu kieleen $A$.

(Komplementin $\overline{A}$ määrittelyssä oletetaan, että puhutaan jonkin tietyn aakkoston $\Sigma$ merkkijonoista, jolloin $\overline{A} = \Sigma^* - A$. Yleensä aakkosto $\Sigma$ selviää asiayhteydestä.)



Nimenomaan merkkijonojoukoille on kätevää määritellä myös *konkatenaatio* ja *tähti*:

- kieli $A \circ B$ koostuu merkkijonoista $w$, jotka voidaan esittää muodossa $w = xy$ joillakin $x \in A$ ja $y \in B$ ja
- kieli $A^*$ koostuu merkkijonoista $w_1 \ldots w_k$, missä $k \geq 0$ ja $w_i \in A$ kaikilla $i$.

Toisin sanoen
$$A \circ B = \{ xy \mid x \in A \text{ ja } y \in B \}$$
ja
$$A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \cup \ldots,$$
missä $A^0 = \{ \varepsilon \}$ ja muuten $A^k$ on kielen $A$ konkatenaatio itsensä kanssa $k$ kertaa:
$$A^k = \underbrace{A \circ A \circ \ldots \circ A}_{k \text{ kertaa}}.$$

**Esimerkki** Tarkastellaan aakkoston $\{\,\mathsf{a},\ldots,\mathsf{z},0,\ldots,9\,\}$ kieliä $A = \{\,\mathsf{aa},\mathsf{bb}\,\}$ ja $B = \{\,01,02\,\}$. Nyt

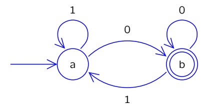
$$\begin{aligned} A \cup B &= \{\,\mathsf{aa},\mathsf{bb},01,02\,\} \\ A \circ B &= \{\,\mathsf{aa01},\mathsf{aa02},\mathsf{bb01},\mathsf{bb02}\,\} \\ A^* &= \{\,\varepsilon,\mathsf{aa},\mathsf{bb},\mathsf{aaaa},\mathsf{aabb},\mathsf{bbaa},\mathsf{bbbb}, \\ &\qquad \mathsf{aaaaaa},\mathsf{aaaabb},\mathsf{aabbaa},\mathsf{aabbbb},\mathsf{bbaaaa},\ldots\,\}\,. \end{aligned}$$

□

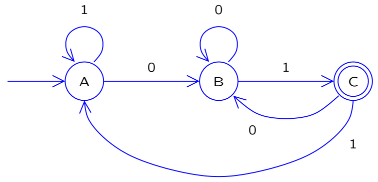
Seuraavana tavoitteenamme on osoittaa, että säännöllisten kielten luokka on *suljettu* operaatioiden $\cup$, $\circ$ ja $^*$ suhteen. Toisin sanoen jos $A$ ja $B$ ovat säännöllisiä, niin myös $A \cup B$, $A \circ B$ ja $A^*$ ovat. Tapaus $A \cup B$ on melko suoraviivainen, ja aloitamme siitä.



Aloitetaan konkreettisella esimerkillä. Seuraava automaatti hyväksyy merkkijonot, jotka loppuvat nollaan:




Toisaalta seuraava automaatti hyväksyy merkkijonot, jotka loppuvat 01:




Tilat on nimetty pienillä ja isoilla kirjaimilla jatkoa silmälläpitäen.

Voidaanko näistä automaateista muodostaa *yhdisteautomaatti*, joka hyväksyy merkkijonot, jotka päättyvät 01 *tai* 0?



Tarkastellaan esimerkkinä merkkijonoa 00101, jonka ensimmäinen automaatti hylkää:

$$\mathsf{a} \xrightarrow{0} \mathsf{b} \xrightarrow{0} \mathsf{b} \xrightarrow{1} \mathsf{a} \xrightarrow{0} \mathsf{b} \xrightarrow{1} \mathsf{a}$$

ja toinen hyväksyy:

$$\mathsf{A} \xrightarrow{0} \mathsf{B} \xrightarrow{0} \mathsf{B} \xrightarrow{1} \mathsf{C} \xrightarrow{0} \mathsf{B} \xrightarrow{1} \mathsf{C}.$$

Yhdisteautomaatin pitäisi simuloida kumpaakin laskentaa, koska etukäteen ei voi tietää, kumpi hyväksyy, jos kumpikaan.

Laskentojen laittaminen *peräkkäin* ei onnistu, koska äärellinen automaatti lukee kunkin syötemerkin vain kerran, minkä jälkeen se on mennyttä. Siis ensimmäisen simulaation jälkeen ei olisi enää mahdollista simuloida uudestaan samalla syötteellä.

Yhdisteautomaatin pitää siis simuloida laskentoja *rinnakkain*.



Rinnakkaisessa simuloinnissa täytyy pitää muistissa molempien automaattien sen hetkinen tila. Yhdisteautomaatissa on oma tilansa jokaista alkuperäisten automaattien tilojen yhdistelmää varten. Sen tilojen joukko on siis

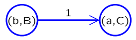
$$\{\,(\mathsf{a},\mathsf{A}),(\mathsf{a},\mathsf{B}),(\mathsf{a},\mathsf{C}),(\mathsf{b},\mathsf{A}),(\mathsf{b},\mathsf{B}),(\mathsf{b},\mathsf{C})\,\}$$

eli alkuperäisten tilajoukkojen karteesinen tulo $\{\,\mathsf{a},\mathsf{b}\,\} \times \{\,\mathsf{A},\mathsf{B},\mathsf{C}\,\}$.

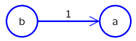
Laskenta jonolla 00101 on nyt:

$$(\mathsf{a},\mathsf{A}) \xrightarrow{0} (\mathsf{b},\mathsf{B}) \xrightarrow{0} (\mathsf{b},\mathsf{B}) \xrightarrow{1} (\mathsf{a},\mathsf{C}) \xrightarrow{0} (\mathsf{b},\mathsf{B}) \xrightarrow{1} (\mathsf{a},\mathsf{C})$$

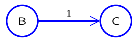
Siirtymien suhteen esim.




tarkoittaa, että alkuperäisissä automaateissa on siirtymät




ja




Huomaa, että kummassakin on sama merkki 1.

Graafisesti tämän voi esittää laittamalla yhdisteautomaatin tilat taulukoksi, jossa

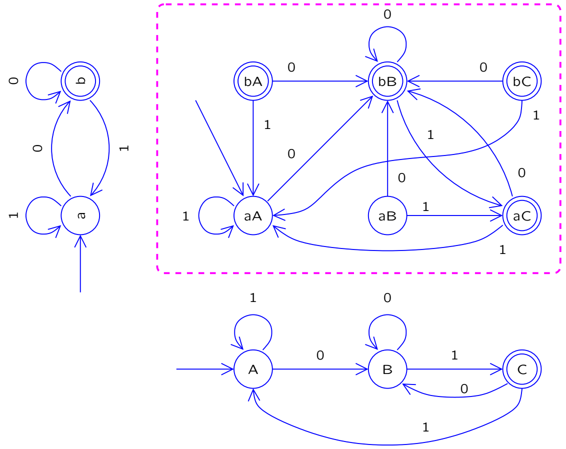
- ensimmäisen automaatin tilat vastaavat rivejä
- toisen automaatin tilat vastaavat sarakkeita.

Vastaavasti ensimmäinen automaatti antaa siirtymien pystykomponentin ja toinen automaatti vaakakomponentin.

Alkutilaksi tulee $(a, A)$ eli alkuperäisten automaattien alkutilojen kombinaatio.

Koska kyseessä on *unioni*, hyväksyviä tiloja ovat kaikki, joissa esiintyy $b$ tai $C$ eli ainakin toisen alkuperäisen automaatin hyväksyvä tila.








Esitetään edellinen konstruktio yleisemmässä muodossa.

**Lause 1.1:** [Sipser Thm. 1.25] Jos kielet $A$ ja $B$ ovat säännöllisiä, niin myös $A \cup B$ on.

**Todistus:** Oletetaan siis, että $A = L(M_1)$ ja $B = L(M_2)$ äärellisillä automaateilla $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ ja $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. Muodostamme äärellisen automaatin

$$M_\cup = (Q_\cup, \Sigma, \delta_\cup, q_\cup, F_\cup),$$

jolla $L(M_\cup) = A \cup B$.

Koska $A \cup B$ on kieli samassa aakkostossa $\Sigma$ kuin $A$ ja $B$, niin

- aakkosto $\Sigma$ pysyy samana.

Edellä esitetyn mukaan yhdisteautomaatin tilat ovat pareja $(q', q'')$, missä $q'$ ja $q''$ ovat automaattien $M_1$ ja $M_2$ tiloja; siis

- $Q_\cup = Q_1 \times Q_2$.



Edelleen

- alkutilaksi tulee alkuperäisten alkutilojen kombinaatio $q_\cup = (q_1, q_2)$.

Tila on hyväksyvä, jos se vastaa ainakin toisen alkuperäisen automaatin hyväksyvää tilaa, eli

- $F_\cup = \{\, (r, s) \in Q_1 \times Q_2 \mid r \in F_1$ tai $s \in F_2 \,\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Siirtymäfunktio määritellään siten, että automaatti $M_1$ kertoo siirtymät ensimmäisen ja $M_2$ toisen komponentin suhteen:

- $\delta_\cup((r, s), a) = (\delta_1(r, a), \delta_2(s, a))$, kun $r \in Q_1$, $s \in Q_2$ ja $a \in \Sigma$.

Konstruktion perusteella on selvää, että $L(M_\cup) = L(M_1) \cup L(M_2)$ kuten haluttiin. □

## Reunahuomautus (jos tuntuu sekavalta, niin sivuuta)

Jos ei halua hyväksyä edellisen konstruktion olevan "selvästi" toimiva, asian voi perustella yksityiskohtaisemminkin.

Todistetaan ensin induktiolla merkkijonon $w$ pituuden suhteen, että $\delta^*_\cup((r,s),w) = (\delta^*_1(r,w), \delta^*_2(s,w))$, missä $\delta^*$ on siirtymäfunktion $\delta$ yleistys kuten kalvoilla 26–27.

Tapaus $|w| = 0$: Siis $w = \varepsilon$, ja suoraan määritelmistä

$$\delta^*_\cup((r,s),\varepsilon) = (r,s) = (\delta^*_1(r,\varepsilon), \delta^*_2(s,\varepsilon)).$$



Tapaus $|w| = n+1$: Oletetaan, että väite $\delta^*_\cup((r,s),w) = (\delta^*_1(r,w), \delta^*_2(s,w))$ pätee, kun $|w| = n$. Kun $|w| = n+1$, voidaan kirjoittaa $w = av$, missä $|v| = n$ ja $a \in \Sigma$. Tällöin

$$\begin{aligned}
\delta^*_\cup((r,s),w) &= \delta^*_\cup((r,s),av) \\
&= \delta^*_\cup(\delta_\cup((r,s),a),v) \\
&= \delta^*_\cup((\delta_1(r,a),\delta_2(s,a)),v) \\
&\overset{\text{ind.ol.}}{=} (\delta^*_1(\delta_1(r,a),v), \delta^*_2(\delta_2(s,a),v)) \\
&= (\delta^*_1(r,av), \delta^*_2(s,av)) \\
&= (\delta^*_1(r,w), \delta^*_2(s,w))
\end{aligned}$$

kuten haluttiin.



Nyt

$$\begin{aligned}
w \in L(M_\cup) &\Leftrightarrow \delta^*_\cup(q_\cup, w) \in F_\cup \\
&\Leftrightarrow \delta^*_\cup((q_1,q_2),w) = (r,s) \quad \text{missä } r \in F_1 \text{ tai } s \in F_2 \\
&\Leftrightarrow \delta^*_1(q_1,w) \in F_1 \quad \text{tai} \quad \delta^*_2(q_2,w) \in F_2 \\
&\Leftrightarrow w \in L(M_1) \quad \text{tai} \quad w \in L(M_2)
\end{aligned}$$

eli

$$L(M_\cup) = L(M_1) \cup L(M_2).$$

Tämäntyyppisiä induktiotodistuksia tarvitaan monimutkaisempien konstruktioiden oikeellisuudelle. Niillä voidaan tarkistaa, ettei konstruktioon jäänyt virheitä.

Yleensä (kuten tässäkin) induktio ei kuitenkaan tuo mitään lisävalaistusta konstruktion ideaan, vaan helposti peittää sen alleen. Ensin pitää ymmärtää konstruktion idea, minkä jälkeen induktiotodistuksen laatiminen on (toisinaan vaativa) tekniikkaharjoitus.

Siksi emme jatkossa vie oikeellisuustodistuksia näin formaaleiksi, ellei konstruktion toimivuus ole oikeasti epäilmeinen.

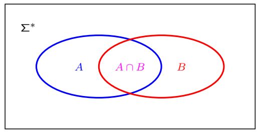
(Reunahuomautus loppuu.)



## Varoitus: älä tee näin

Eräässä tentissä pyydettiin todistamaan, että kahden säännöllisen kielen $A$ ja $B$ leikkaus $A \cap B$ on säännöllinen.

Tyypillinen virheellinen todistusyritys oli seuraavanlainen:
Oletuksen mukaan joukkojen $A$ ja $B$ merkkijonot ovat säännöllisiä.
Siis myös kaikki joukon $A \cap B$ merkkijonot ovat säännöllisiä (ks. kuva)
Siis $A \cap B$ on säännöllinen.




Tässä **ei ole mitään järkeä**, koska ei ole olemassa sellaista kuin "säännöllinen merkkijono". Säännöllisyys on *kielen* eli merkkijono*joukon* ominaisuus.

(Varoitus loppuu.)

Oikea tapa ratkaista tehtävä on soveltaa edellisen lauseen konstruktiota (tai kohta esitettäviä vaihtoehtoisia tekniikoita).