Kennisrepresentatie

Formele taal die geschikt is voor het:

- vastleggen van kennis
- redeneren met kennis

Eisen formalisme voor representatie domeinkennis:

- voldoende uitdrukkingskracht
- nette semantiek
- toestaan efficiënte algoritmische interpretatie
- aansluiten bij probleemdomein op natuurlijke wijze

Vormen van kennisrepresentatie

Vele vormen:

- logica (bijvoorbeeld predicatenlogica)
- verzamelingenleer (bijvoorbeeld relationele calculus)
- productieregels
- semantische netwerken
- frames (objectrepresentatie)
- kansrekening (bijvoorbeeld Bayesiaanse netwerken)
- programmeertalen

Geen enkel formalisme is ideaal!

Productieregelformalisme

Vorm regel:

als aan een aantal condities is voldaan
dan mogen bepaalde conclusies worden getrokken

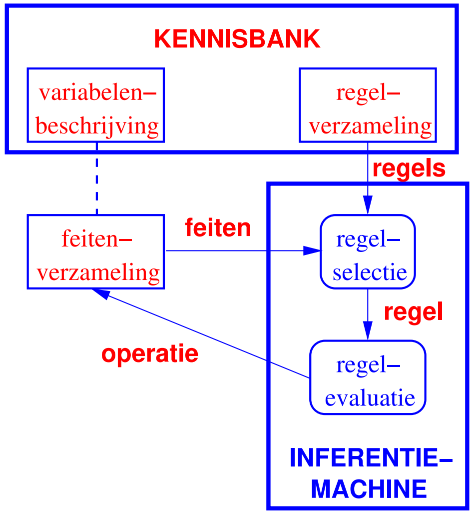
Productieregel: formeel analogon van regel
Toepassingen:

- regel-gebaseerde kennissystemen (MYCIN)
- regel-gebaseerde expert-system shells (EMYCIN)
- regel-gebaseerde programmeertalen (OPS5)
- hybride ontwikkelomgevingen (CLIPS)

Geschiedenis productiesystemen

- A. Newell & H.A. Simon (1972): modelleren menselijk gedrag (boek “Human Problem Solving”, 1972):
  - short-term memory
  - long-term memory
- DENDRAL (Feigenbaum, Lederberg, Buchanan): (beperkt) gebruik in computersysteem
- (E)MYCIN (Shortliffe, van Melle): gebruik voor diagnostische taken
- OPS (o.a. Forgy): model van berekening

Componenten van productiesysteem



- Feitenverzameling: data (geobserveerd of geconcludeerd)
- Regelverzameling: rule-base

De feitenverzameling

- Feit: variabele met waarde(n).
- Twee soorten variabelen:
  - eenwaardige variabelen: $x^s = c$, met c een constante
  - meerwaardige variabelen: $x^m = C$, met $C = \{c_1, \ldots, c_n\}, n \geq 0$, een verzameling constanten
- Meta-constante 'unknown' (een- en meerwaardige variabelen); 'x = unknown' betekent 'x heeft geen waarde gekregen'

Voorbeeld:
$F = \{$geslacht $=$ man, leeftijd $= 45$, beroep $= \{$informaticus$\}$,
lidmaatschap $= \{$IEEE, ACM, SP$\}\}$

Domeindeclaratie

- Domeindeclaratie beschrijft de variabelen:
  - eenwaardig of meerwaardig
  - additionele informatie, zoals vraagbaarheid, doel
- Getypeerde domeindeclaratie beschrijft bovendien domein van variabele

Voorbeeld:
$D = \{$geslacht$^s = \{$man, vrouw$\}$,
leeftijd$^s =$ int,
lidmaatschap$^m = 2^{\{\text{IEEE, ACM, SP, CDA, ...}\}}$,
beroep$^m = 2^{\{\text{informaticus, arts, ...}\}}\}$

De regelverzameling

Productieregel is uitspraak van de vorm:

<prodregel> ::= **if** <antecedent>
then <consequent> **fi**
<antecedent> ::= <disjunctie> $\{$**and** <disjunctie>$\}^*$
<disjunctie> ::= <conditie> $\{$**or** <conditie>$\}^*$
<consequent> ::= <conclusie> $\{$**also** <conclusie>$\}^*$
<conditie> ::= <predikaat>**(**<variabele>**,**<constante>**)**
<conclusie> ::= <actie>**(**<variabele>**,**<constante>**)**

Regelverzameling: verzameling productieregels

Voorbeeld

if greaterthan(temp, 50) **or**
same(druk, verhoogd)
then add(toestand, alarm) **fi**

Predicaten

- Conditie: test op inhoud van feitenverzameling, mbv predicaat; Betekenis:

Predicaat (voorbeeld)	Eenwaardige variabelen	Meerwaardige variabelen
same(x, c)	$x = c$	$c \in x$
notsame(x, c)	$x \neq c$, $x \neq$ unknown	$c \notin x$, $x \neq$ unknown
lessthan(x, c)	$x < c$	-
equal(x, c)	$x = c$ (numeriek)	-
greaterthan(x, c)	$x > c$	-
known(x)	$x \neq$ unknown	$x \neq$ unknown
notknown(x)	$x =$ unknown	$x =$ unknown

- Meta-predikaten 'known' en 'notknown': test op toestand van inferentie

Voorbeeld

same(x, c):
conditie: same(temp, hoog)
feit: temp = laag $\Rightarrow$ false

conditie: same(lidmaatschap, ACM)
feit: lidmaatschap = {IEEE, ACM} $\Rightarrow$ true

notsame(x, c):
conditie: notsame(lidmaatschap, CDA)
feit: lidmaatschap = {IEEE, SP} $\Rightarrow$ true

conditie: notsame(lidmaatschap, CDA)
feit: lidmaatschap = unknown $\Rightarrow$ false

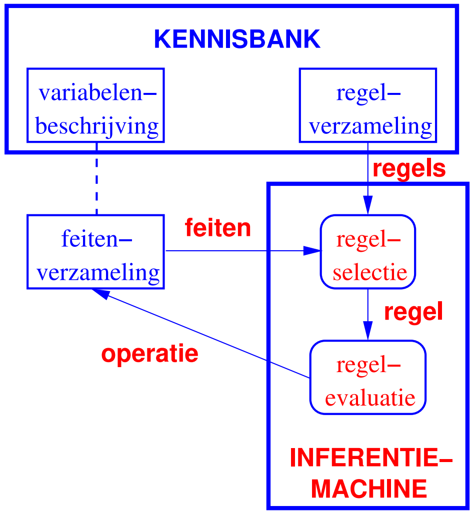
Acties

- Conclusie: operatie op inhoud van feitenverzameling, mbv actie.

Actie (voorbeeld)	Betekenis voor eenwaardige variabelen	Betekenis voor meerwaardige variabelen
add(x, c)	$x := c$	$x := x \cup \{c\}$
remove(x, c)	$x :=$ unknown	als $x = \{c\}$ dan $x :=$ unknown anders $x := x \setminus \{c\}$

Voorbeeld:
conclusie: add(toestand, alarm)
$F = \{$temp = 30, druk = verhoogd$\}$
$\Rightarrow F' := \{$temp = 30, druk = verhoogd, toestand = alarm$\}$

Componenten van productiesysteem II



- Regelselectie: match feiten en productieregels
- Regelevaluatie: evalueren predicaten en (bij succes) uitvoeren acties

Inferentie in productiesysteem

Twee vormen van inferentie met regels:

- top-down inferentie: doelgestuurd
- bottom-up inferentie: datagestuurd

Voorbeeld:
Feitenverzameling:

$$F = \{A\}$$

en regelverzameling:

$$\mathcal{R} = \{R_1 : \textbf{if } B \textbf{ then } G \textbf{ fi}, \quad R_2 : \textbf{if } G \textbf{ then } C \textbf{ fi}, \quad R_3 : \textbf{if } A \textbf{ then } B \textbf{ fi}\}$$

Top-down inferentie met doel G resulteert in: $F' = \{A, B, G\}$
Bottom-up inferentie resulteert in: $F' = \{A, B, G, C\}$

Top-down inferentie: de basis

Traceren van variabele: bepalen van waarde(n) van variabele

```
procedure TraceValues(variable)
    Infer(variable);
    if    not established(variable)
          and askable(variable)
    then  Ask(variable) fi
end
```

- Doelen: doelvariabelen (x_g^s en x_g^m in domeindeclaratie)
- Algoritme top-down inferentie: tracht m.b.v. regels de doelen te traceren
- Van vraagbare variabelen mag waarde aan gebruiker gevraagd worden (x_a^s en x_a^m in domeindeclaratie)

Afleiden van waarden

Afleiden van waarde(n) voor variabele gebeurt mbv regelverzameling.

```
procedure Infer(variable)
    Select(rule-base, variable, selected-rules);
    foreach rule in selected-rules do
        Apply(rule)
    od
end
```

- Selectie van alle relevante productieregels uit regelverzameling
- Geselecteerde regels worden één voor één toegepast. In basisalgoritme ligt volgorde niet vast

Selectie van regels

Selectie van regel: naamsovereenkomst (sub)doelvariabele en (een) variabele in consequent van regel (‘match’)

```
procedure Select(rule-base, variable, selected-rules)
    selected-rules ← ∅;
    foreach rule in rule-base do
        matched ← false;
        foreach concl in consequent(rule)
            and not matched do
            pattern ← var(concl);
            if match(pattern, variable)
            then selected-rules ← selected-rules ∪ {rule};
                matched ← true fi
        od
    od
end
```

Verzameling ‘selected-rules’ heet conflict set voor variabele ’variable’

Voorbeeld

R_1 : **if** same(x,a) **and** same(x,b)
then add(z,f) **fi**
R_2 : **if** same(x,b) **then** add(z,g) **fi**
R_3 : **if** greaterthan(w,10) **then** add(y,d) **fi**
R_4 : **if** same(x,c) **then** add(v,h) **fi**
R_5 : **if** equal(w,8)
then add(v,e) **also** add(z,f) **fi**

Zij z de doelvariabele. De conflictset voor z is:

$$\mathrm{CS}_z = \{R_1, R_2, R_5\}$$

Toepassen van een productieregel

Toepassen regel: evalueren condities en eventueel evalueren conclusies

```
procedure Apply(rule)

    EvalConditions(rule);
    if    not failed(rule)
    then EvalConclusions(rule)
    fi
end
```

Productieregel faalt als bij evaluatie tenminste één (disjunctie van) conditie(s) false oplevert; anders slaagt de regel

Evalueren van condities

Evalueren condities: evalueren predikaten van condities

```
procedure EvalConditions(rule)

    foreach condition in antecedent(rule) do
        var ← variable(condition);
        TraceValues(var);
        ExecPredicate(condition);
        if failed(condition)
        then return
        fi
    od
end
```

Evalueren van condities $\Rightarrow$ nieuwe subdoelen voor inferentie
Condities worden één voor één geëvalueerd. In basisalgoritme ligt volgorde niet vast

Evalueren van conclusies

Evalueren conclusies: uitvoeren van acties van conclusies.

```
procedure EvalConclusions(rule)

    foreach concl in consequent(rule) do
        ExecAction(concl)
    od
end
```

Conclusies worden één voor één geëvalueerd. In basisalgoritme ligt volgorde niet vast

Voorbeeld

R_1 : **if** same(y,a) **and** notsame(w,a)
then add(z,b) **fi**
R_2 : **if** same(w,a) **then** add(z,a) **fi**
R_3 : **if** lessthan(x,10) **then** add(y,a) **fi**
R_4 : **if** greaterthan(x,5) **then** add(y,b) **fi**
R_5 : **if** lessthan(x,40) **then** add(w,b) **fi**
$D = \{w^m, x^s_a, y^m, z^s_g\}$; $F = \{x = 8\}$.

- Initieel: selectie R_1 en R_2
- R_1 als eerste toegepast
- 'same(y, a)' als eerste geëvalueerd

Voorbeeld – vervolg

R_1 : **if** same(y,a) **and** notsame(w,a)
then add(z,b) **fi**
R_2 : **if** same(w,a) **then** add(z,a) **fi**
R_3 : **if** lessthan(x,10) **then** add(y,a) **fi**
R_4 : **if** greaterthan(x,5) **then** add(y,b) **fi**
R_5 : **if** lessthan(x,40) **then** add(w,b) **fi**
$D = \{w^m, x^s_a, y^m, z^s_g\}$; $F = \{x = 8\}$.

- Variabele y: nieuw subdoel
- Selectie en toepassing R_3 en R_4; beide regels slagen

Waarden a en b afgeleid voor variabele y!

Voorbeeld – vervolg

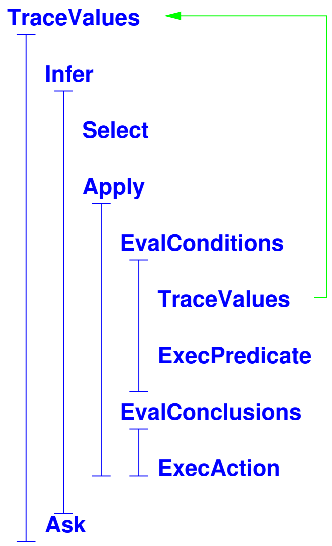
R_1 : **if** same(y,a) **and** notsame(w,a) **then** add(z,b) **fi**
R_2 : **if** same(w,a) **then** add(z,a) **fi**
R_3 : **if** lessthan(x,10) **then** add(y,a) **fi**
R_4 : **if** greaterthan(x,5) **then** add(y,b) **fi**
R_5 : **if** lessthan(x,40) **then** add(w,b) **fi**
$D = \{w^m, x^s_a, y^m, z^s_g\}$; $F' = \{x = 8, y = \{a, b\}\}$

- Evaluatie 'same(y,a)' levert true
- Evaluatie 'notsame(w,a)'
- Variabele w: nieuw subdoel
- Selectie en toepassing R_5; de regel slaagt
- Evaluatie 'notsame(w,a)': levert true

Bij evaluatie 'notsame(w,a)': variabele w is uitputtend getraceerd

Top-down inferentie: samenvatting

Globale structuur algoritme:



Verfijningen van het algoritme

Verfijningen basisalgoritme top-down inferentie:

- oplossingen voor twee vormen van non-determinisme
- variabele ten hoogste één maal getraceerd
- productieregel ten hoogste één maal toegepast
- look-ahead faciliteit toegevoegd

Verfijnd algoritme is lineair in aantal productieregels

Non-determinisme van de eerste soort

Non-determinisme van eerste soort: volgorde van toepassen productieregels uit conflictset is niet gespecificeerd

Voorbeeld:

R_1 : **if** B **and** C **then** A **fi**
R_2 : **if** E **then** D **fi**
R_3 : **if** F **and** G **and** D **then** A **fi**

Zij A het doel; $\mathrm{CS}_A = \{R_1, R_3\}$

- Evaluatie-volgorde R_1, R_3: achtereenvolgens traceren A, B, C, F, G, D, E
- Evaluatie-volgorde R_3, R_1: achtereenvolgens traceren A, F, G, D, E, B, C

Conflictresolutie

Non-determinisme van eerste soort: oplossen m.b.v. conflictresolutie strategie
Voorbeelden:

- Kies regel die als eerste in verzameling voorkomt
- Kies regel met meeste vervulde condities
- Kies regel met belangrijkste conclusie

Keuze van strategie beïnvloedt uiterlijk gedrag van systeem, rekentijd en inhoud van feitenverzameling

Non-determinisme van de tweede soort

Non-determinisme van tweede soort: volgorde evaluatie condities en conclusies van geselecteerde productieregel is niet gespecificeerd.

Voorbeeld:
R_1 : **if** B **and** C **and** D **and** E **then** A **fi**
Mogelijke volgorden evaluatie condities ($n!$):

- B, C, D, E
- C, B, D, E
- C, D, B, E
- $\ldots$

Oplossing

Non-determinisme van tweede soort: oplossing m.b.v. strategie

Voorbeelden:

- Evalueer condities in volgorde van waarschijnlijkheid van slagen
- Evalueer condities in volgorde van specificatie

Keuze van strategie beïnvloedt uiterlijk gedrag van systeem, rekentijd en inhoud van feitenverzameling

Backward chaining

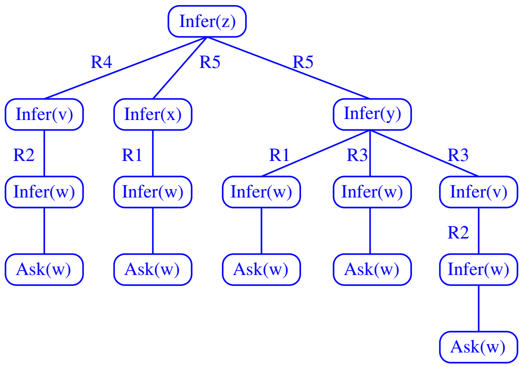
Backward chaining: vorm van top-down inferentie waarbij evaluatie van:

- productieregels uit conflict set, en
- condities van regel, en
- conclusies van regel

plaats vindt in volgorde van specificatie
Backward chaining: strategie voor non-deter-minisme van eerste en tweede soort

Onnodig traceren

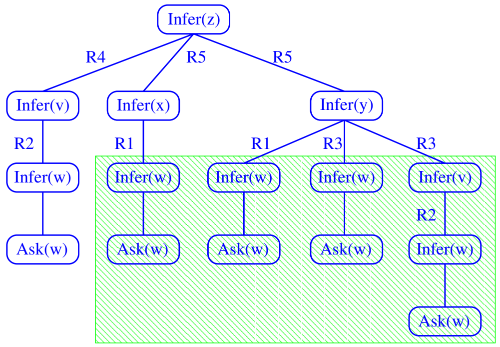
R_1 : **if** notsame(w,b) **then** add(x,b) **also** add(y,m) **fi**
R_2 : **if** same(w,a) **then** add(v,c) **fi**
R_3 : **if** same(w,d) **and** same(v,c) **then** add(y,e) **fi**
R_4 : **if** same(v,c) **then** add(z,k) **fi**
R_5 : **if** same(x,b) **and** notsame(y,e) **then** add(z,h) **fi**



Oplossing

Markeer variabele als getraceerd zodra deze uitputtend getraceerd is

- Alle vragen ten hoogste eenmaal gesteld
- Productieregels met één conclusie ten hoogste eenmaal toegepast (als variabele niet in zowel antecedent als consequent voorkomt)



Markeren van variabelen

```
procedure TraceValues(variable)
    Infer(variable);
    if  not established(variable)
        and askable(variable)
    then Ask(variable)
    fi;
    traced(variable) ← true
end
procedure EvalConditions(rule)
    foreach condition in antecedent(rule) do
        var ← variable(condition);
        if not traced(var) then TraceValues(var)
        fi;
        ExecPredicate(condition);
        if failed(condition) then return
        fi
    od
end
```


Onnodig toepassen van regels

Regel met meer dan één conclusie hoeft slechts eenmaal toegepast te worden. Voorwaarde: alleen add-acties in regelverzameling (monotonie)

Voorbeeld:

R_1 : **if** same(x,a) **then**
add(y,b) **also** add(z,c) **fi**

R_2 : **if** same(y,b) **and** notsame(z,d)
then add(w,e) **fi**

R_3 : **if** notsame(x,b) **then** add(z,e) **fi**

$D = \{x_a^m, y^m, z^m, w_g^m\}$; $F = \{x = \{a\}\}$.

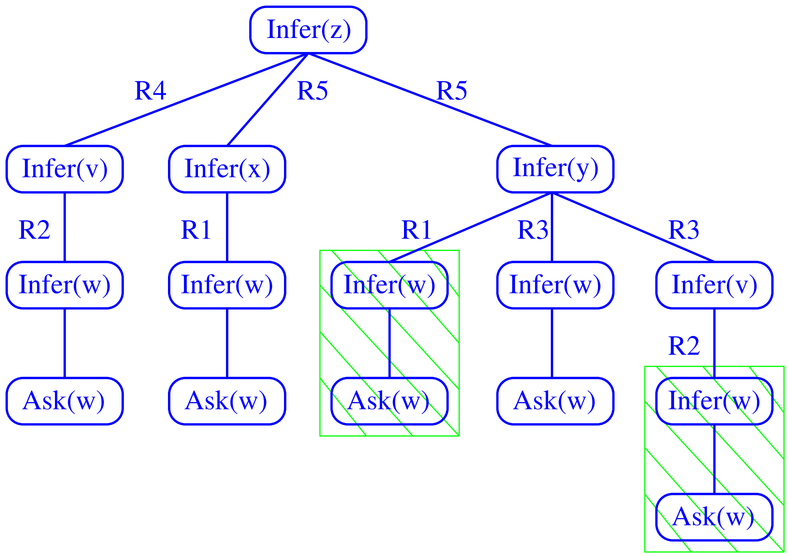
1. Traceren y: toepassen R_1. Toekennen waarde c aan z; z is niet getraceerd!
2. Traceren z: alleen toepassen R_3

Markeren van regels

Geen onnodig toepassen productieregel: markeer geselecteerde regel (voor toepassing) als gebruikt

```
procedure Infer(variable)
    Select(rule-base, variable, selected-rules);
    foreach rule in selected-rules
        with not used(rule) do
            used(rule) ← true;
            Apply(rule)
    od
end
```


Effect van regelmarkering



Self-referencing rules

Self-referencing rule: productieregel met variabele die in zowel antecedent als consequent voorkomt

Voorbeeld:
R_1 : **if** notsame(x,a) **then** add(x,b) **fi**

Basisalgoritme: kans op oneindige lus. Markeer daarom geselecteerde regel voor toepassing als gebruikt

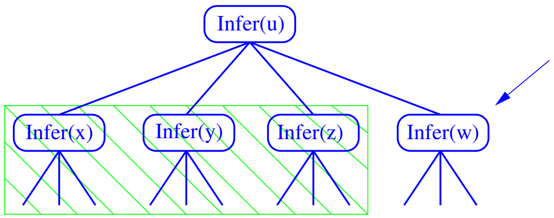
Voorbeeld:
R_1 : **if** same(x,a) **then** add(y,5) **fi**
R_2 : **if** greaterthan(y,10) **then** add(x,b) **fi**

R_1 en R_2 zijn cyclisch gekoppelde regels

Look-ahead faciliteit – voorbeeld

if same(x,a) **and**
same(y,b) **and**
notsame(z,c) **and**
greaterthan(w,20) **and**
lessthan(w,45)
then
add(u,d)
fi

$F = \{y = \{b\}, w = 85\}$



Look-ahead faciliteit

Look-ahead faciliteit: vóór evaluatie van regel onderzoeken of deze al bij voorbaat faalt

function LookAhead(rule)
foreach condition **in** antecedent(rule) **do**
var ← variable(condition);
if traced(var)
then ExecPredicate(condition);
if failed(condition)
then return(false) **fi fi**
od;
return(true)
end

Toepassen look-ahead faciliteit beïnvloedt uiterlijk gedrag van systeem, rekentijd en inhoud van feitenverzameling

Look-ahead faciliteit – vervolg

Look-ahead faciliteit toepassen voor evaluatie productieregel

procedure Infer(variable)
Select(rule-base, variable, selected-rules);
foreach rule **in** selected-rules
with not used(rule) **do**
used(rule) ← true;
succeeded ← LookAhead(rule);
if succeeded
then Apply(rule)
od
end

Verfijningen: samenvatting

Globale structuur verfijnd algoritme top-down inferentie:

- TraceValues*
  - Infer*
    - Select
    - Markeren regel
    - LookAhead
      - Test markering variabele
      - ExecPredicate
    - Apply
      - EvalConditions
        - Test markering variabele
        - TraceValues*
        - ExecPredicate
      - EvalConclusions
        - ExecAction
  - Ask
  - Markeren variabele

Meta-constanten en meta-predikaten

- Traceren van variabele x heeft geen waarde opgeleverd voor x: $x =$ unknown
- Testen op onbekend zijn van variabele met meta-predikaten known en notknown
- Meta-predikaten dwingen traceren van variabele af

Predicaat (voorbeeld)	Betekenis voor eenwaardige variabelen	Betekenis voor meerwaardige variabelen
known(x)	$x \neq$ unknown	$x \neq$ unknown
notknown(x)	$x =$ unknown	$x =$ unknown

Objecten en attributen

Groeperen variabelen met objecten; variabele heet dan attribuut

Voorbeeld

Object: volvo
Attributen: type
leeftijd
vorige_eigenaren
⋮
Object: motor
Attributen: vermogen
type-motor
brandstof
aantal-cilinders
⋮

Object-attribuut-waarde tupels

In productieregels en feiten: object-attribuut-waarde tupels i.p.v. variabele-waarde paren

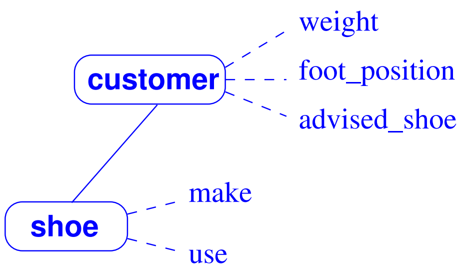
Voorbeeld (sportschoenadvies):

if the weight of the customer is less than 80
the foot_position of the customer is pronation
the make of the shoe is Adidas
the use of the shoe is training
then the advised_shoe of customer is
adidas_torsion_integral_support

if lessthan(customer, weight, 80) **and**
same(customer, foot_position, pronation) **and**
same(shoe, make, Adidas) **and**
same(shoe, use, training)
then add(customer, advised_shoe,
adidas_torsion_integral_support) **fi**

Objectschema

Objectschema: relatie tussen objecten en attributen en tussen objecten onderling (mogelijk ER, NIAM)



Aanvullende informatie in objectschema:

- eenwaardig of meerwaardig zijn van attributen
- vraagbaar of niet vraagbaar zijn van attributen
- toegestane waarden van attributen
- informatie voor gebruikersinterface

Top-down inferentie en o-a-v tupels

```
procedure Activate(object)
   foreach attr in attributes(object) do
      if goal(attr)
      then TraceValues(object, attr) fi
   od
end
procedure TraceValues(object, attribute)
   Infer(object, attribute);
   if  not established(attribute)
       and askable(attribute)
   then Ask(attribute) fi;
   traced(attribute) ← true
end
```


Productieregelformalisme en semantiek

Productieregelformalisme: productieregels en feiten
Semantische vragen:

- Is het mogelijk een betekenis aan feiten en productieregels toe te kennen, los van het inferentie-algoritme?

  $\Rightarrow$ **declaratieve semantiek**
- Als productieregelformalisme een declaratieve semantiek heeft, is de inferentiemethode dan gezond (sound) en volledig (complete)?

  $\Rightarrow$ **procedurele semantiek**

Kennisbank als logische theorie

Voordelen vertaling:

- beoordeling of inferentieregels in inferentie machine gezond (sound) zijn:

$$(\text{KB} \cup F) \vdash G \;\Rightarrow\; (\text{KB} \cup F) \vDash G$$

  met kennisbank KB, feitenverzameling F en doel G
- beoordeling of gebruikte stelsel inferentieregels volledig (complete) is: $(\text{KB} \cup F) \vDash G \;\Rightarrow\; (\text{KB} \cup F) \vdash G$

Productieregels en logica:

1. condities vs. literals
2. conclusies vs. literals
3. productieregels vs. implicaties
4. feiten vs. literals

Condities versus literals

Meerwaardige attributen

Conditie 'same(o,a,v)':

- Idee: a is binaire relatie tussen objecten en waarden, $a \subseteq O \times V$, met O en V verzamelingen objecten en waarden
- Vertaling naar logica-representatie:
  - attribuut $\rightarrow$ 2-plaatsig predicaatsymbool
  - object $\rightarrow$ constante
  - waarde $\rightarrow$ constante
  - 'same' $\rightarrow$ literal

Voorbeeld:

same(patient, klacht, koorts)
$\rightarrow$ klacht(patient, koorts)

Condities versus literals

Eenwaardige attributen

Conditie 'same(o,a,v)':

- Idee: a is functie van verzameling objecten O naar verzameling waarden V, $a : O \rightarrow V$
- Vertaling naar logica-representatie:
  - attribuut $\rightarrow$ 1-plaatsig functiesymbool
  - object $\rightarrow$ constante
  - waarde $\rightarrow$ constante
  - 'same' $\rightarrow$ gelijkheidspredicaat '$=$'

Voorbeeld:

same(patient, leeftijd, 45)
$\rightarrow$ leeftijd(patient) = 45

Conclusies versus literals

Een- en meerwaardige attributen

Conclusie 'add(o,a,v)'
Vertaling conclusie met actie 'add' analoog aan vertaling conditie met predikaat 'same'

Voorbeelden:

add(car, problem, ignition)
$\rightarrow$ problem(car, ignition)
waarin 'problem' een meerwaardig attribuut is
add(car, make, volvo)
$\rightarrow$ make(car) = volvo
waarin 'make' een eenwaardig attribuut is

Vertaling enkele predicaten en acties

Voorbeeld	Logica-representatie voor eenwaardige variabelen	Logica-representatie voor meerwaardige variabelen
same(o, a, v)	$a(o) = v$	$a(o, v)$
notsame(o, a, v)	$\neg a(o) = v$	$\neg a(o, v)$
lessthan(o, a, v)	$a(o) < v$	–
equal(o, a, v)	$a(o) = v$	–
greaterthan(o, a, v)	$a(o) > v$	–
add(o, a, v)	$a(o) = v$	$a(o, v)$

- Meta-predikaten 'known' en 'notknown' niet goed te vertalen naar predicatenlogica
- Acties 'modify' en 'remove'?

Productieregels versus implicaties

Productieregel van de vorm:

$$
\begin{array}{ll}
\textbf{if} & c_{11} \textbf{ or } \cdots \textbf{ or } c_{1n_1} \textbf{ and} \\
& \quad\vdots \\
& c_{m1} \textbf{ or } \cdots \textbf{ or } c_{mn_m} \\
\multicolumn{2}{l}{\textbf{then } a_1 \textbf{ also } \cdots \textbf{ also } a_p \textbf{ fi}}
\end{array}
$$

waarin $m \geq 1$, $n_i \geq 1$, $i = 1, \ldots, m$, $p \geq 1$

Vertaling in logische implicatie:

$$
\begin{array}{ll}
(& (c'_{11} \vee \cdots \vee c'_{1n_1}) \wedge \\
& \quad\vdots \\
& (c'_{m1} \vee \cdots \vee c'_{mn_m})) \\
\multicolumn{2}{l}{\to (a'_1 \wedge \cdots \wedge a'_p)}
\end{array}
$$

waarin c'_{ij} en a'_k de logische representatie van c_{ij}, a_k

Voorbeeld

Beschouw productieregel:

if lessthan(customer, weight, 80) **and**
same(customer, foot_position, pronation) **and**
same(shoe, make, Adidas) **and**
same(shoe, use, training)
then
add(customer, advised_shoe,
adidas_torsion_integral_support)

Vertaling in logische implicatie:
(weight(customer) $<$ 80 $\wedge$
foot_position(customer) = pronation $\wedge$
make(shoe) = Adidas $\wedge$ use(shoe,training))
$\to$ advised_shoe(customer, adidas_torsion_integral_support)

Feiten versus literals

Eenwaardige attributen

Feit '$o.a = v$'
Vertaling feit analoog aan vertaling conditie met predikaat 'same': $a(o) = v$

Voorbeeld:

customer.weight = 85
$\to$ weight(customer) = 85

Feiten versus literals

Meerwaardige attributen

Feit '$o.a = \{v_1, \ldots, v_m\}$', $m \geq 1$
Vertaling in principe als volgt:

$$
\begin{array}{l}
a(o, v_1) \\
\quad\vdots \\
a(o, v_m)
\end{array}
$$

Productiesysteem gaat uit van gesloten wereldbeeld (closed-world assumption) $\Rightarrow$ toevoegen negatieve literals voor resterende waarden:

$$
\begin{array}{l}
\neg a(o, v_{m+1}) \\
\quad\vdots \\
\neg a(o, v_n)
\end{array}
$$

Dit heet negation by absence

Voorbeeld

In productieregelformalisme:

student.klacht = $2^{\{\text{koorts, geelzucht, braken}\}}$
student.klacht = {koorts, braken}
if same(student, klacht, koorts) **and**
notsame(student, klacht, geelzucht) **and**
...
then add(student, aandoening, griep) **fi**

In logica-representatie:
klacht(student, koorts) ∧ klacht(student, braken) ∧
¬ klacht(student, geelzucht)

(klacht(student, koorts) ∧ ¬ klacht(student, geelzucht) ∧
...)
→ aandoening(student, griep)

Gezondheid en volledigheid

- Is top-down inferentie is gezond? (ja bij acceptatie van negation by absence)
- Is top-down inferentie volledig?

  R_1: **if** same(o, a, t) **then** add(o, b, w) **fi**
  R_2: **if** same(o, a, t) **and** lessthan$(o, c, 10)$
  then add(o, b, u) **fi**

  $D = \{o.a_a^m, o.b_g^s, o.c_a^s\}$ en $F = \{o.a = \{t, z\}, o.c = 7\}$

Toepassing top-down inferentie resulteert in:

1. Selectie R_2 en $R_1 \Rightarrow F' = F \cup \{o.b = w, o.b = u\}$;
   logica-representatie: analoog (inconsistent).
2. Selectie R_1 voor R_2 (of omgekeerd); stoppen na toekenning waarde aan $b \Rightarrow F' = F \cup \{o.b = w\}$ of $F' = F \cup \{o.b = u\}$

Onvolledigheid backward chaining

Voorbeeld:

R_1: **if** same(o, a, t) **and** notsame(o, a, s)
then add(o, b, u) **fi**
R_2: **if** same(o, b, u) **and** lessthan$(o, c, 10)$
then add(o, d, v) **fi**
R_3: **if** same(o, d, v) **then** add(o, e, w) **fi**

$D = \{o.a_a^s, o.b^s, o.c_a^s, o.d_g^m, o.e^s\}$
$F = \{o.a = t, o.c = 7\}$

- Toepassing backward chaining resulteert in:
  $F' = \{o.a = t, o.c = 7, o.b = u, o.d = \{v\}\} \Rightarrow o.e = w \notin F'$
- Terwijl in logica:
  $\{a(o) = t, c(o) = 7, (a(o) = t \wedge \neg a(o) = s \to b(o) = u),$
  $(b(o) = u \wedge c(o) < 10 \to d(o, v)), (d(o, v) \to e(o) = w\}$
  $\models e(o) = w$

Logica vs. productiesystemen

Enkele verschillen zijn:

- Productiesysteem:
  - exploitatie onderscheid tussen feiten en productieregels
  - niet-monotoon redeneren mogelijk (‘remove’, ‘notknown’)
  - redeneren met onzekerheid mogelijk
- Logica: in bepaald opzicht expressiever:
  - $\forall x P(x)$
  - $P(a, b) \wedge Q(c, a)$
  - n-plaatsige predicaten en functies met $n > 2$

Nadelen productieregelformalisme

Productieregelformalisme:

- ongeschikt voor representatie descriptieve kennis
- sterk operationeel karakter
- vastleggen en manipuleren verschillende typen kennis op uniforme wijze
- semantiek soms onduidelijk, waardoor uitbreidingen moeilijk te realiseren zijn