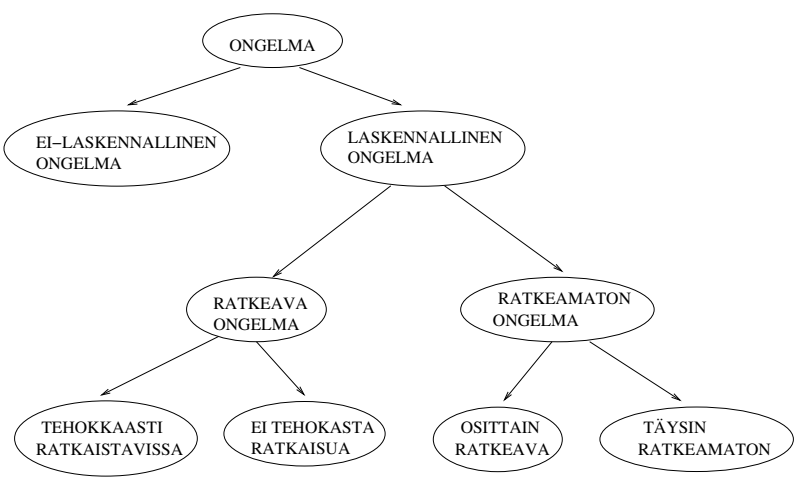
Luku 2

Laskennalliset ongelmat

Ongelmat voidaan jakaa yleisellä tasolla kahteen luokkaan: *laskennallisiin* ja *laskennallisiin ongelmiin*. Laskennallisia ongelmia ovat kaikki sellaiset ohgelmat, jotka voidaan mallintaa ratkaistaviksi digitaalisella tietokoneella. Huomaa, että laskennalisen ongelman ei silti tarvitse olla ratkeava, ts. voi olla, ettei periaatteessakaan ole mahdollista laatia tietokoneohjelmaa, joka ratkaisisi ongelman. Esimerkkejä laskennallisista ongelmista ovat vaikkapa kokonaislukujen kertolasku, kirjastokortiston aakkostaminen, yrityksen palkanlaskenta, yliopistollisen kurssin kurssitietojen ylläpito. Sen sijaan sellaiset ongelmat kuin "Kuinka opettaa sodat?", "Onko oikein huijata tentissä?" ovat luonteeltaan ei-laskennallisia. Jatkossa viittaamme sanalla "ongelma" aina laskennalliseen ongelmaan.



Kuva 2.1: Ongelmien luokittelu.

Laskennallisen ngelman ratkaiseva ohjelma on sen yksi esitystapa, mutta kaikille ongelmille emme voi tai osaa laatia ratkaisevaa ohjelmaa. Yleisempi ja abstraktimpi esitystapa helpottaa myös ongelman analysointia, ja voimme analysoida sen ratkeavuuden ja vaikeuden, ennen kuin kukaan on keksinyt ongelman ratkaisevaa ohjelmaa. Tästä syystä laskennanteoriassa ongelmia mallinnetaan formaaleilla malleilla. Formaalien mallien ei kuitenkaan tarvitse olla tylsiä, kuten seuraava esimerkki osoittaa.

Luvun loppupuolella tarkastelemme laskennallisten ongelmien formalisointia yleensä sekä erästä tärkeää ongelmien alaluokkaa, *päätösongelmia*, ja niitä vastaavia *formaaleja kieliä*. Lopuksi annamme esimerkin ratkeamattomasta ongelmasta.

2.1 Ongelma: MIU-järjestelmä

Kissastanian logiikakoulussa on tunnin aiheena MIU-järjestelmä, jonka kaikki lauseet koostuvat kolmesta kirjaimesta M, I ja U. Järjestelmässä on vain yksi aksiooma MI. Uusia lauseita (teoreemoja) saa muodostaa edellisistä lauseista (aksioomiasta tai teoreemoista) seuraavilla säännöillä:

1. Jos merkkijonon viimeinen kirjain on I, saat lisätä U:n loppuun.
2. Jos sinulla on Mx (missä x voi olla mikä tahansa merkkijono, myös tyhjä merkkijono), saat päätellä merkkijonon Mxx.
3. Jos jossain merkkijonossa esiintyy III, saat korvata sen U:lla.
4. Jos merkkijonossa esiintyy UU, sen saa pudottaa pois merkkijonosta.

Sääntöjä saa soveltaa vapaasti aina, kun ne sopivat aksioomaan tai jo johdettuihin teoreemoihin, mutta mitään muuta ei saa tehdä (siksi järjestelmää kutsutaan formaaliksi).

Kissakoululaisten tehtävänä on johtaa aksioomasta MI teoreema MIU. Osaatko suorittaa päättelyn? Entä osaatko johtaa aksioomasta MI teoreeman MU?!

MIU-järjestelmää voidaan siis ajatella eräänlaisena kissojen kielipelinä. Käytössä on *aakkosto* $\Sigma = \{M, I, U\}$ ja kielioppi, jonka mukaan voidaan muodostaa sanoja. Esitetään sananmuodostussäännöt hieman formaalimmin:

$xI \to xU$
$Mx \to Mxx$
$xIIIy \to xUy$
$xUUy \to xy$,

missä x ja y voivat olla mitätahansa merkkijonoja (myös tyhjä merkkijono).

Tarkastellaan aakkoston kaikkien mahdollisten merkkijonojen joukkoa Σ^*, joka koostuu merkkijonoista $\{\epsilon, M, I, U, MM, MI, MU, IM, II, IU, UM, UI, UU, MMM, MMI, MMU, MIM, ...\}$ (ϵ = tyhjä merkkijono). Nyt voidaan esittää erilaisia kysymyksiä aakkoston merkkijonoista. Esim. ongelma $\pi_1(x)$: "Mitä merkkijonoja voit tuottaa annetusta merkkijonosta x järjestelmän säännöillä?" tai $\pi_2(x)$: "Voitko tuottaa merkkijonon x MI:stä järjestelmän säännöillä?" Ensimmäisen kysymyksen vastaus on joukko merkkijonoja, itse asiassa Σ^*:n osajoukko (tai mahdollisesti tyhjä joukko), kun taas jälkimmäisen kysymyksen vastaus on "Kyllä" tai "Ei". Tällaisia kyllä/ei-ongelmia kutsutaan *päätösongelmiksi*. Formaalisti voidaan määritellä päätösongelma π kuvauksena $\pi : \Sigma^* \to \{0, 1\}$. Se liittää siis jokaiseen aakkoston merkkijonoon joko vastauksen 1 tai 0.

Toisaalta voimme kysyä, mitä kaikkia Σ^*:n merkkijonoja päätösongelma π_A hyväksyy? (T.s. mille x $\pi(x) = 1$ tai käänteisrelaatio $\pi^{-1}(1) = x$?) Vastausjoukkoa A kutsutaan *formaaliksi kieleksi* ja vastaavaa päätösongelmaa π_A kielen A tunnistusongelmaksi.

Tehtävä: Mistä sanoista seuraavat kielet koostuvat?

- Kaikki M:stä johdettavat merkkijonot.
- Kaikki U:sta johdettavat merkkijonot.
- Kaikki MU:sta johdettavat merkkijonot.

(MIU-järjestelmän idea on alkujaan Hofstadterin hauskasta kirjasta Gödel, Escher, Bach.)

2.2 Formalisointi

- ongelmalla potentiaalisesti ääretön joukko *tapauksia* ("syötteitä")

- ongelman ratkaisu on *algoritmi*, joka liittää kuhunkin tapaukseen sen oikean *vastauksen* ("tulosteen").
- esim. kokonaislukujen kertolaskuongelma
  - tapaukset: kaikki mahdolliset kokonaislukuparit
  - annettuun tapaukseen liittyvä vastaus: lukuparin tulo
  - ongelman ratkaisu: mikä tahansa yleinen kertolaskualgoritmi.
- kunkin yksittäisen tapauksen ja sen vastauksen oltava *äärellisesti esitettäviä.*
- Laskennallinen ongelma = *kuvaus äärellisesti esitettävien tapausten joukosta äärellisesti esitettävien vastausten joukkoon*

Äärellinen esitys

- kaikki tietokoneen käsittelemä tieto on viime kädessä voitava koodata bittijonoiksi.
- luontevaa sallia koodaukseen käytettävän myös muita merkkejä kuin 0 ja 1 (muut merkit voidaan tietenkin tarvittaessa edelleen esittää bittijonoina).
- Määr. "äärellinen esitys" = jonkin aakkoston *merkkijono.*
- esim. em. kertolaskun syöte (kaksi kokonaislukua) voitaisiin esittää merkkijonona $x\#y$ tai $mul(x, y)$, missä x ja y ovat kokonaislukujen merkkijonoesitykset

Merkkijonoihin liittyviä peruskäsitteitä ja merkintöjä

- *Aakkosto* on äärellinen, epätyhjä joukko alkeis*merkkejä* t. *symboleita.* esim. *binääriaakkosto* $\{0, 1\}$ ja *latinalainen aakkosto* {A,B,...,Z}.
- *Merkkijono* on äärellinen järjestetty jono jonkin aakkoston merkkejä. Esim. "01001" ja "000" ovat binääriaakkoston merkkijonoja, ja "LTE" ja "XYZZY" ovat latinalaisen aakkoston merkkijonoja.
- Merkkijonon x *pituus* on siihen sisältyvien merkkien määrä. Merk. $|x|$. Esim.

$$|01001| = |\mathrm{XYZZY}| = 5, \qquad |000| = |\mathrm{OTE}| = 3.$$

- *Tyhjä merkkijonon* ϵ pituus on $|\epsilon| = 0$.

- Merkkijonojen välinen perusoperaatio on *katenaatio* eli jonojen peräkkäin kirjoittaminen. (Katenaation operaatiomerkkinä käytetään joskus selkeyden lisäämiseksi symbolia $\hat{}$.) Esimerkkejä:

  (i) KALA^KUKKO = KALAKUKKO;

  (ii) jos $x = 00$ ja $y = 11$, niin $xy = 0011$ ja $yx = 1100$;

  (iii) kaikilla x on $x\epsilon = \epsilon x = x$;

  (iv) kaikilla x, y on $|xy| = |x| + |y|$.

- Aakkoston Σ kaikkien merkkijonojen joukko on Σ^*. Esim. $\Sigma = \{0, 1\}$, $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, \ldots\}$.

Päätösongelmat ja formaalit kielet

- yleisesti laskennallinen ongelma π on kuvaus

$$\pi : \Sigma^* \to \Gamma^*,$$

  missä Σ ja Γ ovat aakkostoja

- päätösongelmat ovat tärkeä laskennallisten ongelmien aliluokka, jossa kunkin ongelman tapauksen vastaus on "kyllä" tai "ei"

- formaalisti ongelma on muotoa $\pi : \Sigma^* \to \{0, 1\}$.

- esimerkiksi päätösongelma "onko annettu luku alkuluku?" voidaan esittää aakkoston $\Sigma = \{0, 1, 2, \ldots, 9\}$ kuvauksena

$$\pi : \Sigma^* \to \{0, 1\}, \qquad \pi(x) = \begin{cases} 1, & \text{jos } x \text{ on alkuluku;} \\ 0, & \text{jos } x \text{ ei ole alkuluku.} \end{cases}$$

- Jokaista päätösongelmaa $\pi : \Sigma^* \to \{0, 1\}$ vastaa merkkijonojoukko

$$A_\pi = \{x \in \Sigma^* \mid \pi(x) = 1\},$$

  so. niiden ongelman tapausten joukko, joihin vastaus on "kyllä"

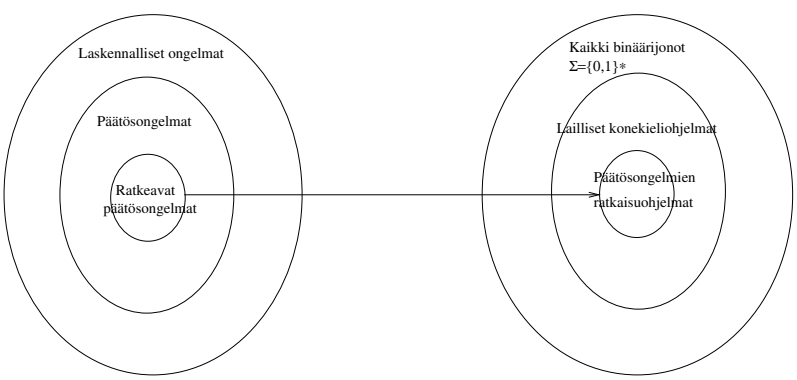
- jokaista merkkijonojoukkoa $A \subseteq \Sigma^*$ vastaa päätösongelma

$$\pi_A : \Sigma^* \to \{0, 1\}, \qquad \pi_A(x) = \begin{cases} 1, & \text{jos } x \in A; \\ 0, & \text{jos } x \notin A. \end{cases}$$

- aakkoston Σ *(formaali) kieli* (engl. formal language) = mielivaltainen merkkijonojoukko $A \subseteq \Sigma^*$
- kielen A *tunnistusongelma* (engl. recognition problem) = merkkijonojoukkoon liittyvä päätösongelmaa π_A
- ts. formaali kieli $\sim$ päätösongelma

2.3 Laskennallisten ongelmien ratkeavuus

- sanotaan, että ohjelma P *ratkaisee* laskentaongelman π, jos kullakin syötteellä x ohjelma P laskee ja tulostaa arvon $\pi(x)$.
- Voidaanko kaikki mahdolliset laskentaongelmat ratkaista ohjelmilla?
- Osoittautuu, ettei voida, sillä kaikkien merkkijonojen (millä tahansa ohjelmointikielellä mahdollisten ohjelmien) joukko on numeroituva, mutta kaikkien päätösongelmien joukko on ylinumeroituva ($\aleph_1$ vierasta ei voi majoittaa ω:aan huoneeseen).
- Laskentaongelmia on siis tietyssä mielessä "enemmän" kuin niiden mahdollisia ratkaisuja, ja siksi *millään ohjelmointikielellä ei voida laatia ratkaisuja kaikille laskentaongelmille.*



Kuva 2.2: Vain pieni osa päätösongelmista on tietokoneella ratkeavia.

Lause 1 Minkä tahansa aakkoston Σ merkkijonojen joukko Σ^* on numeroituvasti ääretön.

Todistus: Olkoon $\Sigma = \{a_1, a_2, ..., a_n\}$. Kiinnitetään merkeille "aakkosjärjestys", esim. $a_1 < a_2 < ... < a_n$.

Joukon Σ^* merkkijonot voidaan järjestää seuraavasti (*kanoniseen järjestykseen*):

1. ensin luetellaan 0:n mittaiset merkkijonot ($= \epsilon$), sitten 1:n mittaiset ($= a_1, a_2, \ldots, a_n$), sitten 2:n mittaiset jne.;
2. kunkin pituusryhmän sisällä merkkijonot luetellaan aakkosjärjestyksessä.

- jokaiseen luonnolliseen lukuun n voidaan siis liittää Σ^*:n merkkijono ja päinvastoin $\rightarrow \Sigma^*$ on numeroituva.

Vaadittu bijektio $f : \mathbb{N} \rightarrow \Sigma^*$ on:

$$\begin{array}{rcl}
0 & \mapsto & \epsilon \\
1 & \mapsto & a_1 \\
2 & \mapsto & a_2 \\
\vdots & & \vdots \\
n & \mapsto & a_n \\
n+1 & \mapsto & a_1 a_1 \\
n+2 & \mapsto & a_1 a_2 \\
\vdots & & \vdots \\
2n & \mapsto & a_1 a_n \\
2n+1 & \mapsto & a_2 a_1 \\
\vdots & & \vdots \\
3n & \mapsto & a_2 a_n \\
\vdots & & \vdots \\
n^2+n & \mapsto & a_n a_n \\
n^2+n+1 & \mapsto & a_1 a_1 a_1 \\
n^2+n+2 & \mapsto & a_1 a_1 a_2 \\
\vdots & & \vdots
\end{array}$$

$\square$

Lause 2 Minkä tahansa aakkoston Σ päätösongelmien joukko on ylinumeroituva.

**Todistus:* (Cantorin diagonaaliargumentti.)

Merkitään kaikkien Σ:n päätösongelmien kokoelmaa Π:llä:

$$\Pi = \{\pi \mid \pi \text{ on kuvaus } \Sigma^* \to \{0,1\}\}.$$

Vastaväite: Oletetaan, että Π on numeroituva, so. on olemassa numerointi

$$\Pi = \{\pi_0, \pi_1, \pi_2, \ldots\}.$$

Olkoot Σ^*:n merkkijonot kanonisessa järjestyksessä lueteltuina $x_0, x_1, x_2, \ldots$.

Muodostetaan uusi päätösongelma $\hat{\pi}$:

$$\hat{\pi} : \Sigma^* \to \{0,1\}, \qquad \hat{\pi}(x) = \begin{cases} 1, & \text{jos } \pi_i(x_i) = 0; \\ 0, & \text{jos } \pi_i(x_i) = 1. \end{cases}$$

Koska $\hat{\pi} \in \Pi$,niin $\hat{\pi} = \pi_k$ jollakin $k \in \mathbb{N}$.
Tällöin

$$\hat{\pi}(x_k) = \begin{cases} 1, & \text{jos } \pi_k(x_k) = \hat{\pi}(x_k) = 0; \\ 0, & \text{jos } \pi_k(x_k) = \hat{\pi}(x_k) = 1. \end{cases}$$

RISTIRIITA. Siis oletus, että joukko Π on numeroituva, on väärä. $\square$

$\hat{\pi}$ $\searrow$	π_0	π_1	π_2	π_3	$\cdots$
x_0	$\overset{1}{\cancel{0}}$	0	0	1	$\cdots$
x_1	0	$\overset{0}{\cancel{1}}$	0	0	$\cdots$
x_2	1	1	$\overset{0}{\cancel{1}}$	1	$\cdots$
x_3	0	0	0	$\overset{1}{\cancel{0}}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- Käy kuin Valehtelijan paradoksissa! (”Puhuuko x totta, jos x sanoo: ’Minä valehtelen’?”)

- kaikista laskentaongelmista voidaan esimerkiksi C-ohjelmilla ratkaista vain häviävän pieni osa: ylinumeroituvan joukon numeroituva osajoukko.
- sama pätee kaikilla ohjelmointikielillä, sillä kaikki "riittävän vahvat" ohjelmointikielet määrittävät täsmälleen saman ratkeavien ongelmien luokan (ns. Churchin–Turingin teesi).
- useimmat laskennalliset ongelmat ovat *absoluuttisesti ratkeamattomia.*
- ratkeamattomat ongelmat käsittävät myös mielenkiintoisia/käytännöllisiä ongelmia
- esim. *pysähtymisongelma*: annettu ohjelma P ja sen syöte x; pääteltävä pysähtyykö P:n laskenta syötteellä x, vai jääkö se ikuiseen silmukkaan.

2.4 Ekskursio: pysähtymisongelman ratkeamattomuus

Pysähtymisongelman C-tulkinta on: "Ei ole olemassa totaalista (aina pysähtyvää) C-ohjelmaa, joka ratkaisisi, pysähtyykö annettu C-ohjelma P annetulla syötteellä w".

Oletetaan, että voitaisiin kirjoittaa totaalinen C-funktio

int H(char *p, char *w),

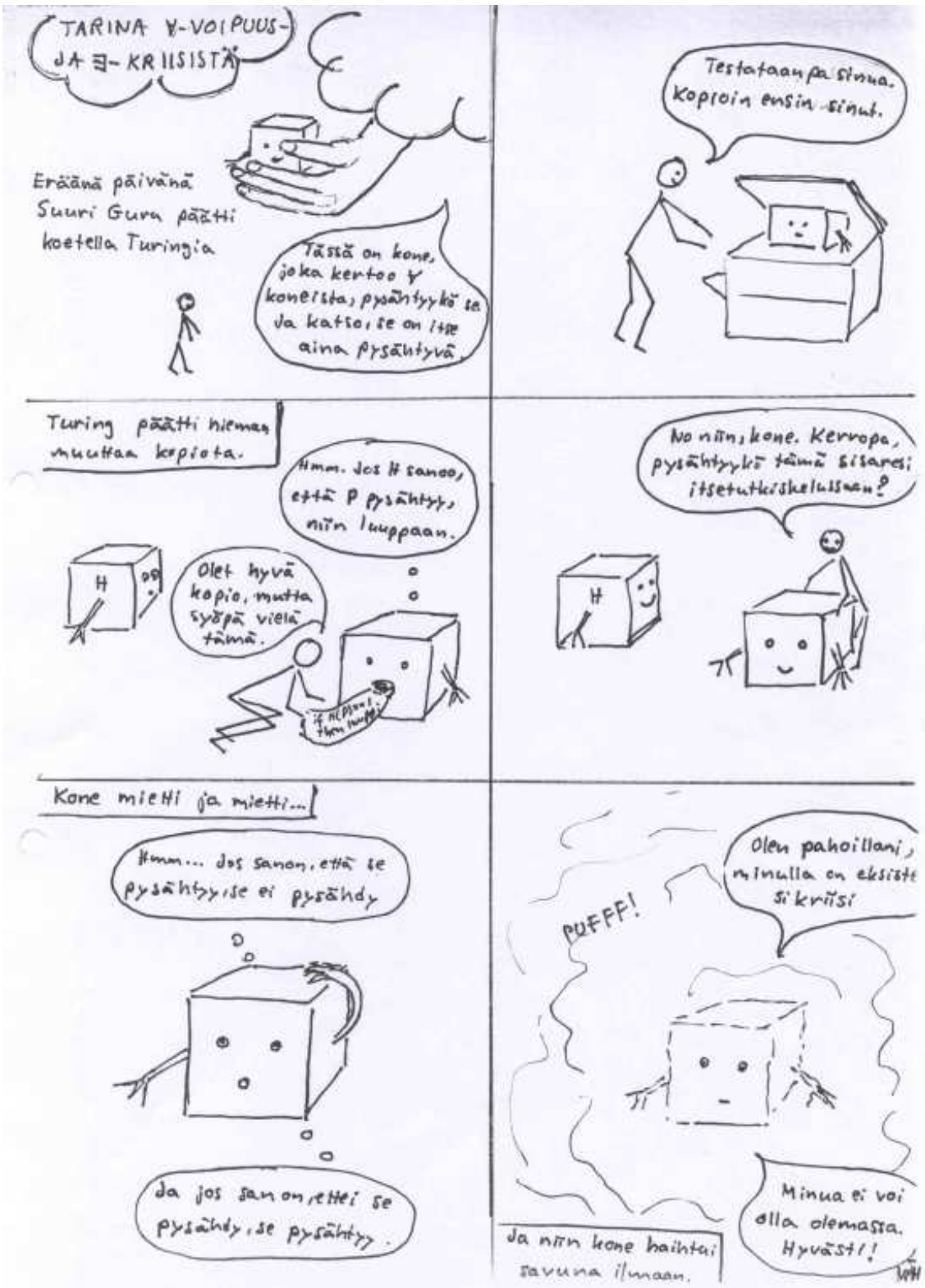
joka saa arvon $\mathbf{1}$, jos merkkijonoparametrin p esittämä funktio pysähtyy syötteellä w, ja $\mathbf{0}$ muuten. Kirjoitetaan tämän perusteella toinen C-funktio $\hat{H}$:

```
void Ĥ(char *p){
        if H(p,p) while (1) ;
    }
```

Merkitään edellä kuvattua funktion $\hat{H}$ ohjelmatekstiä $\hat{h}$:lla ja tarkastellaan funktion $\hat{H}$ laskentaa tällä omalla kuvauksellaaan. Saadaan ristiriita:

$$\hat{H}(\hat{h}) \text{ pysähtyy} \quad \Leftrightarrow \quad H(\hat{h},\hat{h}) = \mathbf{0} \quad \Leftrightarrow \quad \hat{H}(\hat{h}) \text{ ei pysähdy.}$$

Ristiriidasta seuraa, että oletettua totaalista pysähtymisentestausohjelmaa H ei voi olla olemassa.


TARINA ∀-VOIPUUS- JA ∃-KRIISISTÄ
Eräänä päivänä Suuri Guru päätti koetella Turingia
Tässä on kone, joka kertoo ∀ koneista, pysähtyykö se ja katso, se on itse aina pysähtyvä.
Testataan pa sinua. Kopioin ensin sinut.
Turing päätti hieman muuttaa kopiota.
Hmm. Jos H sanoo, että P pysähtyy, niin luuppaan.
H
Olet hyvä kopio, mutta syöpä vielä tämä.
Jos H pysähtyy, tämä luuppaa
No niin, kone. Kerro pa, pysähtyykö tämä sisäresi itsetutkiskelussaan?
H
Kone mietti ja mietti...
Hmm... Jos sanon, että se pysähtyy, se ei pysähdy
Ja jos sanon, ettei se pysähdy, se pysähtyy.
PUFFF!
Olen pahoillani, minulla on eksistenssikriisi
Minua ei voi olla olemassa. Hyvästi!
Ja niin kone haihtui savuna ilmaan.

2.5 Liite: Vakiintuneita merkintätapoja

- Aakkostot: Σ, Γ, ... (isoja kreikkalaisia kirjaimia).
  esim. binääriaakkosto $\Sigma = \{0, 1\}$.
- Aakkoston koko (tai yleisemmin joukon mahtavuus): $|\Sigma|$.
- Alkeismerkit: a, b, c, ... (pieniä alkupään latinalaisia kirjaimia).
  esim. olkoon $\Sigma = \{a_1, \ldots, a_n\}$ aakkosto; tällöin $|\Sigma| = n$.
- Merkkijonot: u, v, w, x, y, ... (pieniä loppupään latinalaisia kirjaimia).
- Merkkijonojen katenaatio: $x\hat{\ }y$ tai vain xy.
- Merkkijonon pituus: $|x|$. *Esimerkkejä:*
  (i) $|abc| = 3$;
  (ii) olkoon $x = a_1 \ldots a_m$, $y = b_1 \ldots b_n$; tällöin $|xy| = m + n$.
- Tyhjä merkkijono: ϵ.
- Merkkijono, jossa on n kappaletta merkkiä a: a^n. *Esimerkkejä:*
  (i) $a^n = \underbrace{aa \ldots a}_{n \text{ kpl}}$;
  (ii) $|a^i b^j c^k| = i + j + k$.
- Merkkijonon x toisto k kertaa: x^k. *Esimerkkejä:*
  (i) $(ab)^2 = abab$;
  (ii) $|x^k| = k|x|$.
- Aakkoston Σ kaikkien merkkijonojen joukko: Σ^*.
  Esimerkki: $\{a, b\}^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$.

2.6 Tehtäviä lukuun 1

1. Lue satu päätösongelmista
   `http://www.cs.joensuu.fi/pages/whamalai/tepe04/satu.htm` ja täydennä se loppuun!

2. Tarkastellaan taas Kissastanian logiikkakoulua. Tällä kertaa tunnin aiheena on hieman monimutkaisempi MIAU-järjestelmä, joka koostuu seuraavista säännöistä:

   xUAx → xAUy
   xUUx → xIUy
   x → MxM
   x → xUI
   xx → x
   xI → xUA,

   missä x ja y voivat olla mitätahansa merkkijonoja.

   Tehtävänä on osoittaa, että tyhjästäkin (merkkijonosta) voi syntyä kunnon naukaisu (MIAU) järjestelmän säännöillä!

3. Tarkastellaan aakkostoa $\Sigma = \{m, i, u\}$. Määritellään aakkoston "potenssit" seuraavasti:

   $\Sigma^0 = \{\epsilon\}$ (tyhjä merkkijono)
   $\Sigma^{k+1} = \Sigma \times \Sigma^k = \{ax | a \in \Sigma \text{ ja } x \in \Sigma^k\}$.

   Esim. $\Sigma^1 = \{m, i, u\}$, $\Sigma^2 = \{mm, mi, mu, im, ii, iu, um, ui, uu\}$. Montako alkiota ("sanaa") on Σ^n:ssä? Entä montako sanaa on koko kielessä $\Sigma^* = \bigcup_{i=0}{}^{\infty}$?