3. Knowledge Representation

- Predicate Calculus
- Rules
- Nonmonotonic logic
- Bayesian Networks
- Semantic Networks
- Frames
- Conceptual Dependency
- Scripts

3.1 Predicate Calculus

- well form formulas (wffs) คือสูตรที่ถูกต้องตามกฎเกณฑ์ของ predicate calculus

<u>syntax and semantic of atomic formulas</u>

- predicate calculus language ประกอบด้วย
  - predicate symbols เช่น P, Q, R
  - variable symbols เช่น x, y, z
  - function symbols เช่น f, g, h
  - constant symbols เช่น A, B, C
  - { } [ ] ( ) ,

Syntax and Semantic of Atomic Formulas

- predicate symbolใช้แสดงความสัมพันธ์(relation)ในdomainที่กล่าวถึง

  เช่น FATHER(SOMCHAI,SOMSRI) atomic formula
  - SOMCHAI, SOMSRI เป็น constant symbols
- FATHER(x,y)
  - x, y เป็น variable symbols
- HAS-MONEY(SOMCHAI,salary(SOMCHAI))
  - salary เป็น functionที่mapจากtermหนึ่งไปอีกtermหนึ่ง
- interpretationของwff คือ assignmentของค่าของpredicates, constants, functionsในdomainนั้น

Syntax and Semantic of Atomic Formulas

- assingmentsเหล่านี้นิยามsemanticsของpredicate calculus language
- เมื่อมีการนิยามinterpretationสำหรับatomic formulaแล้ว เราบอกว่า formulaมีค่าเป็น T (true) ถ้า statementที่ถูกแสดงโดยformulaนั้นเป็นจริงในdomain และจะมีค่าเป็น F (false) ถ้าเป็นเท็จ

<u>Connectives</u>

- $\Rightarrow$ (implication), $\sim$ (not), $\vee$ (or), $\wedge$ (and) ใช้เชื่อมatomic formulas หลายตัวเข้าด้วยกันเป็นformulaใหม่

  เช่น John lives in a yellow house.

  LIVE(JOHN,HOUSE-1) $\wedge$ COLOR(HOUSE-1,YELLOW)

Connectives

- เราเรียกformulaที่เชื่อม2 formulasด้วย ⇒ ว่า implication ใช้แสดง if-then

  เช่น

  If the car belongs to John then it is green.

  OWNS(JOHN,CAR-1) ⇒ COLOR(CAR-1,GREEN)

- ~ (not) ใช้เปลี่ยนค่าความจริงของformular

  เช่น

  John did not write computer-chess.

  ~WRITE(JOHN,COMPUTER-CHESS)

Quantification

- ∀ (universal quantifier), ∃ (existential quantifier)

  เช่น All elephants are gray.

  ∀x (ELEPHANT(x) ⇒ COLOR(x,GRAY)

  There is a person who wrote computer-chess.

  ∃x (WRITE(x,COMPUTER-CHESS))
- ถ้าquantifierปรากฏในwff เราอาจคำนวณค่าความจริงของwffนั้นไม่ได้ เช่น กำหนดให้wffเป็น ∀x (P(x)), ให้interpretationของP และให้ infinite domainของentitiesแล้ว เราไม่สามารถเช็คค่าความจริงของentitiesได้ทุกค่า
- First-order predicate calculusคือpredicate calculusที่ไม่มีquantifier ของpredicateหรือของfunction symbols

Examples and Properties of wffs

- (∃x) { (∀y) [P(x,y) ∧ Q(x,y) ⇒ R(x)] }
- ~ (∀y) { (∃x) [P(x) ∨ R(y)] }
- ~ P(A,g(A,B,A))
- ~ (P(A) ⇒ P(B)) ⇒ P(B)

wffs

- ~ f(A)
- f(P(A))
- Q[f(A),(P(B) ⇒ Q(C))]
- A or ~ ⇒ (∀ ~)

ไม่เป็น wffs

Truth Table

- เมื่อกำหนดinterpretationแล้วค่าความจริงของwffสามารถหาได้โดยใช้ truth table

P	Q	P∨Q	P∧Q	P⇒Q	~P
T	T	T	T	T	F
F	T	T	F	T	T
T	F	T	F	F	F
F	F	F	F	T	T

Rules of Inference, Theorem and Proofs

- Rules of inferenceใช้สร้างwffsใหม่จากwffsที่มีอยู่

ตัวอย่าง: modus ponens

$$\frac{W1 \Rightarrow W2 \quad W1}{W2}$$

universal specialization

$$\frac{(\forall x)\ W(x)}{W(A)}$$

A เป็นconstant symbol

- Wffsใหม่ที่เกิดขึ้นเรียกว่า theorems และsequenceของinference rulesที่ใช้ในการสร้างtheoremsเรียกว่า proofs ของtheorems

Unification

$$\frac{(\forall x)\ (\ W1(x) \Rightarrow W2(x)\ ) \quad W1(A)}{W2(A)}$$

โดยการmatch xกับA และแทนค่า Aให้กับ x

- Substitution instanceของ wff ใด ๆได้จากแทนค่า(substitute) termsให้กับvariablesใน wff นั้น ๆ

ตัวอย่าง: instancesของ P(x,f(y),B) เช่น

P(z,f(w),B)

P(C,f(A),B)

Unification (substitution)

- Substitutionสามารถแสดงในรูปของเซ็ตของคู่ลำดับ

$$s = \{ t_1 /v_1\ ,\ t_2 /v_2\ , \ldots , t_n /v_n \}$$

โดยที่คู่ลำดับ t_i /v_i หมายถึงterm t_i ถูกแทนค่าให้กับvariable v_i

- ในตัวอย่างที่แล้ว

s1 = { z/x, w/y }

s2 = { C/x, A/y }

- เราเขียน wff ที่ได้จากกระทำsubstitution s กับ E ด้วย Es

P(z, f(w), B) = P(x, f(y), B) s1

P(C, f(A), B) = P(x, f(y), B) s2

Unification (mgu)

- E_1 และ E_2 unifyกันได้ ถ้ามีsubstitution sที่ทำให้ $E_1s = E_2s$ และในกรณีนี้เราเรียก s ว่าเป็น unifier ของE_1และE_2
- ตัวอย่าง P[x,f(y),B] และ P[x,f(B),B] unifyกันได้โดยมีunifier s = {A/x, B/y} และผลของunificationคือ P[A,f(B),B]
- g เป็น most general unifier (mgu) ของ E_1 และ E_2 ก็ต่อเมื่อ ถ้ามีsเป็นunifierอื่นของE_1และE_2แล้ว จะต้องมีunifier s' ที่ทำให้ $E_1s = E_1gs'$ และ $E_2s = E_2gs'$
- mguของ P[x,f(y),B] และ P[x,f(B),B] คือ {B/y}

Unification Algorithm

Algorithm Unify(L1,L2)

1. IF L1 หรือ L2 เป็นvariablesหรือconstants THEN

   IF L1เท่ากับL2 THEN คืนค่า NIL

   ELSE IF L1เป็นvariable THEN

   IF L1ปรากฏในL2 THEN คืนค่า {FAIL} ELSE คืนค่า {L2/L1}

   ELSE IF L2 เป็นvariable THEN

   IF L2ปรากฏในL1 THEN คืนค่า {FAIL} ELSE คืนค่า {L1/L2}

   ELSE คืนค่า {FAIL}

2. IF predicate หรือ function symbolsของL1ไม่เท่ากับของL2 THEN คืนค่า {FAIL}
3. IF L1มีจำนวนargumentsไม่เท่ากับL2 THEN คืนค่า {FAIL}

Unification Algorithm

4. SUBST := NIL
5. FOR i := 1 TO จำนวนargumentsของL1 DO

   5.1 เรียก algorithm unify ด้วยargumentsตัวที่ i ของL1และL2

   ใส่ผลลัพธ์ไว้ที่ S

   5.2 IF Sประกอบด้วยFAIL THEN คืนค่า {FAIL}

   5.3 IF S <> NIL THEN

   5.3.1 แทนค่าtermsให้กับvariablesในL1และL2ตามS

   5.3.2 SUBST := append(S,SUBST)

6. คืนค่า SUBST

Resolution

- Resolutionเป็นinference ruleที่ใช้กับwffsประเภทที่เรียกว่า clause
- Clauseคือwffที่ประกอบด้วยdisjunctionของliterals

<u>การแปลงpredicate calculusเป็นclause</u>

$(\forall x) \{ P(x) \Rightarrow \{ (\forall y) [P(y) \Rightarrow P(f(x,y))] \wedge \sim(\forall y)[Q(x,y) \Rightarrow P(y)]\}\}$

1. Eliminate implication symbols : เปลี่ยนรูปของ $X \Rightarrow Y$ เป็น $\sim X \vee Y$

   $(\forall x) \{\sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge \sim(\forall y)[\sim Q(x,y) \vee P(y)]\}\}$

2. Reduce scope of negation symbols

   $(\forall x) \{\sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge (\exists y)[Q(x,y) \wedge \sim P(y)]\}\}$

3. Standardize variables : เปลี่ยนชื่อvariablesตามscopeของquantifiers

   $(\forall x) \{\sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge (\exists w)[Q(x,w) \wedge \sim P(w)]\}\}$

Resolution (clause)

4. Eliminate existential quantifiers : แทนค่าvariablesด้วย Skolem function

   $(\forall x) \{\sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge [Q(x,g(x)) \wedge \sim P(g(x))]\}\}$

5. Convert to prenex form : ย้ายuniversal quantifiersทุกตัวมาอยู่หน้าสุด และformที่ได้ใหม่นี้เรียกว่าprenex form

   $(\forall x)(\forall y) \{ \sim P(x) \vee \{[\sim P(y) \vee P(f(x,y))] \wedge [Q(x,g(x)) \wedge \sim P(g(x))]\}\}$

6. Put prenex form in conjuctive normal form : formที่ทุก wffs เชื่อมกันด้วยเคื่องหมาย $\wedge$

   $(\forall x)(\forall y) \{ [\sim P(x) \vee \sim P(y) \vee P(f(x,y))]$

   $\wedge [\sim P(x) \vee Q(x,g(x))] \wedge [\sim P(x) \vee \sim P(g(x))]\}$

Resolution (clause)

7. Eliminate universal quantifier : เราสามารถตัดuniversal quantifier ทิ้งได้เลยเพราะรู้ว่าvariablesทุกตัวเป็นของuniversal quantifier
8. Eliminate ∧ symbol : แทน (X1∧X2∧ · · · ∧Xn) ด้วยเซต {X1,X2, · · · ,Xn} โดยที่Xiใด ๆเป็นdisjunction of literalsหรือclause จากตัวอย่างจะได้3clauses

   (1) ~P(x) ∨ ~P(y) ∨ P(f(x,y))   (2) ~P(x) ∨ Q(x,g(x))

   (3) ~P(x) ∨ ~P(g(x))
9. Rename variables : ไม่ให้variableหนึ่ง ๆปรากฏในหลายclauses

   (1) ~P(x1) ∨ ~P(y) ∨ P(f(x1,y))   (2) ~P(x2) ∨ Q(x2,g(x2))

   (3) ~P(x3) ∨ ~P(g(x3))

Resolution for ground clause

P1∨P2∨ · · · ∨Pn

~P1∨Q2∨ · · · ∨Qm ⟵ parent clauses

P2∨ · · · ∨Pn∨Q2∨ · · · ∨Qm ⟵ resolvent

Parent clauses	Resolvent(s)
P and ~P∨Q	Q
P∨Q and ~P∨Q	Q
P∨Q and ~P∨~Q	~Q∨Q and ~P∨P
~P and P	NIL
~P∨Q and ~Q∨R	~P∨R

General resolution

- ในกรณีที่จะกระทำresolutionกับclausesที่มีvariables เราต้องหา substitutionที่ทำให้parent clausesประกอบด้วยcomplementary literals (literalsตัวที่ต่างกันเฉพาะเครื่องหมาย ~)
- เราสามารถแทนclauseหนึ่ง ๆด้วยเซตของliterals
- กำหนดให้
  - parent clausesเป็น { L_i } และ { M_i }
  - { l_i } และ { m_i } เป็นเซตย่อยของ { L_i } และ { M_i } ตามลำดับ ซึ่งมี s ที่เป็นmguของ { l_i } และ { ~m_i }
  - resolventของclauses { L_i } กับ { M_i } คือ

    { { L_i } - { l_i } }s ∪ { { M_i } - { m_i } }s

Examples of General Resolution

- สำหรับ2clausesใด ๆอาจจะมีresolvent clauseได้มากกว่า1 ซึ่งขึ้นอยู่กับการเลือก { l_i } และ { m_i }

<u>ตัวอย่าง</u>

ให้clausesเป็น

{ L_i } = { P[x,f(A)],P[x,f(y)],Q(y) }   { M_i } = { ~P[z,f(A)],~Q(z) }

และ { l_i } = { [p(x,f(A)] }   { m_i } = { ~P[z,f(A)] }

กรณีนี้resolvent clauseเป็น { P[z,f(y)], Q(y), ~Q(z) }

แต่ถ้าให้ { l_i } = { [P(x,f(A)], P[x,f(y)] }   { m_i } = { ~P[z,f(A)] }

resolvent clauseจะเป็น { Q(A), ~Q(z) }

Resolution Refutation

- เราสามารถกำหนดว่าliterals 2ตัวใด ๆขัดแย้งกัน(contradictory)หรือไม่ โดยดูว่าตัวหนึ่งสามารถunifyกับนิเสธของอีกตัวหนึ่งได้หรือไม่
  - เช่น MAN(x) กับ ~MAN(Spot) ขัดแย้งกัน
    MAN(x) unifyกับ MAN(Spot)ได้
- วิธีของresolution refutationคือ การที่จะพิสูจน์ว่า wff Wเป็นผลสรุปของเซ็ตของwffs S ทำได้โดยการพิสูจน์ว่า S ∪ {~W} ขัดแย้ง (unsatisfiable)
  - เช่น S = { MAN(Marcus), ~MAN(x) ∨ MORTAL(x) }
    W = MORTAL(Marcus)
    S ∪ {~W} = { MAN(Marcus),
    ~MAN(x) ∨ MORTAL(x),
    ~MORTAL(Marcus) }

  จาก2clausesบน เราได้ MORTAL(Marcus) ซึ่งขัดแย้งกับclauseที่3

Resolution Refutation Algorithm

<u>Algorithm Resolution</u> : F สามารถพิสูจน์ P (Pเป็นผลสรุปของF)

1. แปลง F ให้อยู่ในรูปของ clause
2. เปลี่ยน P ให้อยู่ในรูปของนิเสธ
3. Clauses := F ∪ {~P}
4. UNTIL (NILเป็นสมาชิกของClauses(พบความขัดแย้ง))
   OR (Clausesไม่เปลี่ยนแปลง) DO

   4.1 เลือก2clauses C_i, C_j ที่resolveกันได้

   4.2 คำนวณresolventของC_iและC_j เรียกresolventนั้นว่า R_{ij}

   4.3 Clauses := Clauses ∪ {R_{ij}}

Resolution Refutation : Example

Given clauses:

1. MAN(Marcus)
2. POMPEIAN(Marcus)
3. ~POMPEIAN(x1) ∨ ROMAN(x1)
4. RULER(Caesar)
5. ~ROMAN(x2) ∨ LOYALTO(x2,Caesar) ∨ HATE(x2,Caesar)
6. LOYALTO(x3,f1(x3))
7. ~MAN(x4) ∨ ~RULER(y1) ∨ ~TRYASSASSINATE(x4,y1) ∨ ~LOYALTO(x4,y1)
8. TRYASSASSINATE(Marcus,Caesar)

Prove:

HATE(Marcus,Caesar)