BAGIAN 1

PENGENALAN SISTEM
BERBASIS PENGETAHUAN / SISTEM PAKAR

1.1. Pengertian A.I

- Merup. Sub-bid pengetahuan komp. yang ditujukan u/ membuat software (S/W) dan haardware (H/W) yang sepenuhnya bisa menirukan beberapa fungsi otak manusia.

- Sbg. Cabang sains komp. yang mempelajari otomatisasi tingkah laku cerdas (intelligent)

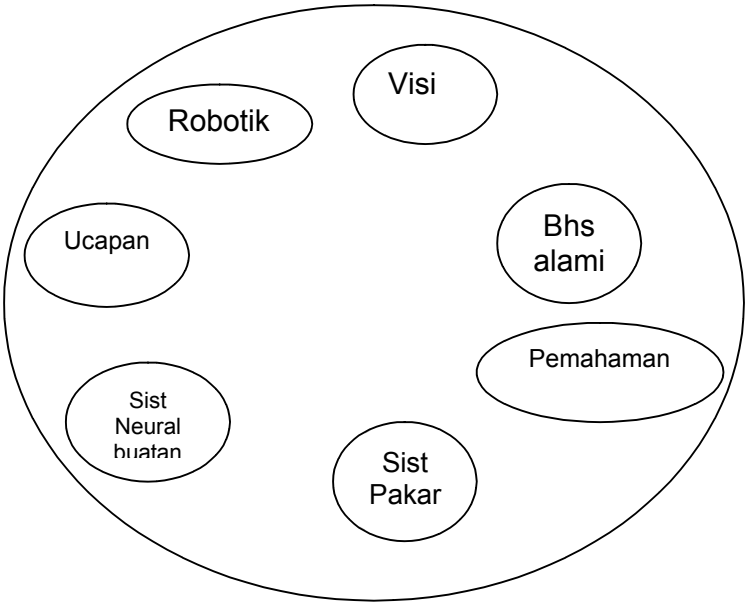
- *Intelligence/Intelegensia :* seseorang yang pandai melaksanakan pengetahuan yang dimilikinya.

- Mahasiswa/Pelajar ?
- Tukang becak ?
- Bayi ?
- Penjahat ?

dkl : kemampuan manusia u/ memperoleh pengetahuan dan pandai melaksanakannya dalam praktek

- *Artificial Intelligence adalah* bidang ilmu komputasi yang memungkinkannya u/ memahami, bernalar & bertindak

- Bagian dari Intelegensi Buatan



1.2. Pengertian Sistem Pakar (Expert Sistem)

- Membuat S/W Expert Systems → prog. Sebagai penasehat/konsultan pakar
- Dapat mengumpulkan dan penyimpan pengetahuan seorang/beberapa orang pakar ke dalam komp. → u/ semua orang yang memerlukan
- Tidak u/ menggantikan kedudukan seorang pakar ttp u/ memasyarakatkan pengetahuan & pengalamaan pakar tsb.
- Memungkinkan orang lain meningkatkan produktivitas, memperbaiki kualitas keputusan dll.

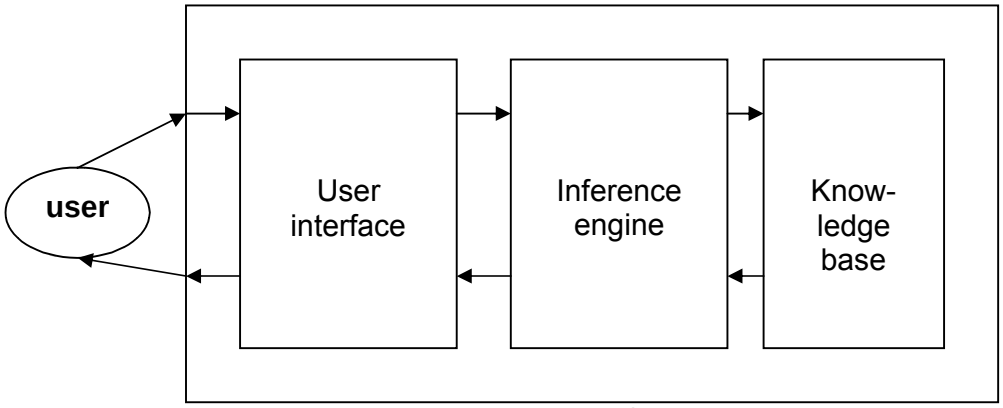
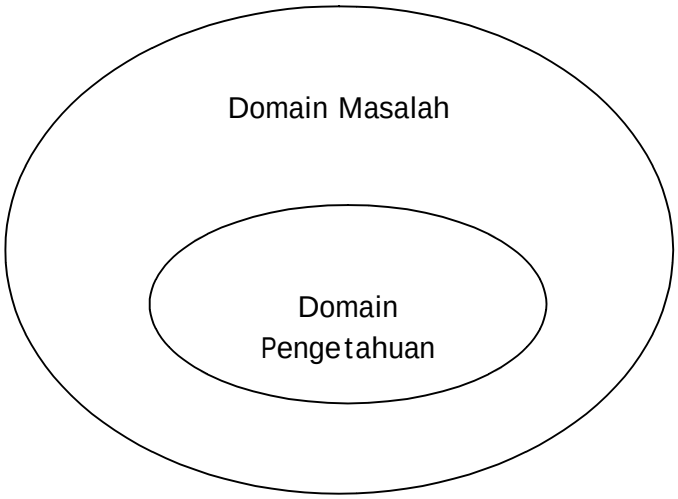


Diagram blok Expert Systems (umum)

- Knowledge base berisi semua fakta, ide, hubungan
- Motor inferensi bertugas u/ menganalisis pengetahuan dan menarik kesimpulan berdasarkan knowledge base.
- S/W user interface berfungsi sbg media pemasukan pengetahuan ke dalam (KB)

Domain Pengetahuan Expert



Keuntungan / Kelebihan Sistem Pakar

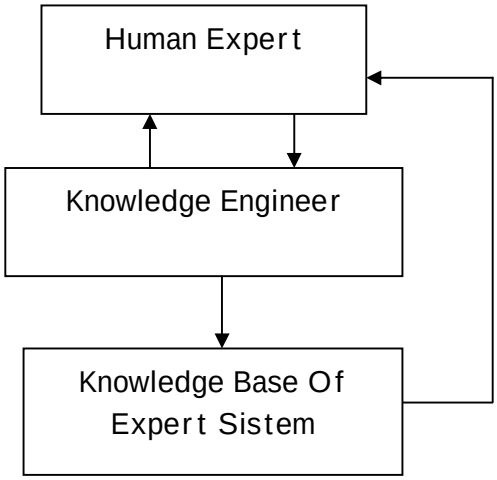
- Availability-bertambah
- Cost-rendah
- Danger-reduced
- Performance
- Multiple expertise
- Reability-bertambah
- Explanation
- Response-cepat
- Steady, unemotional and complete response

* Intelligent tutor
* Intelligent dB

KONSEP UMUM SISTEM PAKAR (SP)

- Salah satu metode representasi pengetahuan:
  IF….. THEN

- Proses pembuatan SP → *knowledge engineering* yg dilakukan oleh *knowledge engineer*. Selain itu *domain expert* dan *end user*.



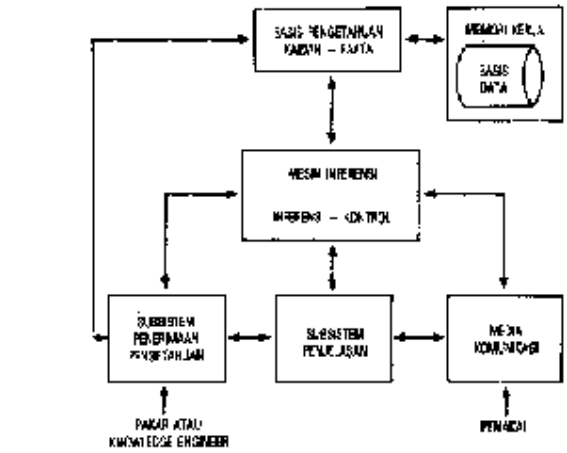
Perkembangan Sistem Pakar

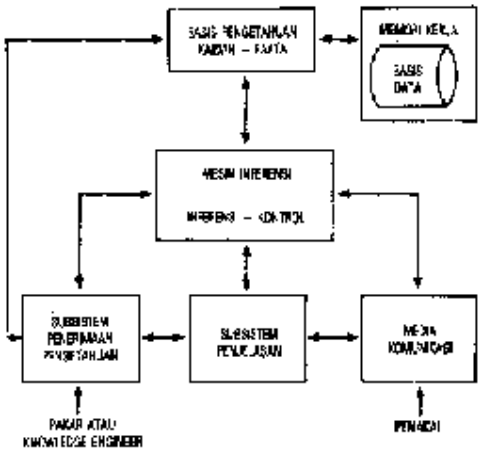
- Tugas *knowledge engineer* adalah memilih S/W & H/W u/ pembuatan SP, membantu mengambil pengetahuan yg dibutuhkan dari pakar domain, serta implementasi pengetahuan pada basis pengetahuan yg benar & efisien

- Tugas pakar domain : menyediakan pengetahuan tentang bidang problem yang dihadapi, memahami teknik-teknik pemecahan problema yang dipakai.

- Batasan praktis dari beberapa SP → *causal knowledge*

- SP lebih mudah untuk diprogram dengan *shallow knowledge*, yaitu *berdasarkan pada pengalaman dan pengetahuan heuristik*.





Keterangan

- **Basis Pengetahuan**
  - inti prog SP
  - representasi pengetahuan dari seorang pakar.
  - Macam-macam ……

- **Mesin Inferensi**
  - mekanisme fungsi berfikir dan pola-pola penalaran sistem yg digunakan pakar
  - menganalisa suatu maslah tertentu
  - mencari jawaban atau solusi yg terbaik.
  - Ada 2 pelacakan backward & forward chaining

KARAKTERISTIK SP

- High Performance
- Adequate response time
- Good reliability
- Understanable
- Flexibility

PENGEMBANGAN TEKNOLOGI SP

- Akar SP pada banyak disiplin ilmu “cognitive science” yaitu study bagaimana orang memikirkan dlm pemecahan masalah. “cognitive processor” yaitu menemukan aturan yg akan diaktifkan.

SP YANG TERKENAL

1. MYCIN

- Dirancang oleh Edward Feigenbaum (Universitas Stanford) th ’70 an
- SP medical yg dpt mendiagnosa infeksi bakteri & rekomendasi pengobatan antibiotik

2. DENDRAL

- SP struktur molekular & kimia

3. PROSPECTOR

- Membantu ahli geologi yg mencari & menemukan biji deposit (mineral& batu-batuan)
- Didesign oleh Sheffield Research Institute, akhir ‘70an

4. XCON (R1)

- SP konfigurasi sistem komputer dasar
- Dikembangkan oleh Digital Equipment Corporation (DEC) dan Carnegie Mellon Universitas (CMU), akhir ’70 an
- Untuk sistem komputer DEC VAC 11 1780

5. DELTA

- Didesign & dikembangkan oleh General Electric Company
- SP personal maintenance dg mesin lokomotif listrik diesel.

6. YESMVS

- Didesign oleh IBM awal th ‘80an

- Membantu operator komputer & mengontrol sistem operasi MVS (multiple virtual storage)

7. ACE

- Didesign & dikembangkan oleh AT&T Bell Lab awal th ‘80an
- SP troubleshooting pd sistem kabel telpon

KLASIFIKASI APLIKASI SP

1. CONTROL

- Aplikasi komputer yg sangat umum
- Ada 2 jenis kontrol : loop terbuka & tertutup

2. DEGUGGING

- Proses mencari kesalahan & memperbaiki solusi.

3. DESIGN

- Pengumpulan informasi mengenai spesifikasi sistem & produk tertentu
- Untuk merancang sirkit elektronik, bangunan, dan rumah.

4. DIAGNOSIS

- Untuk mendiagnosa produk atau sistem yg sudah tdk berfungsi.

5. INSTRUKSIONAL

- Untuk membantu dalam proses belajar mengajar

6. INTERPRETASI

- Membantu seorang dlm menafsir & memahami situasi/perspektif suatu peristiwa.
- Contoh : analisa intelegensia, daya tahan, citra dan sinyal

7. PLANNING

- Merumuskan metode, penataan yg dapat mendekatkan pd tujuan.
- Contoh : proyek manajemen, taktik & strategi militer, pemrograman robot

8. PREDIKSI

- Meramalkan apa yg terjadi di masa yg akan datang.

9. REPARASI

- Memperbaiki barang yg rusak ke keadaan semula

10. KONFIGURASI

ELEMEN SP

1. User Interface --- kom antara user & SP
2. Explanation Facility --- pemberian alasan pd user
3. Working Memori
4. Inference Engine --- penentuan aturan yg hrs dipenuhi, prioritas aturan yg tercukupi, & prioritas yg tertinggi
5. Agenda --- daftar yg diprioritaskan dari aturan (4)
6. Fasilitas Pemrolehan Pengetahuan --- cara otomatis bagi pemakai untuk memasukkan pengetahuan dlm sistem.

Rangkaian Forward (Forward chaining)

→ merupakan pemberi alasan dari fakta untuk kesimpulan hasil dari fakta

Contoh :

Jika kita melihat bahwa hari ini akan turun hujan sebelum pergi (nyata)
Maka kita harus membawa payung (kesimpulan)

Mis : Programan OPS5, CLIPS

Rangkaian Backward (Backward chaining)

- Pemberian alasan sebaliknya dari hipotesa, kesimpulan potensial dibuktikan, pada fakta yg mendukung hipotesa

Contoh:

Jika kita tidak melihat keluar dan seseorang masuk dg sepatu basah dan payung.
Hipotesa kita adalah bahwa hari hujan

Mis : EMYCIN

SISTEM PRODUKSI

- Salah satu type SP yg paling terkenal adalah system yg berdasarkan pd aturan.
- Alasannya :
  1. Modular nature

2. Explanation facility
3. Similarity to the human cognitive process

- POST

  Idenya :

  - System matematika & logika merupakan set aturan sederhana untuk menentukan bagaimana mengubah 1 string simbol ke dlm simbol lainnya.
  - Yaitu dg input string, kejadian sebelumnya,
  - ALGORITMA MARKOV
    - Merupakan kelompok produksi yg terorder yg diterapkan untuk prioritas ke input string.
    - Algoritma akan berakhir dg baik jika:

      (1). Produksi terakhir tidak dapat diterapkan pada string

      (2). Suatu produksi yg berakhir dg periode diterapkan.
      - Jika input string GABKAB

        System produksi AB → HIJ

        Maka hasil akhir GHIJKHIJ
      - *Karakter ^ → string nol*
      - Mis A → ^ artinya menghilangkan seluruh kejadian karakter A dlm suatu string
      - *Karakter tunggal a,b,c,……*
      - Mis AxB → BxA artinya mengubah karakter A dan B
      - Huruf Yunani α,β

Contoh : Memindahkan huruf pertama string input ke akhir

Aturan 1. αxy → yαx

2. α → ^

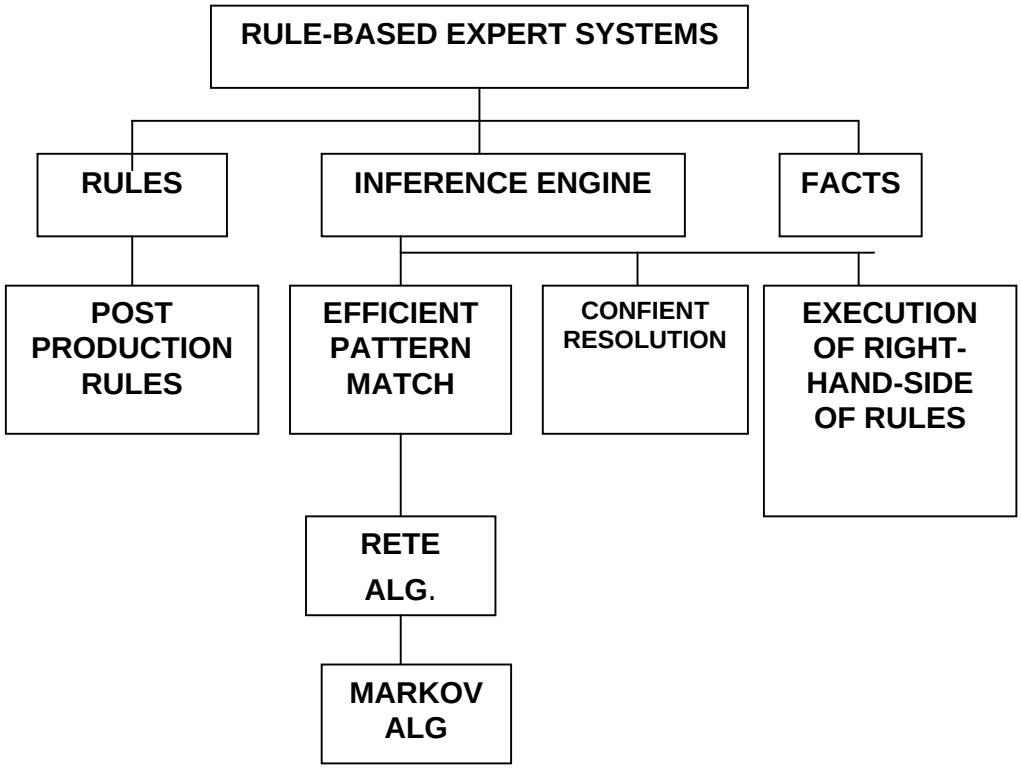
3. ^ → α

Input ABC

Aturan	Sukses atau Gagal	String
1	G	ABC
2	G	ABC
3	S	αABC
1	S	BαAC
1	S	BCαA
1	G	BCαA
2	S	BCA

ALGORITMA RETE

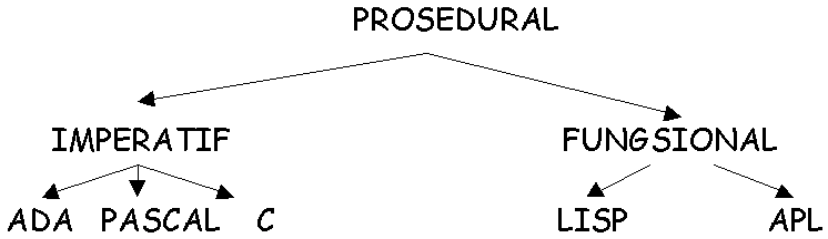
- Pada algoritma Markov diterapkan aturan/baris prioritas lebih tinggi.
- Masalah timbul jika system mempunyai aturan/baris yang banyak, maka tidak akan efisien.
- Solusinya adalah algoritma Rete yang dikembangkan oleh Charles L.F di Carnegie-Mellon University (1979)
- Yaitu algoritma yg mengetahui tentang seluruh aturan/baris seluruh sistem dan dapat menerapkan suatu baris tanpa harus mencoba setiap baris tanpa berangkai (mencari perubahan dalam gabungan setiap cycle)
- Merupakan gabungan pola yang sangat cepat, yang mendapatkan kecepatannya dengan menyimpan informasi tentang baris dalam jaringan.



SP YANG BERSADARKAN ATURAN MODEREN

KLASIFIKASI PARADIGMA PEMROGRAMAN

1. Paradigma Prosedural

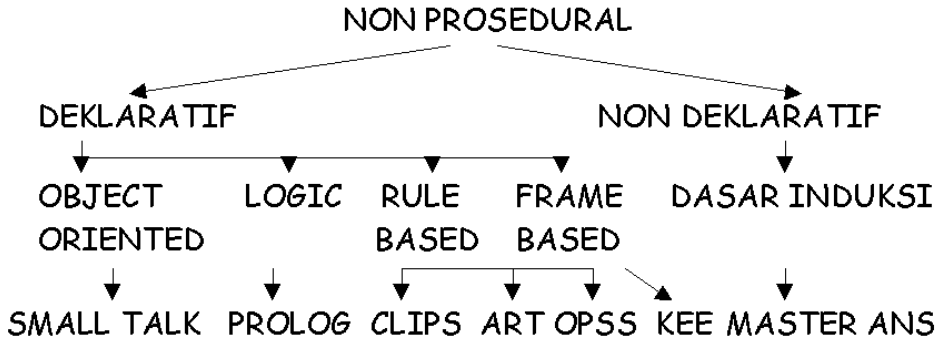


ALGORITMA adalah metode untuk pemecahan masalah dalam sejumlah tahap/langkah tertentu.

✓ Implementasi algoritma dalam suatu program disebut program prosedural.

- ✓ Pemrograman algoritma (prosedural) dan konvensional untuk program type non-AI.
- ✓ Sinonim untuk pemrograman prosedural adalah prog. Sequential.
- ✓ Pada pemrograman prosedural, programmer harus menentukan sesungguhnya bagaimana pemecahan masalah harus di-code-kan.
- ✓ Pembuat code adalah pemrograman non prosedural.

2. Paradigma Non-Prosedural



- ✓ Penekanan pemrograman Non prosedural adalah penentuan apa yg akan diselesaikan dan membiarkan system menentukan bagaimana menyusunnya.

- ➢ **PEMROGRAMAN DEKLARATIF**

  Memisahkan tujuan dari metode yg digunakan untuk mencapai tujuan.

- ➢ **PEMROGRAMAN OBJECT ORIENTED**

  Ide : membuat dsign program dg mempertimbangkan data yg digunakan dalam program sebagai objek dan mengimplemnetasikan operasi pada objek tersebut.

- ➢ **PEMROGRAMAN LOGIKA**

  Pembuktian teori logika dg Logic Theorirt Program (Newell & Simon) pada Darmouth Conference A.I (1956)

Rangkaian *backward* dapat digunakan untuk mengekspresikan pengetahuan dalam representasi deklaratif maupun kontrol proses pemberian alasan.

Keuntungannya : pembuatannya dapat diproses secara paralel yaitu jika ada beberapa processor dapat bekerja secara simultan.

EXPERT SYSTEM

- *Disebut pemrograman deklaratif krn programmer tdk menetukan bagaimana prog. hrs mendapatkan tujuannya pada level algoritma*

ANS (ARTIFICIAL NEURAL SYSTEMS)

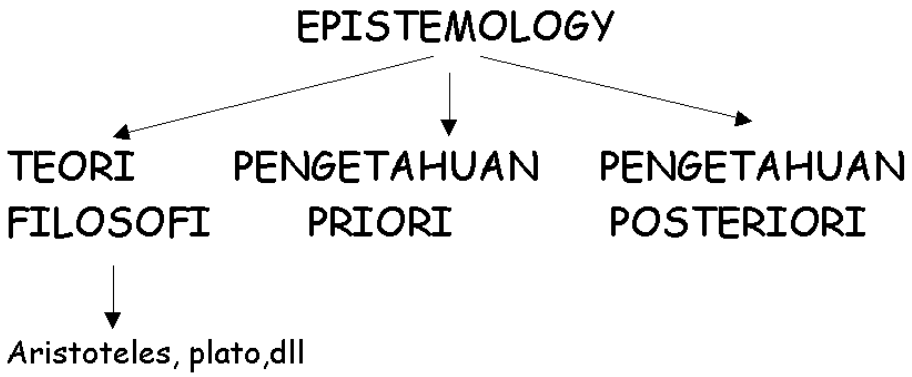
BAGIAN 2

REPRESENTASI PENGETAHUAN (1)

Representasi pengetahuan merupakan hal penting dalam SP karena:

1. Shell SP didesign untuk type representasi pengetahuan tertentu seperti baris dan logika
2. Akan memberikan efek/akibat pengembangan, efisiensi, kecepatan dan perawatan system.

Study pengetahuan disebut ***epistemology***



PENGETAHUAN PRIORI

- ✓ Berasal dari bahasa Latin
- ✓ Berarti yang mendahului (pengetahuan datang sebelumnya dan bebas dari arti)
- ✓ Contoh pernyataan “segalanya memiliki sebab” , “ seluruh triangle dalam pesawat mempunyai 180 derajat”
- ✓ Contoh lain pernyataan logika, hukum matematika
- ✓ Disebut secara universal benar dan tidak dapat ditentukan tanpa kontradiksi.

PENGETAHUAN POSTERIORI

- ✓ Adalah pengetahuan yg diperoleh dari arti
- ✓ Kebenaran dari pengetahuannya menggunakan pengalaman.

PENGETAHUAN PROSEDURAL

✓ Bagaimana melakukan sesuatu

PENGETAHUAN DEKLARATIF

✓ Mengacu pada sesuatu benar atau salah

PENGETAHUAN TACIT/UNCONSCIUS

✓ Tidak dapat diekspresikan dg bahasa**.**

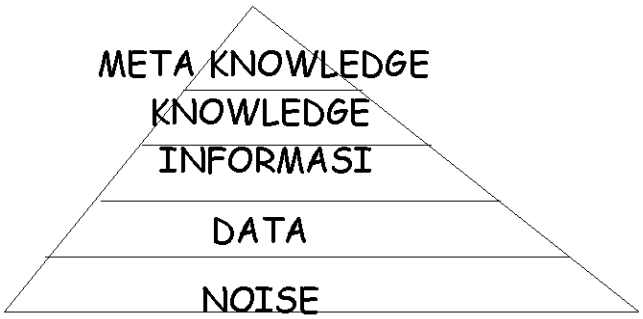
ANALOGY EKSPRESI PENGETAHUAN MENURUT WIRTH

Algorithms + Data Structures = Programs

Untuk SP :

Knowledge + Inference = Expert Systems

HIRARKI PENGETAHUAN :



SP juga :

1. Memisahkan data dari noise
2. Mentrasformasikan data ke dalam informasi
1. Mentrasformasikan data ke dalam pengetahuan

METAKNOWLEDGE

- ✓ Adalah pengetahuan mengenai beberapa perbedaan domain
- ✓ Menentukan basis pengetahuan mana yg sesuai

PRODUKSI

Teknik representasi pengetahuan mencakup : baris, jaringan semantik, frame, scrips, bahasa representasi pengetahuan (spt KL-1)

BNF (BACKUS NAUR FORM)

- ✓ Format notasi untuk menentukan produksi yaitu metalanguage yaitu untuk menentukan syntax bahasa.
- ✓ Metalanguage diatas bahasa normal (meta berati diatas)
- ✓ Type bahasa : bahasa natural, bahasa logika, matematika, bahasa komputer
- ✓ Notasi BNF sderhana : kalimat yg berisi kata benda dan kata kerja diikuti oleh titik
- ✓ Baris produksi :
  <sentence> ::= <subject><verb><end-mark>
- ✓ < > dan ::= merupakan simbol dari metalanguage
- ✓ ::= berarti "ditentukan sebagai" (sama dg →)
- ✓ < > simbol nonterminal (yaitu variabel yg menunjukkan bentuk lain)
- ✓ simbol I berarti atau

Contoh :

<sentence> → <subject> <verb> <end-mark>

<subject> → I I You I We

<verb> → left I came

<end-mark> → . I ? I !

Produksinya ?……..

- ✓ Serangkaian terminal disebut ***string***
- ✓ **Kalimat valid** : jika string didapatkan dari start simbol dg menggantikan nonterminal dg baris definisinya.
- ✓ **Grammar** : set/rangkaian baris produksi lengkap yg menentukan suatu bahsa secara tidak ambigius.

- ✓ **Grammar valid** :

  <sentence> → <subject> <verb> <object> <end-mark>

Contoh :

<sentence> → <subject phrase> <verb> <object phrase>

<subject phrase> → <determiner> <naun>

<object phrase> → <determiner> <adjective> <naun>

<determiner> → a I an I this I these I those

<noun> → man I eater

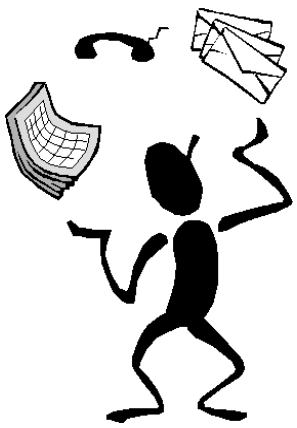
<verb> → is I was

<adjective> → dessert I heavy

Bagaimana derivative tree ?…

- ✓ <**determine**> digunakan untuk menunjukkan item tertentu
- ✓ **Parse tree atau derivation tree** adalah representasi grafik dari kalimat yg diuraikan ke dalam seluruh terminal dan nonterminal yg digunakan untuk mendapatkan kalimat.

REPRESENTASI PENGETAHUAN (2)



- Representasi Pengetahuan (*Knowledge Representation*) dimaksudkan untuk menangkap sifat-sifat penting masalah dan membuat infomasi dapat diakses oleh prosedur pemecahan masalah.

- Bahasa representasi harus dapat membuat seorang programmer mampu mengekspresikan pengetahuan untuk mendapatkan solusi suatu masalah.
- Secara singkat Mylopoulos dan Levesque mengklasifikasikan susunan atau pola represen-tasi menjadi empat katagori :

1. Representasi Logika

Representasi ini menggunakan ekspresi-ekspresi dalam logika formal untuk merepresentasikan basis pengetahuan.

2. Representasi Prosedural

Menggambarkan pengetahuan sebagai sekumpulan instruksi untuk memecahkan suatu masalah. Dalam sistem yang berbasis aturan, aturan if-then dapat ditafsirkan sebagai sebuah prosedur untuk mencapai tujuan pemecahan masalah.

3. Representasi Network

Menyatakan pengetahuan sebagai sebuah graf dimana simpul-simpulnya menggambarkan obyek atau konsep dalam masalah yang dihadapi, sedangkan lengkungannya menggambarkan hubungan antar mereka. Contohnya adalah jaringan semantik.

4. Representasi Terstruktur

Memperluas network dengan cara membuat setiap simpulnya menjadi sebuah struktur data kompleks yang berisi tempat-tempat bernama slot dengan nilai-nilai tertentu. Nilai-nilai ini dapat merupakan data numerik atau simbolik sederhana, pointer ke bingkai (frame) lain, atau bahkan merupakan prosedur untuk mengerja kan tugas tertentu. Contoh : skrip (script), bingkai (frame) dan obyek (object).

REPRESENTASI LOGIKA

Representasi logika terdiri dari dua jenis yaitu *Kalkulus proposisional* (*Propositional logic*) dan *Kalkulus predikatif* (*Predicate logic*).

Kalkulus Proposisional (Propositional Logic)

- Proposisi adalah suatu model untuk mendeklarasikan suatu fakta. Lambang-lambang proposisional menunjukkan proposisi atau pernyataan tentang segala sesuatu yang dapat benar atau salah.

Lambang-lambang kalkulus proposisional :

1. Lambang pernyataan proposisional

   P,Q,R,S,T,... (disebut sebagai atom-atom)

2. Lambang kebenaran

   benar (*True*) , salah (*False*)

3. Lambang penghubung

   $\wedge$ (konjungsi), $\vee$ (disjungsi), $\sim$ (negasi),
   $\rightarrow$ (implikasi), $\leftrightarrow$ (Bi-implikasi),
   $\equiv$ (equivalen)

Berikut ini adalah tabel kebenaran (*truth value*) lambang penghubung :

P	Q	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Equivalen

Suatu kalimat (formula) P dianggap equivalen dengan formula Q jika dan hanya jika *'truth value*' dari P sama dengan '*truth value*' dari G untuk setiap interpretasinya. (ditulis sbg. $P \equiv Q$)

Contoh :

$P \rightarrow Q \equiv \sim P \vee Q$

P	Q	$\sim P$	$P \rightarrow Q$	$\sim P \vee Q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

- Kalimat-kalimat atau formula dalam kalkulus proposisional dibentuk dari lambang-lambang dasar tersebut.

- Nilai-nilai kebenaran yang dikandung oleh kalimat-kalimat proposisional disebut *interpretasi*.

- Secara formal, interpretasi diartikan sebagai pemetaan dari lambang-lambang proposisional menuju ke himpunan {T,F} yakni himpunan 'benar-salah'.

- Suatu formula (kalimat) yang mempunyai n lambang (atom) yang berbeda, mempunyai 2n interpretasi.

- Interpretasi yang menyebabkan suatu formula bernilai benar dikatakan *satisfy the formula.*

- Suatu formula dikatakan *tautology* jika dan hanya jika bernilai benar untuk setiap interpretasinya.

  Contoh : (A $\vee$ ~A).

- Suatu formula dikatakan *inconsistency* jika dan hanya jika bernilai salah untuk setiap interpretasinya.

  Contoh : (A $\wedge$ ~A).

- Suatu formula dikatakan *consistent* jika tidak *inconsistent*. Dengan kata lain, suatu formula yang *consistent*, paling tidak ada satu interpretasi yang benar. Contoh (((B $\vee$ C) $\wedge$ ~C) $\vee$ D).
- Jika suatu *formula tautology* maka *consistent*, tetapi tidak berlaku sebaliknya.
- *Tautology* disebut juga *valid formula*
- *Inconsistency* disebut juga *unsatisfiable formula*
- *Consistency* disebut juga *satisfiable formula*

Hukum yang berlaku untuk ekspresi proposisional P,Q dan R adalah :

1.Hukum de Morgan : $\sim(P \vee Q) \equiv (\sim P \wedge \sim Q)$

2.Hukum de Morgan : $\sim(P \wedge Q) \equiv (\sim P \vee \sim Q)$

3.Hukum distributif :

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

4.Hukum distributif:

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \vee R)$$

5.Hukum komutatif : $(P \wedge Q) \equiv (Q \wedge P)$

6.Hukum komutatif : $(P \vee Q) \equiv (Q \vee P)$

7.Hukum asosiatif :

$$((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$$

8.Hukum asosiatif :

$$((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$$

9.Hukum kontrapositif :

$$(P \rightarrow Q) \equiv (Q \rightarrow \sim P)$$

Prosedur Pembuktian Teorema

- Suatu formula G dikatakan sebagai sebuah konsekuensi logis dari formula F1, F2, … , Fn jika dan hanya jika setiap interpretasi yang memenuhi (F1∧F2∧ …∧Fn) juga memenuhi G.

  F1, F2, … , Fn disebut premis

  G disebut Goal formula
- Dengan kata lain, formula G adalah konsekuensi logis dari premis F1, F2, … , Fn jika dan hanya jika ((F1∧F2∧ … ∧Fn) → G) adalah *Tautology*.
- Karena negasi dari suatu *Tautology* adalah *Inconsistency*, maka ~((F1∧F2 ∧ … ∧Fn) → G) adalah *Inconsistency*.
- Kita tahu bahwa

  ~((F1∧F2∧ … ∧Fn)→G) ≡

  ~(~(F1∧F2∧ … ∧Fn) ∨ G) ≡

  (F1∧F2∧ … ∧Fn) ∧~G)

- **Dua Metode Pembuktian Teorema**:

1. Metode Langsung (*Direct Method*) membuktikan bahwa ((F1∧F2∧ … ∧Fn) →G) adalah *Tautology*.
2. Metode *Refutasi* membuktikan bahwa :

   (F1∧F2∧…∧Fn)∧~G) adalah *Inconsistency*.

Contoh soal:

Buktikan bahwa Q adalah konsekensi logis dari premis P dan (P → Q) !

Solusi:

1. Metode Langsung, membuktikan bahwa

   ((P∧(P → Q)) → Q) adalah *Tautology.*

P	Q	P→Q	P ∧ (P→Q)	(P∧ (P→Q)) →Q
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

2. Metode *Refutasi*, membuktikan bahwa
(P∧(P→Q)∧~Q) adalah *Inconsistency*.

P	Q	~Q	P→Q	P∧(P→Q)	P∧(P→Q)∧~Q
T	T	F	T	T	F
T	F	T	F	F	F
F	T	F	T	F	F
F	F	T	T	F	F

Rules of Inference **(Aturan-aturan Inferensi)**

- Pendekatan lain untuk membuktikan teorema yang menggunakan aturan/rule (dinamakan *Rules of inference*), adalah dengan cara mendeduksi konsekeunsi logis dari premis-premis yang diketahui atau diberikan.

- Beberapa contoh *Rules of Inference* adalah:

1. Introducing Conjunction

If F and G then (F∧G)

2. Eliminating Conjunction

If (F∧G) then F

If (F∧G) then G

3. Introducing Disjunction

If F then (F∨G)

If G then (F∨G)

4. ***Modus Ponens***

   If F and (F → G) then G

5. ***Modus Tollens***

   If ~G and (F → G) then ~F

6. ***Chaining***

   If (F→G) and (G→H) then (F→H)

7. **Equivalen**

   If F and (F ≡ G) then G

   If G and (F ≡ G) then F

Contoh soal:

Bila diberikan premis-premis sebagai berikut:

(i) *John awakens*

(ii) *John brings a mop*

(iii) *Mother is deligthed, if john awakens and cleans his room*

(iv) *If John brings a mop, then he cleans his room*.

Buktikan dengan *Rules of Inference* (deduksi), dimana *goal*-nya adalah : *Mother is deligthed !*

Solusi :

Tuliskan premis tersebut sebagai simbol (atom):

A = *John awakens*

B = *John brings a mop*

C = *John cleans his room*

D = *Mother is delighted*

Goal yang ingin dibuktikan adalah D

Tuliskan premis tersebut sebagai formula:

(1) A

(2) B

(3) A∧C → D

(4) B → C

Deduksi dengan *Rules of Inference*

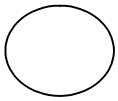
(5) C (dng. *Modus Ponens* (2) dan (4))

(6) A∧C (dng. *Intro. Conjunction* (1) dan (5))

(7) D (dng. *Modus Ponens* (3) dan (6))

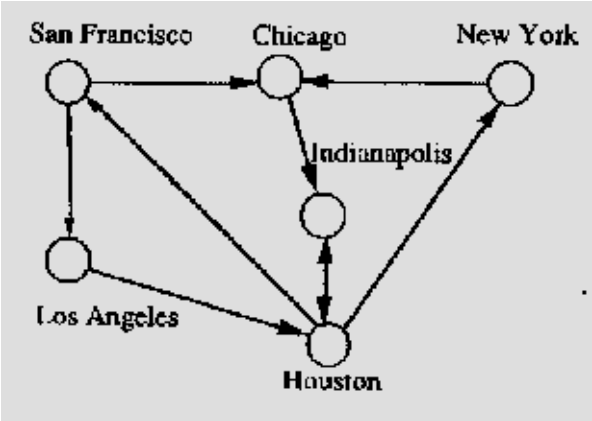
JARINGAN SEMANTIK

- ✓ Sering disebut *proportional net*
- ✓ Bentuk dari pengetahuan deklaratif krn proporsi trb menunjukan fakta
- ✓ Proporsi selalu benar atau salah disebut juga "*atomic*"
- ✓ Merupakan gambaran grafis yg menunjukkan hubungan antar berbagai objek. Yaitu dlm bentuk "*nodes*" dan "*arcs*" yg menghubungkannya



✓ → Nodes disebut juga dgn objek, digunakan untuk menunjukkan objek phisik, konsep, situas

- ✓ → : Links atau edges atau arcs, untuk mengekspesikan suatu relasi
- ✓ Contoh route pesawat terbang (*directed graph*)



- ✓ Disebut juga **associative nets**, krn node dihubungkan dg yang alin.
- ✓ Bentuk links **IS-A, HAS-A,A-KIND-OF (AKO)**
- ✓ **IS-A** menunjukkan hubungan kelas, pada gbr diatas menunjukkan "jarak dari"
- ✓ **HAS-A** digunakan untuk mengidentifikasi karakteristik atau atribut objek noda.
- ✓ **AKO** digunakan untuk menghubungkan satu jenis ke jenis yg lain

- ✓ Salah satu masalah pd jaringan semantik adalah tidak adanya standar definisi nama *link*
- ✓ ***Object-attribute-value triple (OAV) atau triplet*** digunakan untuk memberi karakter semua pengetahuan dlm jaringan semantik
  - *Object* dapat berupa fisik atau konsepsi
  - *Attribute* adalah karakteristik dari object
  - *Values* adalah ukuran spesifik dari *attribute* dalam situasi tertentu.

Contoh: O-A-V item

Object	**Attribute**	**Values**
Rumah	Kamar tidur	2, 3, 4 dst.
Rumah	Warna	Putih, Biru,dst.
Kamar tidur	Ukuran	2x3, 3x3, 3x4, dst
Diterima di universitas	Nilai ujian masuk	A, B, C, atau D

Bahasa PROLOG

- ✓ Model : Pemrograman Logika
- ✓ Jenis data : Simbolik dan numerik, predikat, list

Kalkulus Predikatif	**PROLOG**	**Arti**
$\cap$	,	dan
$\cup$	;	atau
$\leftarrow$	:-	menyebabkan
+	not	tidak,bukan

- ✓ Variabel dinyatakan sebagai string karakter alfanumerik dimulai dengan huruf besar :

  likes(X,ana)

- ✓ Contoh :

  likes(doni,tina),likes(doni,ana)

likes(doni,ana) :- likes(doni,tina)

not(likes(tina,ana))

- ✓ Contoh :

  Proposisi : Mobil berada didalam garasi

  Kalkulus predikat : didalam (mobil,garasi)

- ✓ Contoh lain :
  1. red is a color
  2. Tom is the father of John
  3. Tom and Susan are the parents of John

Jawab :

1. color (red)
2. father_of (Tom,John)
3. parents (Tom,Susan<John)

SCHEMATA

- ✓ Jaringan semantik contoh dari *Shallow knowledge Structure* karena seluruh pengetahuan jaringan semantik diisikan dalam *link* dan *node*
- ✓ *Concept schema* : dengan skema tsb kita dapat menunjukkan konsep.
- ✓ Contoh konsep mengenai binatang, setiap orang mempunyai persepsi sendiri mengenai bintang (berkaki 4 atau 2 , berbulu atau bersisik dsb)

- Banyak pengetahuan yang digunakan dalam proses penalaran kita sehari hari yang sudah pasti dan sudah dikenal dengan baik. Hal ini didasarkan kepada berbagai penampilan situasi dan objek-objek khusus, dan proses yang tak bervariasi.

- Pengetahuan semacam itu kita sebut pengetahuan *stereotype*.

- Skema adalah satu metoda pengorganisasian, presentasi dan penggunaan pengetahuan *stereotype* agar komputer bisa menalar

BINGKAI (FRAME)

- Dengan menggunakan representasi *network*, kita melihat pengetahuan diatur dengan menggunakan penghubung antar obyek dalam basis pengetahuan. Selain itu, kita dapat mengatur pengetahuan ke dalam unit-unit yang lebih kompleks yang menggambarkan situasi atau obyek yang rumit dalarn domain. Unit-unit ini disebut bingkai (*frame*).
- Menurut Minsky, bingkai dapat dipandang sebagai struktur data statik yang digunakan untuk merepresentasikan situasi-situasi yang telah dipahami dan stereotip
- Setiap bingkai individual dapat dipandang sebagai sebuah struktur data yang dalam banyak hal serupa dengan "*record*", dan berisi infomasi yang relevan dengan entitas-entitas stereotip.
- Bingkai mempermudah kita untuk mengatur pengetahuan kita secara hirarki.
- Adalah blok pengetahuan yang relatif besar atau kumpulan pengetahuan tentang suatu objek tertentu, peristiwa, lokasi, situasi atau elemen-elemen lainnya.
- Rinciannya diberikan ke dalam slot yang menggambarkan berbagai atribut dan karakteristik objek.
- *Frame* biasanya digunakan untuk merepresentasikan pengetahuan stereotype atau pengetahuan yang didasarkan kepada karakteristik yang sudah dikenal yang merupakan pengalaman-pengalaman.
- Dalam bentuk fisik, *frame* merupakan suatu gambaran seperti "garis besar" yang sudah dikatagorikan dan sub katagori.
- Slot menggambarkan atribut seperti nama pabrik, model, asa-usul pabrik, jenis mobil, jumlah pintu, mesin dan karakteristik lainnya
- Beberapa slot mempunyai nilai tetap.
- Jenis slot lainnya bersifat prosedural. Hal ini merupalan slot yang memungkinkan penambahan informasi baru yang bisa ditambahkan pada kaidah dasar IF
- Hampir semua sistem *artificial intelligence* terbuat dari kumpulan frame-frame yang dalam hal ini satu sama lain saling berhubungan. Secara bersama-sama mereka (frame-frame) membentuk suatu hirarki yang dapat digunakan untuk maksud penalaran.
- Untuk menggunakan sistem frame, kita harus membuat program rrame itu sendiri dengan menggunakan bahasa pemograman Al.

(CATT : Contoh mekanik Mobil)

SKRIP (*SCRIPT*)

- Skrip (*script*) merupakan representasi terstruktur yang menggambarkan urutan stereotip dari kejadian-kejadian dalam sebuah konteks khusus.
- Skrip mula-mula dirancang oleh Schank dan kelompok risetnya sebagai alat pengorganisasi struktur-struktur *ketergantungan konseptual* menjadi deskripsi khusus.
- Script adalah skema representasi pengetahuan yang sama dengan frame.
- Perbedaannya ialah, frame menggambarkan objek, sedang script menggambarkan urutan peristiwa.
- Dalam mengambarkan urutan peristiwa, script menggunakan serangkaian slot yang berisi informasi tentang orang, objek, dan tindakan-tindakan yang teijadi dalam suatu peristiwa.
- Elemen *script* yang tipikal termasuk kondisi masukan*, prop, role* dan *scene*.

  - Kondisi masukkan menggambarkan situasi yang harus dipenuhi sebelum terjadi atau berlaku suatu peristiwa yang ada dalam script.
  - Prop mengacu kepada objek yang digunakan dalam urutan peristiwa yang terjadi.
  - Role mengacu kepada orang-orang yang terlibat dalam script.
  - Scene menggambarkan urutan peristiwa aktual yang terjadi.
- Komponen-komponen skrip adalah
  - ✓ Kondisi entri atau deskriptor dunia sekitar kita yang harus benar agar skrip dapat dipanggil. Contoh : dalam hal skrip restoran, ini mencakup restoran yang sedang buka dan pelanggan yang sedang lapar.

- ✓ *Hasil atau fakta yang benar* begitu skrip diakhiri. Misalnya, pelanggan sudah kenyang, dan pemilik restoran memiliki uang yang lebih banyak (karena pembayaran oleh pelanggan tersebut).
- ✓ *Penyangga* atau apa-apa yang merupakan isi skrip. Di sini meliputi meja, kursi, pelayan, dan menu.

- ✓ *Peran* adalah tindakan yang dilakukan oleh partisipan individual. Misalnya, pelayan yang mengantar pesanan, dan memberikan tagihan pada pelanggan, serta pesanan pelanggan, makan, dan membayar.
- ✓ Adegan yang merupakan kejadian yang menunjukkan aspek waktu dari skrip. Di sini dapat berupa : masuk ke restoran, memesan, makan, dan lain-lain.

Predicate Logic (Predicate Calculus)

Abjad yang menyusun lambang-lambang kalkulus predikatif terdiri dari :

1. Rangkaian huruf, baik huruf kecil maupun huruf besar dari abjad.
2. Rangkaian digit 0,1,...,9.
3. Garis bawah _.

Lambang-lambang kalkulus predikatif *dimulai dengan huruf dan diikuti oleh sembarang rangkaian karakter yang diperkenankan*

Contoh :
Tono
***98
ya2
ati_dan_tita
YYY
9rumah
“mata”
yach????
&kita

Untuk lambang variabel : dimulai dg huruf besar untuk merancang kelas objek atau sifat yang umum
Contoh : Gumam, POHON

Untuk lambang fungsi/konstanta : dimulai dg huruf kecil
Contoh : gumam, pohon

Kalimat dasar dalam kalkulus predikatif adalah predikat yang diikuti dengan istilah yang berada didalam tanda kurung dan dipisahkan oleh koma. Kalimat kalkulus predikatif dibatasi oleh titik/periode.

Contoh : - *likes*(ani,ida)
- *helps*(anton,tono)

Lambang predikat dalam contoh di atas adalah likes dan helps.
Lambang-lambang predikat merupakan lambang yang dimulai dg huruf kecil.

Kalkulus predikatif juga mencakup dua lambang, $\forall$ dan $\exists$ yang membatasi arti sebuah kalimat. $\forall$ merupakan *penguantifikasi universal* yang menunjukkan bahwa suatu kalimat adalah benar *untuk semua nilai variabelnya*. Sedangkan $\exists$ merupakan *penguantifikasi eksistensial* yang menunjukkan bahwa suatu kalimat adalah benar *untuk suatu nilai tertentu dalam sebuah domain*.

QUANTIFIER UNIVERSAL ($\forall$)

- Menyatakan " untuk setiap" atau "untuk semua"
- Contoh : p menunjukkan kalimat seluruh kucing adalah binatang

  $(\forall x)$ (p) $\equiv$ $(\forall x)$ (if x adalah seekor kucing $\rightarrow$ x adalah seekor binatang)

  atau

  $(\forall x)$ (*x is a cat* $\rightarrow$ *x is a animal*)

  - ✓ Negasi : $(\forall x)$ (p) $\equiv$ $(\forall x)$ (if x is a cat $\rightarrow$ ~x is a animal)
  - ✓ Bagaimana kalimat matematika untuk "seluruh segitiga adalah poligon" ?
  - ✓ Bagaimana Predicate Funcition ?

    $(\forall x)$ (*x is a triangle* $\rightarrow$ *x is a polygon*)

    $(\forall x)$ (*triangle (x)* $\rightarrow$ *polygon (x)*)

  - ✓ Fungsi Predikat dituliskan dg notasi yg lebih singkat dg huruf besar

- ✓ Contoh : T = *triangle* dan P = *Polygon*

  $(\forall x)\ (T(x) \rightarrow P(x))$
- ✓ Dapat diintepretasikan sebagai konjungsi

EXISTENTIAL QUANTIFIER ($\exists$)

- Suatu pernyataan benar untuk minimal satu anggota domain
- Dibaca "*there exists*", "*at least one*", "*for some*", "*there is one*" , "*some*"
- Contoh :

  P=gajah

  Q=binatang mamalia

  Semua gajah adalah mamalia $(\forall x)\ (P(x) \rightarrow Q(x))$

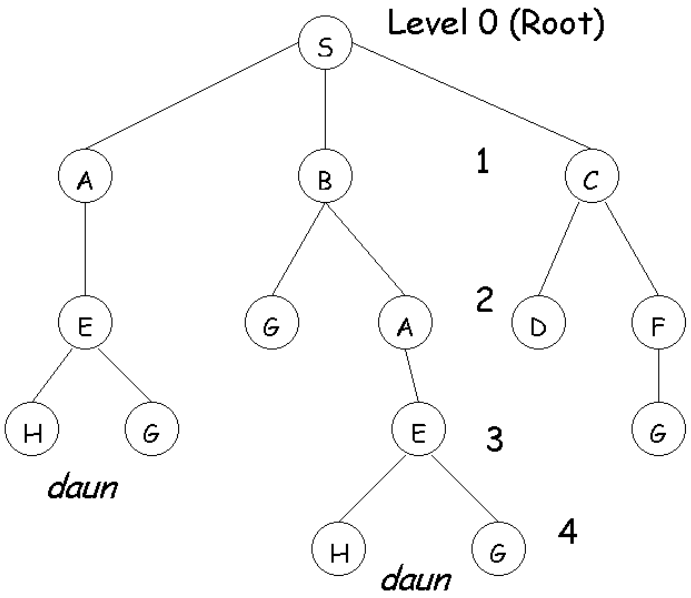
  Beberapa gajah bukan mamalia $(\exists x)\ (P(x) \rightarrow \sim Q(x))$

Beberapa gajah adalah mamalia $(\exists x)\ (P(x) \rightarrow Q$

Set Expression	*Logical Equivalent*
$A = B$	$\forall x\ (x \in A \leftrightarrow x \in B)$
$A \subseteq B$	$\forall x\ (x \in A \rightarrow x \in B)$
$A \cap B$	$\forall x\ (x \in A \wedge x \in B)$
$A \cup B$	$\forall x\ (x \in A \vee x \in B)$
A'	$\forall x\ (x \in \upsilon \mid \sim (x \in A))$
υ (Universe)	T (True)
$\emptyset$ (empty set)	F (False)

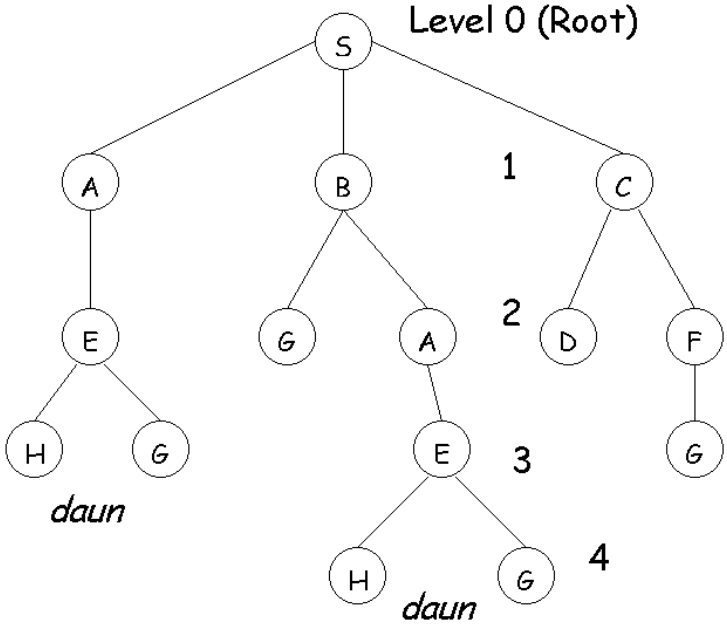
Set	*Logic*
$(A \cap B)' \equiv A' \cap B'$	$\sim(p \wedge q) \equiv \sim p \vee \sim q$
$(A \cup B)' \equiv A' \cap B'$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$

STATE GRAPH



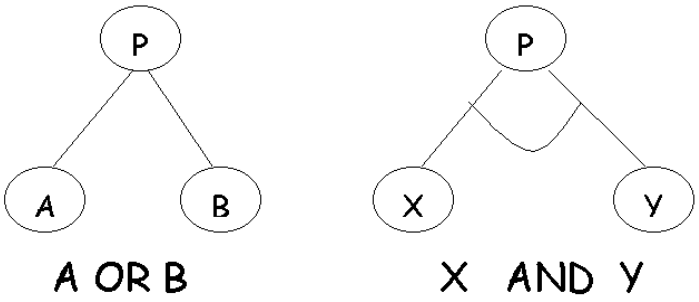
- Peta yg menunjukkan berbagai kota antara kota – kota yg akan dilaluinya agar mencapai kota tujuan yg diinginkan lebih cepat.
- A … H : Node
- ⟶ : Ark/link

POHON PELACAKAN



- Level → sbg hirarki (menggambarkan kedalaman pohon)
  Node merupakan berbagai keadaan dlm ruang pelacakan. Ark → operatornya

POHON AND / OR



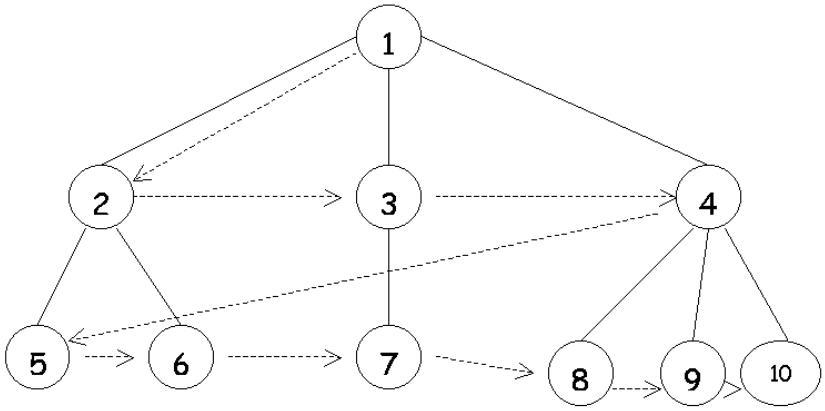
METODE PELACAKAN

1. BLIND SEARCH (Pelacakan Buta)

- ✓ Merupakan sekumpulan prosedur yg digunakan dlm melacak ruang keadaan.
- ✓ Menguji seluruh pohon dgn cara yg teratur dg menggunakan semua operator shg menghasilkan suatu solusi.
- ✓ Lebih tepat u/ soal-soal kecil dg beberapa ruang keadaan dan tepat u/ komputer berkecepatan tinggi.

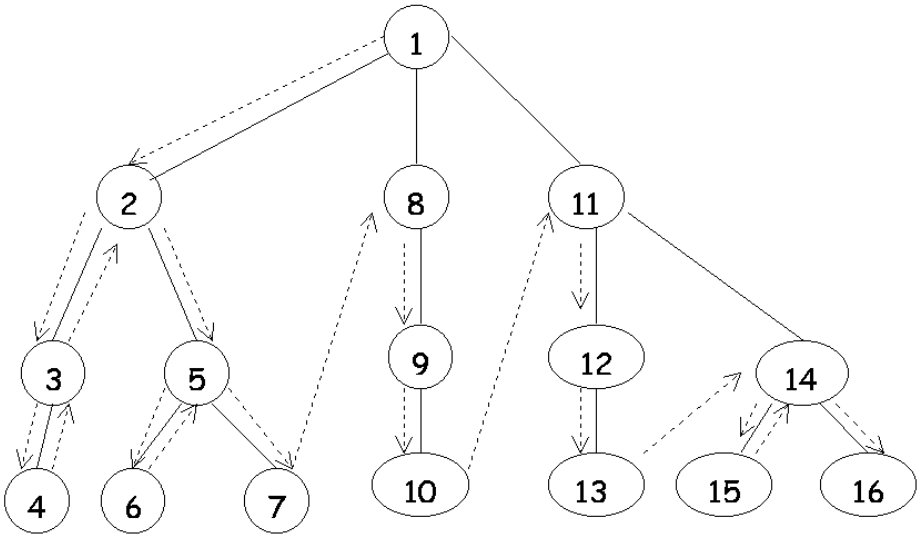
2. BREADTH FIRST (Pelacakan Melebar Pertama)

- ✓ Menguji semua node dlm pohon pelacakan mulai dari node akar
- ✓ Node yg ada pd setiap tingkat seluruhnya diuji sebelum pindah ke tingkat berikutnya.



3. DEPTH FIRST (Pelacakan Pertama Mendalam)

- ✓ Jika keadaan tujuan tidak tercapai maka proses dilakukan dg jalan pelacakan backtrak ke node sebelumnya.
- ✓ Menjamin bisa menemukan solusi tapi waktu pelacakannya lama.
- ✓ Masalah utama : sering terjadi penyimpangan arah node tujuan yg sebenarnya.



4. HEURISTIC SEARCH

- ✓ Istilah yg berasal dari bahasa Yunani yg berarti “menemukan / menyingkap “
- ✓ Membantu mengurangi wilayah pelacakan yg bisa menimbulkan berbagai alternatif solusi shg dapat membimbing ke tujuan yg diinginkan.

5. HILL CLIMBING (Mendaki bukit)

- ✓ Merupakan pelacakan depth first yg memanfaatkan heuristik u/ menentukan jarak yg terpendek atau biaya terendah menuju tujuan yg diinginkan.

6. BEST FIRST SEARCH (Pencarian Terbaik Pertama)

- ✓ Kombinasi dari breadth first dan depth first

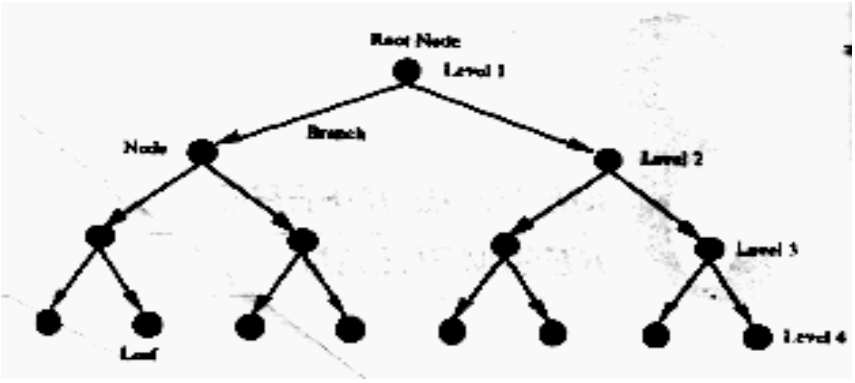
BAGIAN 3

METODE INFERENSI / KESIMPULAN

TREES,*LATTICES* DAN GRAF

Tree: struktur data hirarki yg berisi node/*vertices*/objek yg menyimpan informasi/pengetahuan dan link/*edges*/cabang yg menghubungkan node

- Disebut juga dg tipe jaringan semantik khusus
- Merupakan kasus khusus yg disebut graf
- Suatu graf dapat mempunyai nol atau lebih link, dan tidak ada perbedaan antara root dan child
- Root : node tertinggi, leaves : terendah



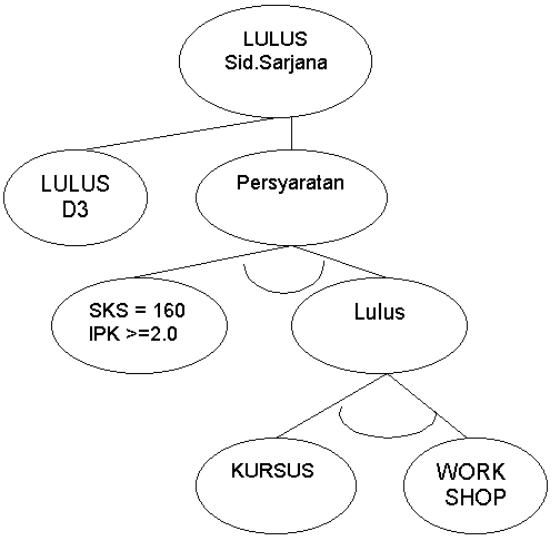
- ✓ **Stuktur keputusan** : skema representasi pengetahuan dan metode pemberian alasan tentang pengetahuannya.
- ✓ Jika suatu keputusan adalah binary, maka tree keputusan binary mudah dibuat dan sangat efisien.
- ✓ Setiap pertanyaan, turun satu tingkat dalam tree.Jika seluruh leaves adalah jawaban dan seluruh node yg turun adalah pertanyaan, maka ada max 2n untuk jawaban dan n pertanyaan

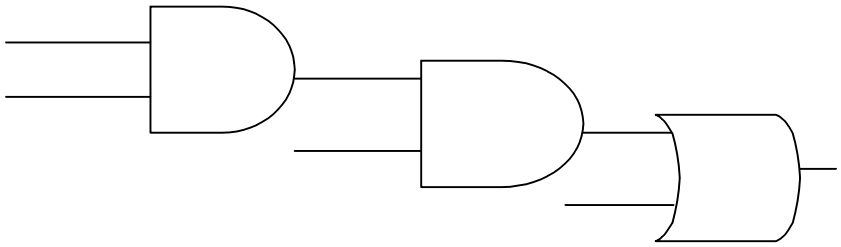
STATE SPACE

- ✓ *State* adalah kumpulan karakteristik yg dapat digunakan untuk menentukan status.
- ✓ *State Space* adalah rangkaian pernyataan yg menunjukkan transisi antara state dimana objek diekspreriman

POHON AND-OR

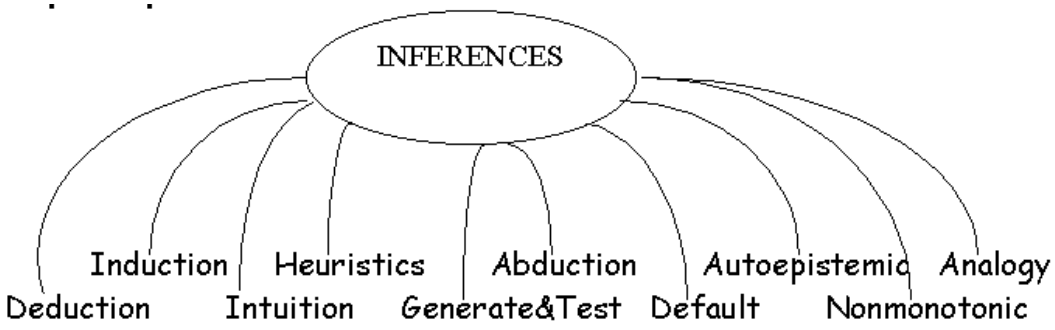
Dalam SP, untuk menemukan solusi problem dapat menggunakan rangkaian backward yaitu dengan tree AND-OR dan AND-OR-NOT





LOGIKA DEDUKTIF DAN SILOGISME

Tipe-tipe inferensi



Deduction

- Pemberian alasan logikal dimana kesimpulan harus mengikuti premis

Induction

- Inferensi dari khusus ke umum

Intuition

- Tidak ada teori yg menjamin. Jawabannya hanya muncul, mungkin dengan penentuan pola yg ada secara tidak disadari.

Heuristic

- Aturan yg didasarkan pada pengalaman

Generate & Test

- Trial dan error. Digunakan dgn perencanaan.

Abduction

- Pemberian alasan kembali dari kesimpulan yg benar ke premis .

Default

- Diasumsikan pengetahuan umum sebagai default

Autoepistemic

- Self-knowledge

Nonmonotonic

- Pengetahuan yg sebelumnya mungkin tdk benar jika bukti baru didapatkan

Analogy

- Kesimpulan yg berdasarkan pada persamaan untuk situasi yg lainnya.

Yang paling sering dipakai : deductive logic, unruk menentukan validitas “argument”.

Silogisme merupakan satu type argumen logika.

Contoh :

Premise : Anyone who can program is intelligent

Premise : John can program

Conclusion : Therefore, John is intelligent

Premise

- Digunakan sebagai bukti untuk mendukung sutu kesimpulan.
- Disebut juga antecedent

Kesimpulan/Conclusion

- Disebut juga ***consequent***

Karakteristik logika deduktif adalah *kesimpulan benar harus mengikuti dari premis yg benar*

Anyone who can program is intelligent
John can program
_____
∴ John is intelligent

Dalam bentuk IF-THEN
IF Anyone who can program is intelligent And
John can program
THEN John is intelligent

Silogisme klasik disebut ***categorical syllogism***.
Premis dan kesimpulan ditentukan sebagai statement categorical dari 4 bentuk berikut :

FORM	SCHEMA
A	All S is P
E	No S is P
I	Some S is P
O	Some S is not P

S : Subjek kesimpulan disebut minor term
P : Predikat kesimpulan disebut major term

Major premise : All M is P
Minor premise : All S is M
_____
Concluusion : All S is P

Silogisme diatas disebut *standard form* dimana major dan minor premis diidentifikasi.

Categorical Silogisme

- **A dan I** disebut “affirmative in quality” , subjek dimasukkan kedalam jenis predikat
- **E dan O** disebut “negative in quality”, subjek tidak masuk dalam jenis predikat
- **IS = capula =** menghubungkan, menunjukkan bentuk tense dari kata kerja “tobe”
- **Middle term (M)**
- **All** dan **No** : *universal quantifier*, ***Some*** :*particular quantifier*
- **Mood** silogisme ditentukan dengan 3 huruf yg memberikan bentuk premis pokok, minor premis, dan kesimpulan.

	Figure 1	**Figure 2**	**Figure 3**	**Figure 4**
Major Premise	M P	P M	M P	P M
Minor Premise	S M	S M	M S	M S

Contoh :

All M is P
All S is M } type AAA-1
∴ All S is P

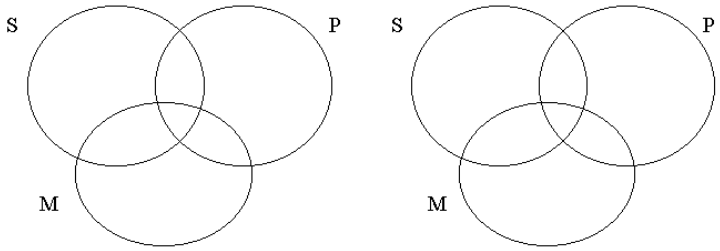
All M is P
No S is M } type ????
∴ No S is P

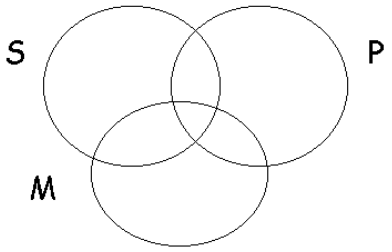
Some P are M
All M are S } type ????
∴ Some S are P

Untuk membuktikan validitas argumen silogisme, ada metode yang dinamakan “decision prosedure” yaitu dengan menggunakan diagram venn.

Contoh :

All M is P
All S is M } type AAA-1
∴ All S is P





BARIS INFERENCE (RULES OF INFERENCE)

Yaitu modus ponens dan modus tollens

Diagram venn tidak sesuai untuk argumen yg lebih kompleks karena menjadi sulit untuk dibaca pada decision tree untuk silogisme

Pada logika proposisional,

If there is power, the computer will work

There is power

$\therefore$ The computer will work

Maka dapat ditulis

$A \rightarrow B$		$p \rightarrow q$		
A	$\equiv$	p	$\equiv$	$p, p \rightarrow q; \therefore q$
$\therefore B$		$\therefore q$		

$\rightarrow$ disebut "*direct reasoning,modus ponenes, law of detachment dan assuming the antecedent*"

p,q disebut variabel logika

A,B disebut konstanta proposisional

Bagaimana dengen skema untuk argumen dari tipe ini :

1. $p \rightarrow q$ 2. $p \rightarrow q$

q $\sim q$

$\therefore p$ $\therefore \sim p$

1. Disebut dg ***fallacy of converse***
2. Disebut dg ***indirect reasoning***, ***modus tollens***, ***law of contrapositive***

Hukum Inferensi	*Skema*
1. Hukum Data Skema	$p \rightarrow q$ p $\therefore q$
2. Hukum kontra positif	$p \rightarrow q$ $\therefore \sim q \rightarrow \sim p$
3. Hukum Modus Tollen	$p \rightarrow q$ $\sim q$ $\therefore \sim q$
4. Aturan Rangkaian (Hukum Silogisme)	$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$
5. Hukum Inferensi Disjunktif	$p \vee q$ $\sim p$ $\therefore q$ $\quad$ $p \vee q$ $\sim q$ $\therefore p$
6. Hukum Negasi Dobel	$\sim(\sim p)$ $\therefore p$
7. Hukum De Morgan	$\sim(p \wedge q)$ $\therefore \sim p \vee \sim q$ $\quad$ $\sim(p \vee q)$ $\therefore \sim p \wedge \sim q$
8. Hukum Penyederhanaan	$p \wedge q$ $\therefore p$ $\quad$ $p \vee q$ $\therefore q$
9. Hukum Konjungsi	p q $\therefore p \wedge q$
10. Hukum Konjungsi	p $\therefore p \vee q$
11. Hukum De Morgan	$\sim(p \wedge q)$ p $\therefore \sim q$ $\quad$ $\sim(p \wedge q)$ q $\therefore \sim p$

Tabel Kondisional dan variantnya

Kondisional	$p \rightarrow q$
Konversi	$q \rightarrow p$
Invensi	$\sim p \rightarrow \sim q$
Kontrapositif	$\sim q \rightarrow \sim p$

Contoh argumen dengan lebih dari 2 promise:

Chip prices rise only if the yen rises

The yen rises only if the dollar falls and

If the dollar falls then the yen rises.

Since chip proses have risen,

the dollar must have fallen

Proposisinya C = chip prices rise

Y = yen rises

D = dollar falls

$C \rightarrow Y$

$(Y \rightarrow D) \wedge (D \rightarrow Y)$

C

$\therefore D$

Buktikan !…..

Solusi :

1. Ingat $p \rightarrow q$ dan $q \rightarrow p$ benar maka p dan q ekuivalen
2. Jika $(p \rightarrow q) \wedge (q \rightarrow p)$ maka ekuivalen dg $p \leftrightarrow q$ dg kata lain $p \equiv q$

Maka argumennya menjadi

$C \rightarrow Y$

$Y \equiv D$

C

$\therefore D$

3. Karena Y sama dengan D maka substitusi D kedalam Y

Maka argumennya menjadi :

$C \rightarrow D$

C

$\therefore D$ (TERBUKTI valid bahwa ini adalah modus ponens)

SOAL :

All men are mortal	(p)
Socrates is a man	(q)
Therefore, Socrates is mortal	$\therefore r$

Buktikan valid atau tidak ?….

FIRST ORDER PREDICATE LOGIC

Kategori silogisme dengan menggunakan predikat logik

TIPE	SKEMA	REPRESENTASI PREDIKAT
A	All S is P	$(\forall x)\ (S(x) \rightarrow P(x))$
E	No S is P	$(\forall x)\ (S(x) \rightarrow \sim P(x))$
I	Some S is P	$(\exists x)\ (S(x) \wedge P(x))$
O	Some S is not P	$(\exists x)\ (S(x) \wedge \sim P(x))$

Rule Hukum Universal Instantion menunjukkan individual yg mungkin digantikan dg universal yaitu simbol ϕ yg berarti fungsi proposisional

$(\forall x)\ \phi(x)$ x= variabel yg mengatur seluruh individual

$\therefore \phi(a)$ a= individual khusus

Contoh : Socrates is human

$(\forall x)\ H\ (x)$

$\therefore$ H (Socrates)

dimana H(x) : fungsi proposissional dg x adalah human

Contoh lain

All men are mortal

Socrates is a man

$\therefore$ Socrates is mortal

dimana H=man, M=mortal, s=socrates

Solusi :

1. $(\forall x)\ (H(x) \rightarrow M(x))$
2. H(s)
3. $\therefore$ M(s)
4. $H(s) \rightarrow M(s)$
5. M(s)

LOGIC SYSTEMS = WFFS = WFF

- ✓ Koleksi objek seperti baris, aksioma, pernyataan dsb
- ✓ Tujuan :
  1. Menentukan bentuk argumen (WFFS=Well Formed Formulas)
     Contoh All S is P
  2. Menunjukkan baris inference yg valid
  3. Mengembangkan sendiri dg menemukan baris baru dari inference shg memperluas rentangan argumen yg dapat dibuktikan
- ✓ **Aksioma** :fakta sederhana atau assertion yg tidak dapat dibuktikan dari dalam sistem

- ✓ **System formal yang diperlukan** :
  1. Alfabet simbol
  2. String finite dari simbol tertentu, wffs
  3. Aksioma, definisi system
  4. Baris inference, yang memungkinkan wff, A untuk dikurangi sebagai kesimpulan dari set finite Γ wff lain dimana Γ = {A1,A2,…An}. Wffs harus berupa aksioma atau teori lain dari sistem logis

RESOLUSI

- ✓ Diperkenalkan oleh Robinson (1965)
- ✓ Merupakan baris inference yg utama dalam prolog
- ✓ Prolog menggunakan notasi “quantifier-free”
- ✓ Prolog didasarkan pada logika predikat first-order
- ✓ Sebelum resolusi diterapkan, wff harus berada dalam keadaan normal (bentuk standar) yaitu hanya menggunakan V , $\wedge$, ~

Mis wff (A V B) $\wedge$ (~B V C) disebut bentuk normal konjungtif
A V B dan ~B V C

Ekspresi clausal umumnyya dituliskan dalam bentuk khusus yg disebut kowalski :
A1, A2, ……. AN → B1, B2, ….BM

Dalam notasi predikat standar :

A1 $\wedge$ A2 $\wedge$ ……. AN → B1 V B2 V, ….BM

Bentuk disjungsinya menggunakan

(p → q) ≡ ~p v q

menjadi :

A1 $\wedge$ A2$\wedge$ V ……. AN → B1 V B2 V, ….BM

≡ ~(A1 $\wedge$ A2 $\wedge$ ……. AN) V (B1 V B2 V, ….BM)

≡ ~A1 V ~A2 V ……. ~AN V B1 V B2 V, …. BM INGAT De Morgan ~(p $\wedge$ q) ≡ ~p v ~q

Dengan klausa Horn menjadi :

A1, A2, ……. AN → B

Dalam prolog :

B :- A1, A2, … AN

Untuk membuktikan teori benar dengan metode klasik “reductio ad absurdum” metode kontradiksi.

Tujuan resolusi adalah meng-infer klause baru “revolvent” dari 2 clause yang disebut parent clauses

Contoh

A V B

A V ~B

$\therefore \forall$ A

dapat ditulis sbb

(A V B) $\wedge$ (A V ~B)

ingat distribusi :

p V (q$\wedge$ r) $\equiv$ (p V q) $\wedge$ (p V r)

sehingga

(A V B) $\wedge$ (A V ~B) $\equiv$ A V (B$\wedge$ ~B) $\equiv$ A (resolvent)

ingat (B$\wedge$ ~B) $\equiv$ nil/null

Clause sekarang	*Resolven*	*Arti*
$p \rightarrow q$, p or $\neg p \vee q$, p	q	*Ponen Modus*
$p \rightarrow q$, $q \rightarrow r$ or $\neg p \vee q$, $\neg q \vee r$	$p \rightarrow r$ or $\neg p \vee r$	Rangkaian atau silogisme Hipotetis
$\neg p \vee q$ p $\vee$	q	Penggabungan
$\neg p \vee \neg q$, $p \vee q$	$\neg p \vee p$ or $\neg q \vee q$	Benar (Ulangan)
$\neg p$, p	nil	Salah (Kontradiksi)

SISTEM RESOLUSI DAN DEDUKSI

- Refutation adalah salah satu type pembuktian yang salah
- Contoh $A \rightarrow B$

  $B \rightarrow C$

  $C \rightarrow D$

  $\therefore A \rightarrow D$

$A \rightarrow B, B \rightarrow C, C \rightarrow D \vdash A \rightarrow B$

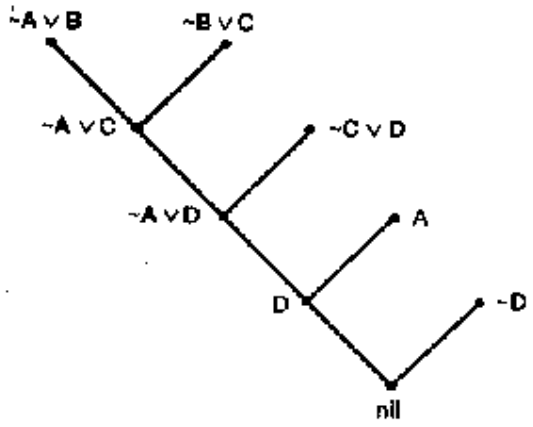
Buktikan **bahwa kesimpulan adalah teori resolusi refulasi**

Solusi :

Gunakan $(p \rightarrow q) \equiv \sim p \vee q$ untuk semua premise dan kesimpulan, kemudian negasikan untuk kesimpulannya, sehingga menjadi

$(\sim A \vee B) \wedge (\sim B \vee C) \wedge (\sim C \vee D) \wedge A \wedge \sim D$

Pohon resolusi refutation



Terbukti bahwa $A \rightarrow D$ adalah teori

Latihan :

$B \rightarrow E$

$E \wedge E$

$E \wedge S \rightarrow F$

$F \wedge G \rightarrow R$

$R \wedge T \rightarrow C$

$B \wedge S \wedge G \wedge T \rightarrow C$

RESOLUSI DAN LOGIKA PREDIKAT *FIRST ORDER*

Sebelum resolusi dapat diterapkan, wff harus diletakkan dalam bentuk casual

Contoh :

Some programmers hate all failures

No programmer hates any success

$\therefore$ No failure is a success

P(x) = x is a progammer

F(x) = x is a failure

S(x) = x is a success

H(x,y) = x hates y

Premise dan kesimpulannya

(1) $(\exists x)\,[P(x) \wedge (\forall y)\,(F(y) \rightarrow H(x,y))]$

(2) $(\forall x)\,(P(x) \rightarrow (\forall y)\,(S(y) \rightarrow \sim H(x,y))]$

(3) $\sim(\forall y)\,(F(y) \rightarrow \sim S(,y))$

Konversi ke bentuk clausal

1. Hilangkan kondisional, $(p \rightarrow q) \equiv \sim p \vee q$
2. Geser negasi ke dalam (reduksi skope ~).
   Negasi digeser hanya berlaku untuk atomik formula
3. Hilangkan quantifier eksistensial
   - Jika $\exists$ tidak ada dalam skope $\forall$, ganti variabel dengan suatu konstanta baru
     $(\exists x)$ P(x) diganti P(a)
   - Jika $\exists$ berada dalam skope $\forall$, ganti variabel dengan suatu fungsi yang memiliki argumen semua variabel dari $\forall$ tersebut

     $\forall x, \forall y, \exists z$ P(x,y,z) diganti menjadi $\forall x, \forall y$, P(x,y,F(x,y))
4. Standarisasi variabel (jika perlu) sehingga tiap quantifier memiliki variabel yang berbeda
5. Geser semua $\forall$ ke kiri (karena semua quantifier punya nama yang berbeda, pergeseran tidak mempengaruhi hasil)
   Bentuk ini disebut prenex normal form terdiri atas prefix quantifier yang diikuti matriks

6. Hilangkan $\forall$. $\forall$ tidak perlu ditulis, diasumsikan semua variabel terkuantifikasi universal
7. Geser disjungsi (V) kedalam, sehingga terbentuk conjungsi normal form
8. Buang konjungsi dan uraikan menjadi klausa-klausa
9. Standarisasi variabel (jika perlu) sehingga tidak ada variabel yang muncul pada lebih dari 1 klausa.

Contoh : Ubah ke bentuk klausal !!!!!!

$\forall x$ (Balok (x) $\rightarrow$ ($\exists y$ (Diatas(x,y) $\wedge$ ~Piramid(y))

$\wedge$ ~$\exists y$ (Diatas(x,y) $\wedge$ Diatas(y,x))

$\wedge$ $\forall y$ (~Balok(y) $\rightarrow$ ~Sama(x,y))))

Solusi :

1. $\forall x$ (~Balok (x) V ($\exists y$ (Diatas(x,y) $\wedge$ ~Piramid(y))

$\wedge$ ~$\exists y$ (Diatas(x,y) $\wedge$ Diatas(y,x))

$\wedge$ $\forall y$ (~Balok(y) V ~Sama(x,y))))

2. $\forall x$ (~Balok (x) V ($\exists y$ (Diatas(x,y) $\wedge$ ~Piramid(y))

$\wedge$ $\forall y$ (~Diatas(x,y) V~ Diatas(y,x))

$\wedge$ $\forall y$ (~Balok(y) V ~Sama(x,y))))

3. $\forall x$ (~Balok (x) V (Diatas(x,f(x)) $\wedge$ ~Piramid(f(x)))

$\wedge$ $\forall y$ (~Diatas(x,y) V~ Diatas(y,x))

$\wedge$ $\forall y$ (~Balok(y) V ~Sama(x,y))))

4. $\forall x$ (~Balok (x) V (Diatas(x,f(x)) $\wedge$ ~Piramid(f(x)))

$\wedge$ $\forall y$ (~Diatas(x,y) V~ Diatas(y,x))

$\wedge$ $\forall z$ (~Balok(z) V ~Sama(x,z))))

5. $\forall x \forall y \forall z$ (~Balok (x) V (Diatas(x,f(x)) $\wedge$ ~Piramid(f(x)))

$\wedge$ (~Diatas(x,y) V~ Diatas(y,x))

$\wedge$ (~Balok(z) V ~Sama(x,z))))

6. (~Balok (x) V ((Diatas(x,f(x)) ∧ ~Piramid(f(x)))

   ∧ (~Diatas(x,y) V~ Diatas(y,x))

   ∧ (~Balok(z) V ~Sama(x,z))))

7. (~Balok (x) V Diatas(x,f(x))

   ∧ (~Balok (x) V ~Piramid(f(x)))

   ∧ (~Balok (x) V ~Diatas(x,y) V~ Diatas(y,x))

   ∧ (~Balok (x) V ~Balok(z) V ~Sama(x,z))))

8. 1. ~Balok (x) V Diatas(x,f(x))
   2. ~Balok (x) V ~Piramid(f(x))
   3. ~Balok (x) V ~Diatas(x,y) V~ Diatas(y,x)
   4. ~Balok (x) V ~Balok(z) V ~Sama(x,z)

9. 1. ~Balok (x) V Diatas(x,f(x))
   2.~Balok (k) V ~Piramid(f(k))
   3.~Balok (m) V ~Diatas(m,y) V~ Diatas(y,m)
   4. ~Balok (n) V ~Balok(z) V ~Sama(n,z)

RANGKAIAN BACKWARD DAN FORWARD

Forward : bottom-up reasoning, breadth first

Backward : top-down reasoning, depth-first

Rangkaian forward	Rangkaian Backward
-Planning, monitoring,control	-Diagnosis
-Saat sekarang ke masa depan	-Sekarang ke masa lalu
-Antecedent ke consequent	-Consequent ke antecedent
-Data driven, bottom-up	-Goal driven, top-down
-Kerja mundur untuk menemukan pemecahan yg mengikuti fakta	-Kerja mundur untuk menemukan fakta yg mendukung hipotesa
-Breadth-first search	-Depth-first search
-Antecedent menentukan pencarian	-Consequent menentukan pencarian
-Fasilitas bukan penjelasan	-Fasilitas penjelasan

METODE LAIN DARI INFERENCE/KESIMPULAN

ANALOGI

- Mencoba dan menghubungkan situasi lama sebagai penuntun ke situasi baru.
- Contoh : diagnosis medical
- Pemberian alasan analogis berhubungan dgn induksi

GENERATE AND TEST

- Pembuatan solusi kemudian pengetesan untuk melihat apakah solusi yg diajukan memenuhi semua persyaratan. Jika solusi memenuhi maka berhenti yg lain membuat sollusi yg baru kemudian test lagi dst
- Contoh : Dendral, prog AM(artificial Mathematician),Mycin

ABDUCTION/PENGAMBILAN

- Metodenya sama dg modus ponens
- Abduction Modus ponens

Abduction	Modus ponens
$p \rightarrow q$	$p \rightarrow q$
q	p
$\therefore p$	$\therefore q$

- Bukan argument deduksi yg valid
- Berguna untuk baris/rules heuristik inference
- Analogi,generate and test, abduction adalah metode bukan deduksi. Dari premise yg benar, metode ini tidak dapat membuktikan kesimpulan yg benar

Perbedaan :

Inference	Start	Tujuan
FORWARD	Fakta	Kesimpulan yg harus mengikuti
BACKWARD	Kesimpulan tdk pasti	Fakta pendukung kesimpulan
ABDUCTION	Kesimpulan benar	Fakta yg dpt mengikuti

NONMONOTONIC REASONING

- Tambahan aksioma yg baru pada sistem logika berarti bahwa banyak teori yg dapat dibuktikan jika ada banyak aksioma dari teori yg didapat, disebut monotonik sistem

METAKNOWLEDGE

- Program meta-DENDRAL menggunakan induksi untuk menyimpulkan baris baru dari struktur kimia.
- Contoh : TEIRESIAS yg menambah pengetahuan secara interaktif dari expert