5.3.3. Yleistetty Modus Ponens

Luvun 5.3.2 esimerkin olennainen kulku oli päätellä sopivia atomikaavoja toisista atomikaavoista Modus Ponensilla.

Tämä on niin yleinen päättelytapa, että kannattaa oikaista sen mutkia erityisellä päättelysäännöllä *yleistetty (generalised) Modus Ponens:*

$$\frac{p_1' \quad \ldots \quad p_n' \quad p_1 \land \ldots \land p_n \Rightarrow q}{\textsc{Subst}(\theta, q)} GMP \tag{56}$$

missä

- jokainen p_i, p_i' ja q on atomikaava
- θ on sellainen sidonta, jolla $\textsc{Subst}(\theta, p_i)$ ja $\textsc{Subst}(\theta, p_i')$ ovat *syntaktisesti sama* kaava kaikilla $1 \leq i \leq n$.

  Nämä sidonnat ovat yksittäisten muuttujasymbolien sidontojen x_i/t_i joukkoja. Lisäksi vaaditaan, ettei niissä ole kehiä: esimerkiksi $\{x/f(y), y/x\}$ ei ole luvallinen sidonta, koska $x = f(y) = f(x) = f(f(y)) = f(f(x)) = \ldots$ olisi kehä joka johtaisi äärettömän pitkään termiin.

  Minimaalinen mutta silti riittävä sidonta voidaan laskea *samastusalgoritmilla (unification algorithm) 13.*

Yleistetty Modus Ponens (56) on erityisen luonteva luvussa 5.1 mainittujen *Hornin lauseiden predikaattiloogisille muodoille*. Nämä ovat kaavoja

$$\forall \vec{x}.p_1 \land \ldots \land p_n \Rightarrow q \tag{57}$$

missä universaalikvantifionti $\forall \vec{x}$ (joka jätetään usein kirjoittamatta) käsittää täsmälleen ne muuttujasymbolit, jotka tässä implikaatiossa esiintyvät vapaina.

Tämä kvantifiointi huolehtii siitä, että säännön ehtojen muuttujat ovat aina *erillisiä*. Samoin johtopäätöksen muuttujat kannattaa pitää erillään ehtojen muuttujista. (Vrt. vapaiden muuttujien ongelma luvussa 5.2.2.)


```
function UNIFY(x, y) returns a substitution to make x and y identical, if possible
  UNIFY-INTERNAL(x, y, {})
--------------------------------------------------------------------------------
function UNIFY-INTERNAL(x, y, θ) returns a substitution to make x and y identical (given θ)
  inputs: x, a variable, constant, list, or compound
          y, a variable, constant, list, or compound
          θ, the substitution built up so far

  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
     return UNIFY-INTERNAL(ARGS[x], ARGS[y], UNIFY-INTERNAL(OP[x], OP[y], θ)
  else if LIST?(x) and LIST?(y) then
     return UNIFY-INTERNAL(REST[x], REST[y], UNIFY-INTERNAL(FIRST[x], FIRST[y], θ))
  else return failure
--------------------------------------------------------------------------------
function UNIFY-VAR(var, x, θ) returns a substitution
  inputs: var, a variable
          x, any expression
          θ, the substitution built up so far

  if {var/val} ∈ θ
     then return UNIFY-INTERNAL(val, x, θ)
  else if {x/val} ∈ θ
     then return UNIFY-INTERNAL(var, val, θ)
  else if var occurs anywhere in x /* occur-check */
     then return failure
  else return add {x/var} to θ
```

HUOM! Viimeisellä rivillä tulisi olla var/x!

Tämän samastusalgoritmin perusidea on täydentää sidontaa θ vertaamalla samastettavia rakenteita x ja y toisiinsa rekursiivisesti. *Esiintymistarkistus* (askel $occur\text{-}check$) estää kehät.

Nämä rakenteet ovat

joko muuttujasymboleita ($\textsc{Variable?}$)

tai rakenteita ($\textsc{Compound?}$) jotka koostuvat

- operaattorista ($\textsc{Op}$), joka on predikaatti- tai funktiosymboli
- ja sen argumenttilistasta ($\textsc{Args}$), jota käydään läpi jäsen jäseneltä ($\textsc{First}$, $\textsc{Rest}$).

Algoritmi 13: Samastusalgoritmi [RN95, kuva 10.3].

Luvun 5.3.2 esimerkin tietämys on esitetty Hornin lauseilla, kunhan sitä muunnetaan ensin hieman tekemällä päättelyaskeleet 1 ja 2: aksiooman (48) nojalla on jokin ohjus, joten annetaan sille nimeksi uusi vakiotermi $\mathrm{M1}$.

Toistetaan sitten esimerkki tällä päättelysäännöllä:

1. Vanha askel 4 on nyt

$$\frac{\overset{(55)}{\mathrm{missile}(\mathrm{M1})} \quad \overset{(51)}{\forall x.\, \mathrm{missile}(x) \Rightarrow \mathrm{weapon}(x)}}{\mathrm{weapon}(\mathrm{M1})}\, GMP$$

koska $\textsc{Unify}(\mathrm{missile}(\mathrm{M1}), \mathrm{missile}(x)) = \{x/\,\mathrm{M1}\}$.

Tämän askeleen sisällä tehtiin myös vanha askel 3. Ennen vaadittiin nokkeluutta valittaessa sopivaa arvoa $\mathrm{M1}$ muuttujasymbolille x, mutta nyt riittää *syntaktinen* samastustoimitus. Valinta on siis rakennettu käytetyn yleistetyn Modus Ponens -päättelysäännön (56) *sisään*.

2. Vanha askel 9 on samanlainen päättely:

$$\frac{\overset{(46)}{\mathrm{enemy}(\mathrm{Nono}, \mathrm{America})} \quad \overset{(52)}{\forall x.\, \mathrm{enemy}(x, \mathrm{America}) \Rightarrow \mathrm{hostile}(x)}}{\mathrm{hostile}(\mathrm{Nono})}\, GMP$$

3. Vanha askel 6 on samanlainen päättely, mutta nyt atomikaavoja on useita:

$$\frac{\overset{(54)}{\mathrm{owns}(\mathrm{Nono}, \mathrm{M1})} \; \overset{(55)}{\mathrm{missile}(\mathrm{M1})} \; \overset{(50)}{\forall x.\, \mathrm{owns}(\mathrm{Nono}, x) \wedge \mathrm{missile}(x) \Rightarrow \mathrm{sells}(\mathrm{West}, \mathrm{Nono}, x)}}{\mathrm{sells}(\mathrm{West}, \mathrm{Nono}, \mathrm{M1})}$$

4. Vanha loppuaskel 11 on seuraavalla sivulla.

Enää ei tarvita erillisten atomikaavojen yhdistelyä konjunktioin kuten vanhassa askeleessa 10, koska tämäkin esivaihe on sisäänrakennettu päättelysääntöön (56).

$$\frac{1 \quad \overset{(47)}{\mathrm{nation}(\mathrm{Nono})} \quad 2 \quad 3 \quad \overset{(45)}{\forall x, y, z.\, \mathrm{american}(x) \wedge \mathrm{weapon}(y) \wedge \mathrm{nation}(z) \wedge \mathrm{hostile}(z) \wedge \mathrm{sells}(x, z, y) \Rightarrow \mathrm{criminal}(x)}}{\mathrm{criminal}(\mathrm{West})}\, GMP$$

Edellinen esimerkki ratkaistiin *eteenpäin ketjuttamalla (forward chaining):* tietämyskannan KB jo sisältämien faktojen perusteella *laukaistiin (fire)* jokin sääntö, joka tuotti uuden faktan kantaan lisättäväksi.

Jokainen tietämyskannan KB sääntö muotoa (57) luetaan siis "jos tietämyskannassa on jo faktat $p_1, \ldots, p_n$ niin lisää siihen myös fakta q".

Näin jatketaan kunnes odotettu fakta ilmestyy kantaan tai ei enää ole mitään lisättävää:

procedure FORWARD-CHAIN(*KB*, *p*)

  if there is a sentence in *KB* that is a renaming of *p* **then return**
  Add *p* to *KB*
  for each $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$ **in** *KB* such that for some *i*, UNIFY(p_i, p) $= \theta$ succeeds **do**
    FIND-AND-INFER(*KB*, $[p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n], q, \theta$)
  end

---

procedure FIND-AND-INFER(*KB*, *premises*, *conclusion*, θ)

  if *premises* = [ ] **then**
    FORWARD-CHAIN(*KB*, SUBST(θ, *conclusion*))
  else for each p' **in** *KB* such that UNIFY(p', SUBST(θ, FIRST(*premises*))) $= \theta_2$ **do**
    FIND-AND-INFER(*KB*, REST(*premises*), *conclusion*, COMPOSE(θ, θ_2))
  end

Aliohjelma $\textsc{Compose}(\theta, \theta_2)$ palauttaa sellaisen sidonnan θ', joka yhdistää argumentteinaan saamansa sidonnat θ ja θ_2 peräkkäin: toisin sanoen, sidonnan θ' käyttö yksinään tuottaa aina saman tuloksen kuin jos olisi käytetty ensin sidontaa θ ja sen perään sidontaa θ_2.

Algoritmi 14: Eteenpäin ketjuttava päättelykone [RN95, kuva 9.1].

- Eteenpäinketjutus *ei välttämättä pysähdy* jos tietämyskannassa KB on argumentillisia funktiosymboleja: esimerkiksi kannasta

$$p(a) \qquad \forall x.p(x) \Rightarrow p(f(x))$$

  ketjuttuu äärettömästi faktoja $p(a)$, $p(f(a))$, $p(f(f(a)))$,. . .
  Valitettavasti esimerkiksi luvun 5.2.3 tilannekalkyylissä on väistämättä tulosfunktio result.

- Eteenpäinketjutuksella *ei ole näkemystä* siitä tuloksesta, mihin kannattaisi pyrkiä, vaan se toimii pelkästään tietämyskannan KB kulloisenkin sisällön ohjaamana.
  Se on siis eräänlaista leveyssuuntaista etsintää.
  Tällaisia ohjelmia sanotaan *tieto-ohjatuiksi (data-driven, data-directed)*.

Tehokkaampaa (ainakin muistin- ja usein myös ajankäytöltään) on *taaksepäin ketjuttaminen (backward chaining):*

- Lähdetään liikkeelle siitä faktasta q, joka halutaan todistaa.
- Valitaan kokeiltavaksi sellainen sääntö r muotoa (57) jonka johtopäätöksenä voitaisiin saada tämä q.
- Katsotaan, mitkä kaikki faktat täytyisi lisäksi todistaa, jotta voitaisiin käyttää tätä valittua sääntöä r.
  Todistetaan nekin vuorostaan taaksepäin ketjuttamalla.
- Jatketaan kunnes etsitty todistus löytyy tai kaikki vaihtoehdot r on kokeiltu.

Jokainen tietämyskannan KB sääntö r

$$\forall \vec{x}. \underbrace{p_1 \wedge \ldots \wedge p_n}_{\text{runko}} \Rightarrow \underbrace{q'}_{\text{pää}} \tag{58}$$

luetaan siis "jos tahdot tehdä johtopäätöksen q', niin yksi tapa on tehdä ensin johtopäätökset $p_1, \ldots, p_n$".

Sääntöjä kokeillaan siis seuraavasti:

1. Valitaan ensin jokin sellainen sääntö r, jonka *pää (head)* q' samastuu haluttuun johtopäätökseen p; olkoon θ' vastaava yleisin samastin.
2. Sitten jatketaan samalla tavalla valitun säännön r *rungon (body)* johtopäätöksien $p'_i = \textsc{Subst}(\theta', p_i)$ metsästämistä.

Algoritmi 15 toimii tällä periaatteella, ja ratkoo osaongelmat p_i' rekursiivisesti. Näin saatu syvyyssuuntainen etsintä on luonteva mutta ei ainoa mahdollinen tapa ketjuttaa taaksepäin.

```
function BACK-CHAIN(KB, q) returns a set of substitutions
  BACK-CHAIN-LIST(KB, [q], {})
--------------------------------------------------------------------
function BACK-CHAIN-LIST(KB, qlist, θ) returns a set of substitutions
  inputs: KB, a knowledge base
          qlist, a list of conjuncts forming a query (θ already applied)
          θ, the current substitution
  static: answers, a set of substitutions, initially empty

  if qlist is empty then return {θ}
  q ← FIRST(qlist)
     for each q'_i in KB such that θ_i ← UNIFY(q, q'_i) succeeds do
       Add COMPOSE(θ, θ_i) to answers
     end
     for each sentence (p_1 ∧ ... ∧ p_n ⇒ q'_i) in KB such that θ_i ← UNIFY(q, q'_i) succeeds do
       answers ← BACK-CHAIN-LIST(KB, SUBST(θ_i, [p_1 ... p_n]), COMPOSE(θ, θ_i)) ∪ answers
     end
  return the union of BACK-CHAIN-LIST(KB, REST(qlist), θ) for each θ ∈ answers
```

HUOM! Muuttujan $answers$ tulisi olla **local** kuten kirjassa eikä **static** kuten kuvassa!

Algoritmi 15: Taaksepäin ketjuttava päättelykone [RN95, kuva 9.2].

Algoritmi 15 ratkoo itse asiassa seuraavaa etsintäongelmaa:

Alkutila saadaan kysymyksestä "päteekö *faktojen konjunktio*

$$q_1 \wedge \ldots \wedge q_m \tag{59}$$

tietämyskannasta KB (joka sisältää jälleen vain Hornin lauseita)?"
Alkutila koostuu tuosta *maalista* (59) ja tyhjästä sidonnasta $\emptyset$.
Tietämyskanta KB kuuluisi oikeastaan myös etsinnän tiloihin, mutta jätetään se pois koska se ei muutu etsinnän aikana.

Operaattorit saadaan käyttämällä yhtälöä (56) takaperin kuten edellä selitettiin.

Koostukoon etsinnän nykyinen tila ylläolevasta maalista (58) ja sidonnasta θ. Valitaan

- jokin *alimaali (subgoal)* q_i maalista (59), ja
- jokin yhtälön (58) mukainen sääntö r tietämyskannasta KB (mutta ensin *nimetään uudelleen* säännön r kaikki muuttujat aiemmin käyttämättömillä muuttujasymboleilla sen merkitystä kuitenkaan muuttamatta, jotta ne pysyisivät muista erillään)
- siten, että valittu alimaali q_i ja valitun säännön pää q' samastuvat; olkoon vastaava yleisin samastin θ'.

Näiden valintojen määräämä seuraava etsinnän tila koostuu maalista

$$\textsc{Subst}(\theta', q_1 \wedge \ldots \wedge q_{i-1} \wedge p_1 \wedge \ldots \wedge p_n \wedge q_{i+1} \wedge \ldots \wedge q_m)$$

ja sidonnasta $\mathrm{compose}(\theta, \theta')$.
Valittu alimaali q_i korvataan siis valitun säännön rungolla ja sovelletaan vastaavaa sidontaa.

Lopputila on sellainen, jonka maalissa ei ole enää yhtään atomikaavaa jäljellä.
Silloin vastataan kertynyt sidonta θ; se kertoo millaisiin arvoihin alkutilan alkuperäisen maalin (59) vapaat muuttujat piti sitoa jotta etsintä onnisti.
Näillä palautusarvoilla voidaan välittää kutsujalle esimerkiksi niitä result-termejä, jotka esittivät tietämyskannan KB ehdottamaa toimintasuunnitelmaa luvussa 5.2.3.

Tätä etsintäongelmaa voi ajatella *ristiriitatodistuksen* etsintänä:

- Alkuperäinen maali (59) on Hornin lause

$$\forall \vec{x}. q_1 \wedge \ldots \wedge q_m \Rightarrow \mathbf{F}, \tag{60}$$

missä $\vec{x}$ ovat nuo vastauksia palauttavat muuttujat. Sen ekvivalentti muotoilu on

$$\neg \exists \vec{x}. q_1 \wedge \ldots \wedge q_m,$$

ja tarkoituksena on osoittaa että sen lisääminen ristiriidattomaksi oletettuun tietämyskantaan KB aiheutti ristiriidan.

- Tämä tehdään palauttamalla vastaussidonnassa sopivat arvot muuttujille $\vec{x}$.
- Ristiriita havaitaan päätymällä lopputilan tyhjään konjunktioon $\mathbf{T}$.

Logiikkaohjelmointikieli Prolog [RN95, luku 10.3] käyttää tätä periaatetta siten, että:

- Aina valitaan *ensimmäinen* alimaali q_1, jolloin maali toimii *rekursiokutsupinona*.

  Tämä valinta on vapaa, koska jokainen alimaali on joskus valittava ja eri valintajärjestykset ovat keskenään vaihtoehtoiset.
- Tietämyskannan eli *ohjelman KB* sääntöjä kokeillaan aina niiden *kirjoitusjärjestyksessä*.

  Tässä valinnassa taas pitäisi taata, että jokaista sääntöä tullaan kokeilemaan. Kirjoitusjärjestys(kään) ei sitä takaa.
- Ristiriidan etsintää suoritetaan *syvyyssuuntaisesti* (luku 3.1.3).
- Samastusalgoritmista jätetään esiintymistarkistus pois.

Nämä valinnat takaavat toisaalta hyvän suorituskyvyn mutta toisaalta tekevät etsinnästä *epätäydellisen* ja joskus jopa loogisesti *virheellisen*. Kuitenkin Prolog on erinomainen toteutuskieli tekoälyohjelmille, koska jo sen omat perusperiaatteet ovat varsin ilmaisuvoimaisia.

Tehdään vielä kerran luvun 5.3.2 esimerkki, nyt Prolog-tyylisellä taaksepäinketjuttamisella. Käytetään myös uusia vastausmuuttujiamme, ja kysytään "kuka on rikollinen?"

1. Alkutilanteessa maali on

$$\mathrm{criminal}(x),$$

   missä x on vastausmuuttuja, ja sijoitus on $\emptyset$.

Loogisesti vastaväitämme siis että "rikollisia ei ole" ja osoitamme sitten tämän vastaväitteen vääräksi etsimällä jonkin rikollisen.

2. Ainoa mahdollisuus on soveltaa sääntöä (45). Muuttujien uudelleennumeroinnin jälkeen saadaan

$$\mathrm{american}(x) \wedge \mathrm{weapon}(y_1) \wedge \mathrm{nation}(z_1) \wedge \\ \mathrm{hostile}(z_1) \wedge \mathrm{sells}(x, z_1, y_1)$$

   ja sijoitukseksi tulee $\{x_1/x\}$.
3. Ainoa mahdollisuus on soveltaa faktaa (49)

$$\mathrm{weapon}(y_1) \wedge \mathrm{nation}(z_1) \wedge \mathrm{hostile}(z_1) \wedge \mathrm{sells}(\mathrm{West}, z_1, y_1)$$

   ja sijoitukseksi tulee $\{x_1/x, x/\,\mathrm{West}\}$.
4. Ainoa mahdollisuus on soveltaa sääntöä (51). Muuttujien uudelleennumeroinnin jälkeen saadaan

$$\mathrm{missile}(y_1) \wedge \mathrm{nation}(z_1) \wedge \mathrm{hostile}(z_1) \wedge \mathrm{sells}(\mathrm{West}, z_1, y_1)$$

   ja sijoitukseksi tulee $\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1\}$.
5. Ainoa mahdollisuus on soveltaa faktaa (55). Saadaan

$$\mathrm{nation}(z_1) \wedge \mathrm{hostile}(z_1) \wedge \mathrm{sells}(\mathrm{West}, z_1, \mathrm{M1})$$

   ja sijoitukseksi tulee $\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}\}$.
6. Ensimmäinen mahdollisuus on soveltaa faktaa (47). Saadaan

$$\mathrm{hostile}(\mathrm{Nono}) \wedge \mathrm{sells}(\mathrm{West}, \mathrm{Nono}, \mathrm{M1})$$

   ja sijoitukseksi tulee $\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}, z_1/\,\mathrm{Nono}\}$. Muistetaan samalla toisena mahdollisuutena fakta (53) siltä varalta ettei tämä ensimmäinen vaihtoehto onnistuisikaan.
7. Ainoa mahdollisuus on soveltaa sääntöä (52). Muuttujien uudelleennumeroinnin jälkeen saadaan

$$\mathrm{enemy}(\mathrm{Nono}, \mathrm{America}) \wedge \mathrm{sells}(\mathrm{West}, \mathrm{Nono}, \mathrm{M1})$$

ja sijoitukseksi tulee

$$\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}, z_1/\,\mathrm{Nono}, x_7/\,\mathrm{Nono}\}\,.$$

8. Ainoa mahdollisuus on soveltaa faktaa (46). Saadaan

$$\mathrm{sells}(\mathrm{West}, \mathrm{Nono}, \mathrm{M1})$$

ja sijoituksena säilyy

$$\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}, z_1/\,\mathrm{Nono}, x_7/\,\mathrm{Nono}\}\,.$$

9. Ainoa mahdollisuus on soveltaa sääntöä (50). Muuttujien uudelleennumeroinnin jälkeen saadaan

$$\mathrm{owns}(\mathrm{Nono}, \mathrm{M1}) \wedge \mathrm{missile}(\mathrm{M1})$$

ja sijoituksena säilyy

$$\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}, z_1/\,\mathrm{Nono}, x_7/\,\mathrm{Nono}\}\,.$$

10. Ainoa mahdollisuus on soveltaa faktaa (54). Saadaan

$$\mathrm{missile}(\mathrm{M1})$$

ja sijoituksena säilyy

$$\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}, z_1/\,\mathrm{Nono}, x_7/\,\mathrm{Nono}\}\,.$$

11. Ainoa mahdollisuus on soveltaa faktaa (55). Saadaan vihdoin tyhjä maali, eli ristiriita on löytynyt. Sijoituksena säilyy

$$\{x/x_1, x_1/\,\mathrm{West}, x_3/y_1, y_1/\,\mathrm{M1}, z_1/\,\mathrm{Nono}, x_7/\,\mathrm{Nono}\}$$

ja sen perusteella vastausmuuttuja x sai arvokseen West.
Vaihtoehtoon 6 ei siis tällä kertaa tarvinnut palata. Jos sittenkin palattaisiin, niin saataisiin seuraava sellainen rikollinen x, joka kyetään päättelemään, jne.

5.3.4. Resoluutio [RN95, luvut 9.6 ja 9.7] [Nil98, luvut 14 ja 16]

Jos tietämystä ei voikaan esittää Hornin lauseilla muodossa (57), niin silloin ei luvun 5.3.3 yleistetty Modus Ponens riitä, vaan on siirryttävä käyttämään yleisempiä päättelymekanismeja.

Resoluutio (resolution) on tunnetuin automatisoituva päättelyjärjestelmä *koko* ensimmäisen kertaluvun predikaattilogiikalle.

Tarkemmin sanoen kyseessä on *ristiriitatäydellisyys (refutation completeness)*: jos annettu tietämys on sisäisesti ristiriitaista, niin on olemassa äärellinen resoluutiopäättelyaskelten jono, joka löytää jonkin ristiriidan.

Tätä voidaan hyödyntää kuten luvun 5.3.3 taaksepäin ketjuttamisessa: oletetaan itse tietämyskanta KB ristiriidattomaksi, lisätään siihen annetun kyselyn negaatio, ja aletaan etsimään ristiriitaa.

Myös vastaussidonnat käsitellään kuten taaksepäin ketjuttamisessa.

Tietämys muunnetaan resoluutiota varten ensin *konjunktiiviseen normaalimuotoon (conjunctive normal form)*

$$\mathrm{CNF}(\phi) \equiv \bigwedge_{1\leq i\leq m} D_i, \text{ missä}$$
$$D_i \equiv \forall \vec{x_i}. \bigvee_{1\leq j\leq n_i} L_{i,j}$$

ja jokainen $L_{i,j}$ on *literaali (literal)*: atomikaava tai sellaisen negaatio. Jokainen $\vec{x_i}$ sisältää jälleen täsmälleen ne muuttujat, jotka esiintyvät vapaana kyseisessä disjunktiossa D_i.

Voisimme käyttää myös *implikatiivista (implicative)* normaalimuotoa, jossa D_i esitettäisiinkin implikaationa

$$\forall \vec{x}_i . P_{i,1} \wedge \ldots \wedge P_{i,k_i} \Rightarrow P_{i,k_i+1} \vee \ldots \vee P_{i,n_i}$$

missä $P_{i,1}, \ldots, P_{i,k_i}$ ovat ne atomikaavat, jotka esiintyivät negatiivisina eli muodossa $L_{i,1} = \neg P_{i,1}$ jne.

- Kun $n_i = k_i + 1$, eli implikaation oikealla puolella on tasan yksi (1) atomikaava, on kyseessä tuttu Hornin lause: fakta $(k_i = 0)$ tai sääntö $(k_i > 0)$.
- Kun $n_i > k_i + 1$, on kyseessä sellainen sääntö, jolla on *useita* vaihtoehtoisia tuloksia: "faktoista $P_{i,1}, \ldots, P_{i,k_i}$ seuraa fakta P_{i,k_i+1} tai P_{i,k_i+1} tai. . . ".
  Tällaista *disjunktiivista* informaatiota ei Hornin lauseilla voinut esittää.
- Kun $n_i = k_i$, on kyseessä $\forall \vec{x}_i . P_{i,1} \wedge \ldots \wedge P_{i,k_i} \Rightarrow \mathbf{F}$, eli taaksepäin ketjutuksen maalikaava (60), eli negaatio $\neg\exists \vec{x}_i . P_{i,1} \wedge \ldots \wedge P_{i,k_i}$.
  Tällaista *negatiivista* informaatiota ei Hornin lauseilla voinut esittää.

Normalisointi voidaan suorittaa seuraavasti:

1. **Korvaa lyhennysmerkinnät määritelmillään** paitsi '$\forall$' ja '$\vee$'.
   Esimerkiksi "jokainen ihminen on laulun arvoinen" on aluksi muotoa

   $$\forall x. \, \mathrm{person}(x) \Rightarrow \exists y. \, \mathrm{song}(y) \wedge \mathrm{deserves}(x, y)$$

   mutta muuntuu muotoon

   $$\forall x. \neg \, \mathrm{person}(x) \vee \exists y. \, \mathrm{song}(y) \wedge \mathrm{deserves}(x, y). \tag{61}$$

2. **Kvantifioi vapaat muuttujat** lisäämällä koko kaavan eteen universaalikvanttori '$\forall x$' jokaiselle muuttujasymbolille 'x' joka esiintyy vapaana.
   Tämä noudattaa määritelmää (32).
3. **Vie negaatiot sisään** eli sovella *uudelleenkirjoitussääntöjä (rewrite rules)*

   $$\neg \left(\alpha \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \beta \right) \text{ muotoon } (\neg\alpha) \begin{smallmatrix} \wedge \\ \vee \end{smallmatrix} (\neg\beta)$$
   $$\neg \begin{smallmatrix} \exists \\ \forall \end{smallmatrix} x\alpha \text{ muotoon } \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} x \neg\alpha$$
   $$\neg\neg\alpha \text{ muotoon } \alpha$$

   niin kauan kuin mahdollista.
   Nämä säännöt säilyttävät selvästi kaavan semantiikan.
4. **Uudelleennimeä muuttujat** siten, että sama muuttujasymboli esiintyy korkeintaan yhdessä kvanttorissa.
   Näin muuttujat eivät enää voi sekaantua toisiinsa, kun kvanttoreita myöhemmin käsitellään.
5. **Tuo kvanttorit ulos** koko kaavan eteen uudelleenkirjoitussäännöillä

   $$\alpha \circ (Qx\beta) \text{ muotoon } Qx(\alpha \circ \beta),$$

   missä '$\circ$' on konnektiivi '$\vee$' tai '$\wedge$' ja Q on kvanttori '$\exists$' tai '$\forall$', sekä sen toiset muodot konnektiivien $\circ$ vaihdannaisuuden suhteen. (Eli $\alpha \circ \beta$ tarkoittaa samaa kuin $\beta \circ \alpha$.)
   On työlästä osoittaa näidenkin sääntöjen säilyttävän kaavan semantiikan. Se perustuu huomioon, että edellisen askeleen nojalla muuttujasymboli x ei enää esiinny vapaana kaavassa α.
   Esimerkiksi kaava (61) muuntuu muotoon

   $$\forall x \exists y. \neg \, \mathrm{person}(x) \vee \mathrm{song}(y) \wedge \mathrm{deserves}(x, y). \tag{62}$$

6. **Skolemisoi** tarkastelemalla edellisen vaiheen tuottamaa kaavaa

   $$Q_1 x_1 \ldots Q_n x_n . \alpha, \tag{63}$$

   missä α ei enää sisällä kvanttoreita. Poista *kvanttoriprefiksin (quantifier prefix)* vasemmanpuoleisin eksistenssikvanttori $Q_i = \exists$ siirtymällä kaavaan

   $$Q_1 x_1 \ldots Q_{i-1} x_{i-1} Q_{i+1} x_{i+1} \ldots Q_n x_n . \\ \textsc{Subst}(\{x_i / f_i(x_1, \ldots, x_{i-1})\}, \alpha),$$

missä f_i on uusi funktiosymboli (tai vakio jos $i = 1$), jota ei ole aikaisemmin esiintynyt tietämyskannassa KB.

Toista tätä kunnes jäljellä on vain universaalikvanttoreita.

Kaavan (63) mukaan jokaiselle muuttujien $x_1, \ldots, x_{i-1}$ mahdolliselle arvoyhdistelmälle $\vec{u} = \langle u_1, \ldots, u_{i-1} \rangle \in \mathcal{U}^{i-1}$ on olemassa ainakin yksi arvo $u_i \in \mathcal{U}$ joka käy muuttujalle x_i. Funktio f_i esittää vastaavaa operaatiota, joka poimii jonkin sopivan arvon, eli semanttisesti "$f_i(\vec{u}) = u_i$".

Funktiosymbolia f_i nimitetään *Skolem*-funktioksi (tai -vakioksi) tekniikan kehittäjän, norjalaisen loogikon Thoralf Skolemin mukaan.

Esimerkiksi kaava (62) muuttuu muotoon

$$\forall x. \neg \operatorname{person}(x) \vee \operatorname{song}(\operatorname{a\_song\_for}(x)) \wedge \operatorname{deserves}(x, \operatorname{a\_song\_for}(x)). \tag{64}$$

Jos Skolem-funktion $\operatorname{a\_song\_for}(x)$ sijasta olisi käytetty vakiota, joka ei olisikaan riippunut muuttujan x arvosta, niin olisikin sanottu "jokainen ihminen on *saman* laulun arvoinen".

Skolemisointi muuttaa hieman kaavan alkuperäistä merkitystä, mutta "turvalliseen" suuntaan: alunperin yhdistelmälle $\vec{u}$ saattoi olla monia sopivia u_i, mutta f_i valitsee niistä yhden erikoisasemaan.

7. Nosta konjunktiot disjunktioiden yli uudelleenkirjoitussäännöllä

$$(\alpha \wedge \beta) \vee \gamma \text{ muotoon } (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$$

sekä sen variantilla disjunktion vaihdannaisuuden suhteen.

Nämä säännöt säilyttävät selvästi kaavan semantiikan.

Esimerkiksi kaava (64) muuttuu muotoon

$$\begin{aligned} \forall x.(&\neg \operatorname{person}(x) \vee \operatorname{song}(\operatorname{a\_song\_for}(x))) \\ &\wedge \\ (&\neg \operatorname{person}(x) \vee \operatorname{deserves}(x, \operatorname{a\_song\_for}(x))). \end{aligned} \tag{65}$$

8. Nosta konjunktiot kvanttoreiden yli uudelleenkirjoitussäännöillä

$$\forall x(\alpha \wedge \beta) \text{ muotoon } (\forall x \alpha) \wedge (\forall x \beta).$$

Nämä säännöt säilyttävät jälleen kaavan semantiikan.

Esimerkiksi kaava (65) muuttuu lopulta muotoon

$$\begin{gathered} (\forall x_1. \neg \operatorname{person}(x_1) \vee \operatorname{song}(\operatorname{a\_song\_for}(x_1))) \wedge \\ (\forall x_2. \neg \operatorname{person}(x_2) \vee \operatorname{deserves}(x_2, \operatorname{a\_song\_for}(x_2))) \end{gathered}$$

eli implikaatioina

$$\begin{gathered} (\forall x_1. \operatorname{person}(x_1) \Rightarrow \operatorname{song}(\operatorname{a\_song\_for}(x_1))) \wedge \\ (\forall x_2. \operatorname{person}(x_2) \Rightarrow \operatorname{deserves}(x_2, \operatorname{a\_song\_for}(x_2))). \end{gathered}$$

Päädyttiin siis Hornin lauseisiin "jokaiseen ihmiseen liittyy jokin laulu" ja "jokainen ihminen on tuon häneen liitetyn laulun arvoinen".

Lopuksi voidaan

- unohtaa tarpeettomat sulut
- jättää kvanttorit merkitsemättä
- esittää disjunktiot D_i literaalien joukkoina
- esittää koko konjunktio literaalijoukkojen joukkona

$$\begin{aligned} \{\{\neg \operatorname{person}(x_1), \operatorname{song}(\operatorname{a\_song\_for}(x_1))\}&, \\ \{\neg \operatorname{person}(x_2), \operatorname{deserves}(x_2, \operatorname{a\_song\_for}(x_2))\}&\} \end{aligned}$$

jolloin ei tarvitse huolehtia

- saman literaalin tai konjunktin toistumisesta
- literaalien tai konjunktien keskinäisestä järjestyksestä.

Normalisoinnin huonoja puolia:

- Ihmisen kirjoittaman tietämyksen ulkoasu muuttuu.
- Päättelyaskeleiden ymmärtäminen vaiheutuu ulkoasun muututtua.
- Normalisointi nojaa klassiseen logiikkaan, eikä aina yleisty muihin logiikoihin.

Toisaalta ilman normalisointia pelkkä resoluutiosääntö ei riitä.

Normalisoinnin jälkeen itse resoluutiosääntö on

$$\frac{P \vee L_1 \vee \ldots L_m \quad \neg P' \vee L_1' \vee \ldots L_n'}{\textsc{Subst}(\theta, L_1 \vee \ldots L_m \vee L_1' \vee \ldots L_n')}\, RES \tag{66}$$

missä

- $\theta = \textsc{Unify}(P, P')$
- merkitsemättä jätetyt universaalikvantifioinnit käsitellään kuten luvun 5.3.3 taaksepäin ketjutuksessa.

Toisin sanoen

1. kun valitaan yhdestä ehdosta jokin positiivinen literaali P (disjunktien järjestyshän oli vapaa)
2. ja toisesta ehdosta jokin negatiivinen literaali $\neg P'$
3. jotka samastuvat kuten algoritmissa 13
4. niin johtopäätös eli *resolventti (resolvent)* saadaan
   (a) pyyhkimällä pois valitut literaalit P ja $\neg P'$
   (b) yhdistämällä loput ehdoista toisiinsa
   (c) ja soveltamalla yleisintä samastinta θ.

Implikatiivisessa normaalimuodossa atomikaavat P ja P' taas valitaan *eri puolilta* ehtojen implikaationuolia.

Luvun 5.3.3 yleistetty Modus Ponens -sääntö (56) on resoluutiosarja

$$\cfrac{p_2' \quad \cfrac{p_1' \quad \neg p_1 \vee \ldots \vee \neg p_n \vee q}{\textsc{Subst}(\theta_1, \neg p_2 \vee \ldots \vee \neg p_n \vee q)}\, RES}{\cfrac{\textsc{Subst}(\textsc{Compose}(\theta_1, \theta_2), \neg p_3 \vee \ldots \vee \neg p_n \vee q)}{\vdots \atop \textsc{Subst}(\theta, q)}}\, RES \tag{67}$$

missä

- $\theta_1 = \textsc{Unify}(\textsc{Rename}(p_1'), p_1)$
- Syntaktinen operaatio $\textsc{Rename}(p_1')$ tuottaa sellaisen kaavan, joka saadaan nimeämällä kaavan p_1' vapaat muuttujat uudelleen sellaisiksi, joita ei käyteta missään muualla.
  Tarkkaan ottaen sääntökin pitäisi nimetä uudelleen, mutta se mutkistaisi merkintöjä entisestään.
- $\theta_2 = \textsc{Unify}(\textsc{Rename}(p_2'), \textsc{Subst}(\theta_1, p_2))$
- $\textsc{Subst}(\theta_2, \textsc{Subst}(\theta_1, \phi)) = \textsc{Subst}(\textsc{Compose}(\theta_1, \theta_2), \phi)$; tämähän on se ehto, joka määrittelee operaation $\textsc{Compose}$.
- $\theta = \textsc{Compose}(\textsc{Compose}(\textsc{Compose}(\theta_1, \theta_2), \theta_3), \ldots), \theta_n)$.
  Tämä on itse asiassa sama sijoitus kuin se, joka täytyy muodostaa sääntöä (56) sovellettaessa.

Esimerkkinä (implikatiivisen normaalimuodon) resoluutiosäännöllä löydetystä refutaatiosta olkoon seuraava ongelma:

- Jackillä on koira:

$$\exists x.\, \mathrm{Dog}(x) \wedge \mathrm{Owns}(\mathrm{Jack}, x) \text{ eli normalisoituna}$$
$$\mathrm{Dog}(\mathrm{D}) \text{ ja}$$
$$\mathrm{Owns}(\mathrm{Jack}, \mathrm{D}).$$

- Jokainen koiranomistaja on eläinrakas:

$$\forall x.(\exists y.\, \mathrm{Dog}(y) \wedge \mathrm{Owns}(x, y)) \Rightarrow \mathrm{AnimalLover}(x)$$
$$\mathrm{Dog}(y) \wedge \mathrm{Owns}(x, y)) \Rightarrow \mathrm{AnimalLover}(x).$$

- Kukaan eläinrakas ei tapa eläimiä:

$$\forall x.\, \mathrm{AnimalLover}(x) \Rightarrow \forall y.\, \mathrm{Animal}(y) \Rightarrow \neg\, \mathrm{Kills}(x, y)$$
$$\mathrm{AnimalLover}(x) \wedge \mathrm{Animal}(y) \wedge \mathrm{Kills}(x, y) \Rightarrow \mathbf{F}.$$

  Tämä ei ole Hornin lause, koska siinä ei ole ainuttakaan positiivista literaalia.

- Jack tai uteliaisuus tappoi Tuna-kissan:

$$\text{Kills}(\text{Jack}, \text{Tuna}) \lor \text{Kills}(\text{Curiosity}, \text{Tuna}).$$

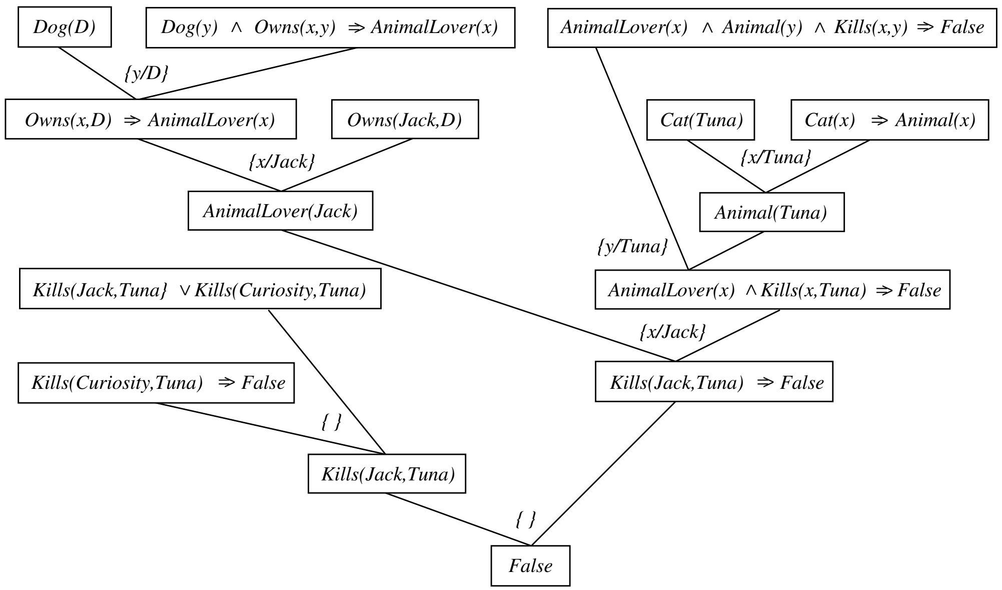
  Normalisointia ei nyt tarvittu. Tämäkään ei ole Hornin lause, koska siinä on enemmän kuin yksi positiivinen literaali.
- Tuna-kissasta tarvitaan vielä:

$$\text{Cat}(\text{Tuna}) \text{ ja}$$
$$\forall x.\, \text{Cat}(x) \Rightarrow \text{Animal}(x) \text{ eli}$$
$$\text{Cat}(x) \Rightarrow \text{Animal}(x).$$

- Tappoiko uteliaisuus Tuna-kissan?
  Vastaväite on "uteliaisuus ei tappanut Tuna-kissaa" eli

$$\text{Kills}(\text{Curiosity}, \text{Tuna}) \Rightarrow \mathbf{F}.$$

Näistä normalisoiduista kaavoista saadaan ristiriita seuraavilla resoluutioaskeleilla:



Kuva 61: Esimerkki resoluutiorefutaatiosta [RN95, kuva 9.7].

Etsittäessä ristiriitaa resoluutiolla mahdollisia alimaalin ja säännön valintoja on hyvin paljon, koska kyseessä ovat yleiset konjunktiivisen normaalimuodon eivätkä ainoastaan Hornin lauseet. Niinpä etsintään on kehitetty erilaisia *strategioita*:

Faktojen suosimisessa (unit preference) ainakin toinen resolvoitavista kaavoista valitaan faktaksi.

Faktojen käyttö pienentää sitä disjunktiota, joka halutaan tyhjentää, kun taas muiden sääntöjen käyttö kasvattaisi sitä.

Faktojen suosiminen ei yksinään vielä riitä etsinnän nopeuttamiseksi – eihän se edes lähde käyntiin jos tietämys on aluksi faktaton!

Päättely (67) oli faktoja suosiva.

Tukijoukko (set of support) on sellainen esitetyn tietämyksen kaavojen osajoukko, jossa ristiriidan oletetaan piilevän. Silloin on johdonmukaista valita ainakin toinen resolvoitava kaava tästä joukosta. Jokainen näin päätelty kaava lisätään tähän joukkoon.

Jos tukijoukko on valittu siten, että sen ulkopuolelle jäävät kaavat ovat keskenään ristiriidattomia, niin silloin strategia säilyttää resoluution täydellisyyden. (Siis jos ristiriita on olemassa, niin *on olemassa* myös jokin tukijoukkostrategian mukainen ristiriitatodistus; mutta tämän todistuksen *löytyminen* taas riippuu käytetyn etsintämenetelmän täydellisyydestä.)

Yleinen tapa onkin ottaa tukijoukoksi kumottavan johtopäätöksen negaatio(sta saadut kaavat) ja olettaa itse tietämyskanta ristiriidattomaksi. Näin löydetyt resoluutiotodistukset ovat "helppolukuisia" koska niissä näkyy tavoite vastata kyselyyn.

Syöteresoluutiossa (input resolution) valitaan aina toiseksi resolvoitavaksi jokin syötteenä saatu kaava.

Syöteresoluutio on täydellinen Hornin lauseille, mutta ei yleisille kaavoille.

Luvun 5.3.3 ketjutukset voitaisiin tehdä myös syöteresoluutiolla.

Lineaarinen (linear) resoluutio on syöteresoluution laajennus, jossa toiseksi resolvoitavaksi valitaan aina joko syötekaava tai jokin nykyisen kaavan todistuksen aikaisempi väivaihe.

Tämä lisäys riittää täydellisyyteen yleisillekin kaavoille.

Luvun 5.3.3 Prolog-taaksepäinketjutus on esimerkki sekä syöte- että lineaarisesta resoluutiosta.

Sisältyminen (subsumption) jättää huomiotta kaikki sellaiset kaavat, jotka seuraavat jostakin tietämyskannassa jo olevasta kaavasta. Tämä vähentää haarautumisasteen kasvua horjuttamatta täydellisyyttä.

Jos esimerkiksi fakta $\mathrm{p}(x)$ on jo tietämyskannassa, niin sinne ei kannata enää lisätä faktaa $\mathrm{p}(\mathrm{a})$ eikä kaavaa $\mathrm{p}(\mathrm{a}) \vee \mathrm{q}(\mathrm{b})$ koska ne seuraavat siitä.

Sisältyminen voidaan selvittää käyttämällä *sovitusta (matching)*, algoritmin 13 samastuksen "yksipuolista" muotoa [CL73, luku 5.8], jossa muuttujia esiintyy vain toisessa kaavoista. esimerkiksi fakta $\mathrm{p}(\mathrm{a})$ sopii hahmoon $\mathrm{p}(x)$ sijoituksella $\{x = \mathrm{a}\}$.

Viitteet

[Bod90] M.A. Boden, editor. *The Philosophy of Artificial Intelligence*. Oxford Readings in Philosophy. Oxford University Press, 1990.

[Car97] B. Carpenter. *Type-Logical Semantics*. The MIT Press, 1997.

[Cho97] N. Chomsky. B. F. Skinnerin teos *verbal behaviour*. In P. Raatikainen, editor, *Ajattelu, kieli, merkitys*, pages 279–310. Gaudeamus, 1997. Suom. R. Vilkko.

[CL73] C-L. Chang and R.C-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[CSC90] P.M. Churchland and P. Smith Churchland. Could a machine think? *Scientific American*, pages 26–31, 1990.

[Dam99] A. Damasio. *The Feeling of What Happens*. William Heinemann, 1999.

[DD88] H.L. Dreyfus and S.E. Dreyfus. Making a mind versus modelling the brain: Artificial intelligence back at a branch-point. *Artificial Intelligence*, 1988.

[Dea98] T.W. Deacon. *The Symbolic Species: The Co-Evolution of Language and the Human Brain*. Penguin Books, 1998.

[MP43] W.S. McCulloch and W.H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–137, 1943. [Bod90, §1].

[Myr74a] L. Myrberg. *Differentiaali- ja integraalilaskenta*, volume 1. Kirjayhtymä, 1974.

[Myr74b] L. Myrberg. *Differentiaali- ja integraalilaskenta*, volume 2. Kirjayhtymä, 1974.

[Nil98] N.J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan-Kaufmann, 1998.

[NKN01] M.K. Nowak, N.L. Komarova, and P. Niyogi. Evolution of universal grammar. *Science*, 291:114–118, 2001.

[Nv01] S. Negri and J. von Plato. *Structural Proof Theory*. Cambridge University Press, 2001.

[Pen89] R. Penrose. *The Emperor's New Mind*. Oxford University Press, 1989.

[Pin94] S. Pinker. *The Language Instinct*. William Morrow and Company, Inc., 1994.

[Pin97] S. Pinker. *How the Mind Works*. W. W. Norton & Company, Inc., 1997.

[Pou91] L.C. Poulson. *ML for the Working Programmer*. Cambridge University Press, 1991.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice-Hall, 1995.

[Saa85] E. Saarinen. *Länsimaisen filosofian historia huipulta huipulle*. Werner Söderström oy, 1985.

[Sea80] J. Searle. Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3:417–424, 1980. [Bod90, §3].

[Sea90] J. Searle. Is the brain's mind a computer program? *Scientific American*, pages 20–25, 1990.

[TE88] T. Tononi and G.M. Edelman. Consciousness and complexity. *Science*, pages 1846–1851, 1988.

[Tur50] A.M. Turing. Computing machinery and intelligence. *Mind*, LIX(2236):433–460, 1950. [Bod90, §2].

[Vää87] J. Väänänen. *Matemaattinen logiikka*. Gaudeamus, 2nd edition, 1987.